MADE: Exploration via Maximizing Deviation from Explored Regions

Tianjun Zhang[*1] **Paria Rashidinejad**[*1] **Jiantao Jiao**[1] **Yuandong Tian**[2]

Joseph E. Gonzalez[1] **Stuart Russell**[1]

[1]University of California, Berkeley
{tianjunz,paria.rashidinejad,jiantao,russell}@berkeley.edu

[2]Facebook AI Research
yuandong@fb.com

Abstract

In online reinforcement learning (RL), efficient exploration remains particularly challenging in high-dimensional environments with sparse rewards. In low-dimensional environments, where tabular parameterization is possible, count-based upper confidence bound (UCB) exploration methods achieve minimax near-optimal rates. However, it remains unclear how to efficiently implement UCB in realistic RL tasks that involve non-linear function approximation. To address this, we propose a new exploration approach via *maximizing* the deviation of the occupancy of the next policy from the explored regions. We add this term as an adaptive regularizer to the standard RL objective to balance exploration vs. exploitation. We pair the new objective with a provably convergent algorithm, giving rise to a new intrinsic reward that adjusts existing bonuses. The proposed intrinsic reward is easy to implement and combine with other existing RL algorithms to conduct exploration. As a proof of concept, we evaluate the new intrinsic reward on tabular examples across a variety of model-based and model-free algorithms, showing improvements over count-only exploration strategies. When tested on navigation and locomotion tasks from MiniGrid and DeepMind Control Suite benchmarks, our approach significantly improves sample efficiency over state-of-the-art methods.[2]

1 Introduction

Online RL is a useful tool for an agent to learn how to perform tasks, particularly when expert demonstrations are unavailable and reward information needs to be used instead [85]. To learn a satisfactory policy, an RL agent needs to effectively balance between exploration and exploitation, which remains a central question in RL [22, 14]. Exploration is particularly challenging in environments with sparse rewards. One popular approach to exploration is based on *intrinsic motivation*, often applied by adding an intrinsic reward (or bonus) to the extrinsic reward provided by the environment. In provable exploration methods, bonus often captures the value estimate uncertainty and then the agent takes an action that maximizes the upper confidence bound (UCB) [4, 7, 37, 43, 40]. In tabular setting, UCB bonuses are often constructed based on either Hoeffding's inequality, which only uses visitation

[*]Equal Contribution.

[2]Our code is available at https://github.com/tianjunz/MADE.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

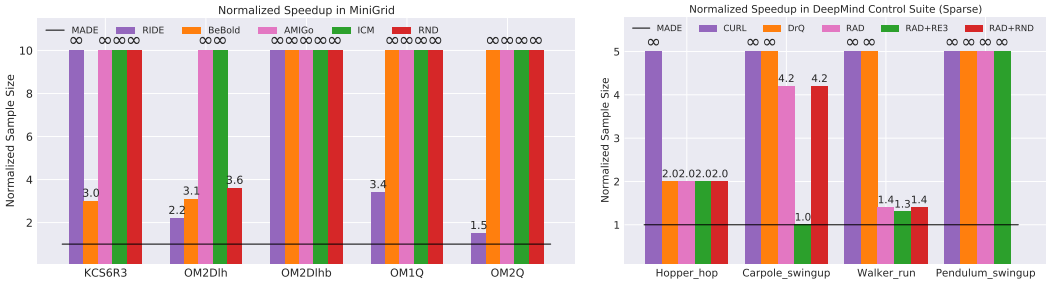

Figure 1: Normalized samples use of different methods with respect to MADE (smaller values are better). MADE consistency achieves a better sample efficiency compared to all other baselines. Infinity means the method fails to achieve maximum reward in given steps.

count, or Bernstein's inequality, which uses value function variance in addition to count. The latter is proved to be minimax near-optimal in environments with bounded rewards [40, 60] as well as bounded total reward [104] and reward-free settings [59, 44, 41, 105]. It remains an open question how one can efficiently compute confidence bounds to construct UCB bonus in non-linear function approximation. Furthermore, Bernstein-style bonuses are often hard to compute in practice beyond tabular setting.

In practice, various approaches are proposed to design intrinsic rewards: visitation pseudo-counts bonuses estimate count-based UCB bonus using function approximation [9, 14]; curiosity-based bonuses seek states where model prediction error is high; uncertainty-based bonuses [72, 80] adopt ensembles of networks for estimating variance of the Q-function; empowerment-based approaches [47, 31, 78, 63] lead the agent to states over which the agent has control, and information gain bonuses [46] that reward the agent based on the information gain between state-action pairs and next states.

Although the performance of practical intrinsic rewards is good in certain domains, empirically they are observed to suffer from issues such as detachment, derailment, and catastrophic forgetting [2, 22]. Moreover, these methods usually lack a clear objective and can get stuck in local optimum [2]. Indeed, the impressive performance currently achieved by some deep RL algorithms often revolves around manually designing dense rewards [12], complicated exploration strategies utilizing a significant amount of domain knowledge [22], or operating in the known environment regime [81, 64].

Motivated by current practical challenges and the gap between theory and practice, we propose a new algorithm for exploration by maximizing deviation from explored regions. This yields a practical algorithm with strong empirical performance. To be specific, we make the following contributions:

1. Exploration via maximizing deviation Our approach is based on modifying the standard RL objective (i.e. the cumulative reward) by adding a regularizer that adaptively changes across iterations. The regularizer can be a general function depending on the state-action visitation density and previous state-action coverage. We then choose a particular regularizer that **MA**ximizes the **DE**viation (MADE) of the next policy visitation d^π from the regions covered by prior policies ρ^k_{cov}:

$$L_k(d^\pi) = J(d^\pi) + \tau_k \sum_{s,a} \sqrt{\frac{d^\pi(s,a)}{\rho^k_{\text{cov}}(s,a)}}. \tag{1}$$

Here, k is the iteration number, $J(d^\pi)$ is the standard RL objective, and the regularizer encourages $d^\pi(s,a)$ to be large when $\rho^k_{\text{cov}}(s,a)$ is small. We give an algorithm for solving the regularized objective and prove that with access to an approximate planning oracle, it converges to the global optimum. We show that objective (1) results in an intrinsic reward that can be easily added to any RL algorithm to improve performance, as suggested by our empirical studies. Furthermore, the intrinsic reward applies a simple modification to the UCB-style bonus that considers prior visitations. This simple modification can also be added to existing bonuses in practice.

2. Tabular studies In the special case of tabular parameterization, we show that MADE only applies some simple adjustments to the Hoeffding-style count-based bonus. We compare the performance of MADE to Hoeffding and Bernstein bonuses in three different RL algorithms, for the exploration task in stochastic diabolical bidirectional lock [2, 61], which has sparse rewards and local optima. Our results show that MADE robustly improves over the Hoeffding bonus and is competitive to the Bernstein bonus, across all three RL algorithms. Interestingly, MADE bonus and exploration strategy appear to be very close to Bernstein bonus, *without computing or estimating variance*, suggesting

that MADE potentially captures some environmental structures. Additionally, we empirically show that MADE regularizer can improve the optimization rate in policy gradient methods.

3. Experiments on MiniGrid and DeepMind Control Suite We empirically show that MADE works both for model-free (IMAPLA [24], RAD [50]) and model-based (Dreamer [32]) RL algorithms, greatly improving the sample efficiency over existing baselines. When tested in MiniGrid environments, MADE manages to converge with 2x to 5x fewer samples compared to SOTA method BeBold [103]. In DeepMind Control Suite [88], we build upon the model-free method RAD [50] and the model-based method Dreamer [32], improving the return up to 150 in 500K steps compared to baselines. Figure 1 shows normalized sample size to achieve maximum reward w.r.t. our algorithm.

2 Background

Markov decision processes. An infinite-horizon discounted MDP is described by a tuple $M = (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is the (extrinsic) reward function, $\rho : \mathcal{S} \mapsto \Delta(\mathcal{S})$ is the initial distribution, and $\gamma \in [0, 1)$ is the discount factor. A stationary (stochastic) policy $\pi \in \Delta(\mathcal{A} \mid \mathcal{S})$ specifies a distribution over actions in each state. Each policy π induces a visitation density over state-action pairs $d^\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ defined as $d^\pi_\rho(s, a) \coloneqq (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \, \mathbb{P}_t(s_t = s, a_t = a; \pi)$, where $\mathbb{P}_t(s_t = s, a_t = a; \pi)$ denotes (s, a) visitation probability at step t, starting at $s_0 \sim \rho(\cdot)$ and following π. An important quantity is the value a policy π, which is the discounted sum of rewards $V^\pi(s) \coloneqq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_t \sim \pi(\cdot \mid s_t)$ for all $t \geq 0]$ starting at state $s \in \mathcal{S}$.

Policy mixture. For a sequence of policies $\mathcal{C}^k = (\pi_1, \ldots, \pi_k)$ with corresponding weights $w^k \in \Delta_{k-1}$, the policy mixture $\pi_{\text{mix},k} = (\mathcal{C}^k, w^k)$ is obtained by first sampling a policy from w^k and then following that policy over subsequent steps [33]. The mixture policy induces a state-action visitation density according to $d^{\pi_{\text{mix}}}(s, a) = \sum_{i=1}^{k} w_i^k d^{\pi_i}(s, a)$. While the π_{mix} may not be stationary in general, there exists a stationary policy π' such that $d^{\pi'} = d^{\pi_{\text{mix}}}$; see [75] for details.

Online reinforcement learning. Online RL is the problem of finding a policy with maximum value from an unknown MDP using samples collected during exploration. Oftentimes, the following objective is considered, which is a scalar summary of the performance of policy π:

$$J_M(\pi) \coloneqq \mathbb{E}_{s\sim\rho}[V^\pi(s)] = (1 - \gamma)^{-1} \, \mathbb{E}_{(s,a)\sim d^\pi_\rho(\cdot,\cdot)}[r(s, a)]. \tag{2}$$

We drop index M when it is clear from context. We denote an optimal policy by $\pi^\star \in \arg\max_\pi J(\pi)$ and use the shorthand $V^\star \coloneqq V^{\pi^\star}$ to denote the optimal value function. It is straightforward to check that $J(\pi)$ can equivalently be represented by the expectation of the reward over the visitation measure of π and slightly abuse the notation and sometimes write $J(d^\pi)$.

3 Adaptive regularization of the RL objective

3.1 Regularization to guide exploration

In online RL, the agent faces a dilemma in each state: whether it should select a seemingly optimal policy (exploit) or it should explore different regions of the MDP. To allow flexibility in this choice and trade-off between exploration and exploitation, we propose to add a regularizer to the standard RL objective that changes throughout iterations of an online RL algorithm:

$$L_k(d^\pi) = \underbrace{J(d^\pi)}_{\text{exploitation}} + \tau_k \underbrace{R(d^\pi; \{d^{\pi_i}\}_{i=1}^k)}_{\text{exploration}}. \tag{3}$$

Here, $R(d^\pi; \{d^{\pi_i}\}_{i=1}^k)$ is a function of state-action visitation of π as well as the visitation of prior policies $\pi_1, \ldots, \pi_k$. The temperature parameter τ_k determines the strength of regularization. Objective (3) is a *population* objective in the sense that it does not involve empirical estimations affected by the randomness in sample collection. In the following section, we give our particular choice of regularizer and discuss how this objective can describe some popular exploration bonuses. We then provide a convergence guarantee for the regularized objective in Section 3.2.

3.2 Exploration via maximizing deviation from policy cover

We develop our exploration strategy MADE based on a simple intuition: maximizing the deviation from the explored regions, i.e. all states and actions visited by prior policies. We define *policy cover* at iteration k to be the density over regions explored by policies $\pi_1, \ldots, \pi_k$, i.e. $\rho^k_{\text{cov}}(s,a) := \frac{1}{k}\sum_{i=1}^k d^{\pi_i}(s,a)$. We then design our regularizer to encourage d^π to be different from ρ^k_{cov}:

$$R_k(d^\pi; \{d^{\pi_i}\}_{i=1}^k) = \sum_{s,a} \sqrt{\frac{d^\pi(s,a)}{\rho^k_{\text{cov}}(s,a)}}. \tag{4}$$

It is easy to check that the maximizer of above function is $d^\pi(s,a) \propto \frac{1}{\rho^k_{\text{cov}}(s,a)}$. Our motivation behind this particular deviation is that it results in a simple modification of UCB bonus in tabular case.

We now compute the reward yielded by the new objective. First, define a policy mixture $\pi_{\text{mix},k}$ with policy sequence $(\pi, \ldots, \pi_k)$ and geometric weights $((1-\eta)^{k-1}\eta, \ldots, \eta)$ for $\eta > 0$. Let $d^{\pi_{\text{mix},k}}$ be the visitation density of $\pi_{\text{mix},k}$. We compute the total reward at iteration k by taking the gradient of new objective with respect to d^π at $d^{\pi_{\text{mix},k}}$:

$$r_k(s,a) = \nabla_d L_k(d)\big|_{d=d^{\pi_{\text{mix},k}}} = r(s,a) + \tau_k \nabla_d R_k(d; \{d^{\pi_i}\}_{i=1}^k)\big|_{d=d^{\pi_{\text{mix},k}}}, \tag{5}$$

which gives the following reward

$$r_k(s,a) = r(s,a) + \frac{\tau_k/2}{\sqrt{d^{\pi_{\text{mix},k}}(s,a)\rho^k_{\text{cov}}(s,a)}}. \tag{6}$$

The intrinsic reward above is constructed based on two densities: ρ^k_{cov} a uniform combination of past visitation densities and $\hat{d}^{\pi_{\text{mix},k}}$ a geometric mixture of past visitation densities. As we will discuss shortly, policy cover $\rho^k_{\text{cov}}(s,a)$ is related to the visitation count of (s,a) pair in previous iterations and resembles count-based bonuses [9, 40] or their approximates such as RND [14]. Therefore, for an appropriate choice of τ_k, MADE's intrinsic reward decreases as the number of visitations increases.

MADE's intrinsic reward is also proportional to $1/\sqrt{d^{\pi_{\text{mix},k}}(s,a)}$, which can be viewed as a correction implied to the count-based bonus. In effect, due to the geometric weights in $\pi_{\text{mix},k}$, the above construction gives a higher reward to (s,a) pairs visited earlier. Experimental results suggest that this correction may alleviate major difficulties in sparse reward exploration, namely detachment and catastrophic forgetting, by encouraging the agent to revisit forgotten states and actions.

Empirically, MADE's intrinsic reward is computed based on estimates $\hat{d}^{\pi_{\text{mix},k}}$ and $\hat{\rho}^k_{\text{cov}}$ from data collected by iteration k. Furthermore, practically we consider a smoothed version of the above regularizer by adding $\lambda > 0$ to both numerator and denominator; see (7).

MADE intrinsic reward in tabular case. In tabular empirical setting, the empirical estimation of policy cover is simply $\hat{\rho}^k_{\text{cov}}(s,a) = \frac{N_k(s,a)}{N_k}$, where $N_k(s,a)$ is (s,a) pair's visitation count and N_k is the total count, by iteration k. Thus, MADE simply modifies the Hoeffding-type bonus via the mixture density and has the following form: $\propto 1/\sqrt{\hat{d}^{\pi_{\text{mix},k}}(s,a)N_k(s,a)}$.

Bernstein bonus is another tabular UCB bonus that modifies Hoeffding bonus via value function variance. Bernstein bonus is shown to improve over Hoeffding count-only bonus by exploiting additional environment structure [98] and close the gap between algorithmic upper bounds and information-theoretic limits up to logarithmic factors [104, 105]. However, a practical and efficient implementation of a bonus that exploits variance information in non-linear function approximation parameterization still remains an open question; see Section 6 for further discussion. On the other hand, our proposed modification based on mixture density can be easily and efficiently incorporated with non-linear parameterization.

Connection to Convex MDP. We discuss how our algorithm is related tothe Frank-Wolfe algorithm (FW) [29]. In Eq.2, we state that the objective is concave with respect to the visitation density d^π. The objective in Eq. 1 shares similar concavity since our choice of regularizer is also concave with respect to the visitation density. Solving the objective using conditional gradient method (e.g., FW) follows the following steps: (1) Find a solution $d^{\pi_{k+1}} \in \arg\max_{d^\pi} \langle d^\pi, \nabla_{d^\pi} L_k(d^\pi)|_{d^\pi = d^{\pi_{mix,k}}} \rangle$. (2) Combine $d^{\pi_{k+1}}$ with the prior densities via geometric weights.

Algorithm 1 Policy computation for adaptively regularized objective

1: **Inputs:** Iteration count K, planning error ϵ_p, visitation density error ϵ_d.
2: Initialize policy mixture $\pi_{\text{mix},1}$ = with $\mathcal{C}_1 = (\pi_1)$ and $w^1 = (1)$
3: **for** $k = 1, \ldots, K$ **do**
4: Estimate the visitation density $\hat{d}^{\pi_{\text{mix},k}}$ of $\pi_{\text{mix},k}$ via a visitation density oracle.
5: Compute reward $r_k(s,a) = r(s,a) + \tau_k \nabla_d R_k(d; \{\pi_i\}_{i=1}^k)\big|_{d=\hat{d}^{\pi_{\text{mix},k}}}$.
6: Run approximate planning on modified MDP $M^k = (\mathcal{S}, \mathcal{A}, P, r_k, \gamma)$ and return π_{k+1}.
7: Update policy mixture $\mathcal{C}^{k+1} = (C_k, \pi_{k+1})$ and $w^{k+1} = ((1-\eta)w^k, \eta)$.
8: **Return:** $\pi_{\text{mix},K} = (\mathcal{C}^k, w^k)$.

Note that step (1) is equivalent to solving a planning via reward function $\nabla_{d^\pi} L_k(d^\pi)|_{d^\pi = d^{\pi_{mix,k}}}$. Thus, we naturally arrive a reward function of Eq. 5. We didn't implement the FW algorithm but instead, we compare with RE3 [79], which is a similar method to the algorithm of [33] and adds state visitation entropy as a regularizer to conduct exploration.

Deriving some popular bonuses from regularization. We now discuss how the regularization in (3) can describe some popular bonuses. Exploration bonuses that only depend on state-action visitations can be expressed in the form (3) by setting the regularizer a linear function of d^π and the exploration bonus $r_i(s,a)$, i.e., $R_k(d^\pi; \{d^{\pi_i}\}_{i=1}^k) = \sum_{s,a} d^\pi(s,a) r_i(s,a)$. It is easy to check that taking the gradient of the regularizer with respect to d^π recovers $r_i(s,a)$. As another example, one can set the regularizer to Shannon entropy $R_k(d^\pi; \{d^{\pi_i}\}_{i=1}^k) = -\sum_{s,a} d^\pi(s,a) \log d^\pi(s,a)$, which gives the intrinsic reward $-\log d^\pi(s,a)$ (up to an additive constant) and recovers the result in [101].

3.3 Solving the regularized objective

We pair MADE's objective with the algorithm proposed by Hazan et al. [33] extended to the adaptive objective. We provide convergence guarantee for Algorithm 1 in the following theorem whose proof is given in Appendix A.

Theorem 1. *Consider the following regularization for $\lambda > 0$ for a valid visitation density d*

$$R_\lambda(d; \{\pi_i\}_{i=1}^{k-1}) = \sum_{s,a} \sqrt{\frac{d(s,a) + \lambda}{\rho_{cov}(s,a) + \lambda}}, \tag{7}$$

with $\tau_k = \tau / k^c$ where $0 < \tau < 1$ and $c > 0$. Then, for any $\epsilon > 0$, there exists $\eta, \epsilon_p, \epsilon_d, c, B$ such that $\pi_{\text{mix},K}$ returned by Algorithm 1 after $K \geq \eta^{-1} \log(10B\epsilon^{-1})$ iteration satisfies $L_K(d^{\pi_{\text{mix},K}}) \geq \max_\pi L_K(d^\pi) - \epsilon$.

Remark 1. *One does not need to maintain the functional forms of past policies to estimate $\hat{d}^{\pi_{\text{mix},k}}$. Practically, one may truncate the dataset to a (prioritized) buffer and estimate the density over that.*

4 A tabular study

4.1 Exploration in bidirectional lock

We consider a stochastic version of bidirectional diabolical combination lock (Figure 2), which is considered a particularly difficult exploration task in tabular setting [61, 2]. This environment is challenging because: (1) positive rewards are sparse, (2) a small negative reward is given when transiting to a good state and thus, moving to dead states is locally optimal, and (3) the agent may forget to explore one chain and get stuck in local minima upon receiving an end reward [2].

RL algorithms and exploration strategies. We compare Hoeffding and Bernstein bonuses [40] with MADE in three different RL algorithms. To implement MADE in tabular setting, we simply use two buffers: one stores all past state-action pairs to estimate ρ_{cov}; another one only maintains the most recent B pairs to estimate d_μ^π. We use empirical counts to estimate both densities, which give a bonus $\propto 1/\sqrt{N_k(s,a)B_k(s,a)}$, where $N_k(s,a)$ is the total count and $B_k(s,a)$ is the recent buffer count of (s,a) pair. We combine three bonuses with three RL algorithms: (1) value iteration with bonus [34], (2) proximal policy optimization with a model [15], and (3) Q-learning with bonus [40].

Results. Figure 2 summarizes our results showing MADE improves over Hoeffding bonus and is competitive to the Bernstein bonus in all three algorithms. Unlike Bernstein bonus that is hard to compute beyond tabular setting, MADE design is simple and can be effectively combined with any deep RL algorithms. The experimental results suggest several interesting properties for MADE. First, MADE applies a simple modification to Hoeffding bonus which improves the performance. Second, as illustrated in Figure 3, bonus values and exploration pattern of MADE is somewhat similar to the Bernstein bonus. This suggests that MADE may capture some structural information of the environment, similar to Bernstein bonus [98].

4.2 Policy gradient in chain MDP

We consider the chain MDP (Figure 4) presented in [1], which suffers from vanishing gradients with policy gradient approach as a positive reward is only achieved if the agent always takes action a_1 [86]. This leads to an exponential iteration complexity lower bound even with access to exact gradients [1]. In this environment, the agent always starts at state s_0. Therefore, recent guarantees on the global convergence of exact policy gradients whose rate depends on the distribution shift [10, 1, 57], i.e. the ratio of the optimal visitation density and initial distribution, are vacuous.

RL algorithms. Since our goal in this experiment is to investigate the optimization effects and not exploration, we assume access to exact gradients.= In this setting, we consider MADE regularizer with the form $\sum_{s,a} \sqrt{d^\pi(s,a)}$. Note that policy gradients take gradient of the objective w.r.t. policy parameters θ not d^π. We compare optimizing the policy gradient objective with four methods: vanilla version PG (e.g. uses policy gradient theorem [92, 86, 48]), relative policy entropy regularization PG+RE [1], policy entropy regularization PG+E [62, 57], and MADE regularization.

Results. Figure 4 illustrates our results on policy gradient methods. As expected [1], the vanilla version has a very slow convergence rate. Both entropy and relative entropy regularization are proved to achieve a linear convergence rate of $\exp(-t)$ in the iteration count t [57, 1]. Interestingly, MADE seems to also enjoy a similar rate and outperforms the policy entropy regularizers.

5 Experiments on MiniGrid and DeepMind Control Suite

In addition to tabular setting, MADE can also be integrated with various model-free and model-based Deep RL training algorithms like IMPALA [24], RAD [52] and Dreamer [32]. The performance of MADE on MiniGrid [18] and DeepMind Control Suite [88] achieve SOTA sample efficiency.

To estimate $N_k(s,a)$ and $d(s,a)$, we adopt the two buffer idea from tabular setting. However, since now the state space is high-dimensional, we use RND [14] to estimate $N_k(s,a)$ and use a VAE to estimate $d(s,a)$. Specifically, for RND, we minimize the difference between a predictor network $\phi'(s,a)$ and a randomly initialized target network $\phi(s,a)$ and train it in an online manner as the agent collects data. For VAE, we sample from the recent buffer B to train a VAE. The length of B is a design choice for which we do an ablation study. The intrinsic reward takes the form: $\tau_k \|\phi(s,a) - \phi'(s,a)\| \sqrt{1/d(s,a)}$.

Model-free RL baselines. We consider several baselines in MiniGrid: **IMPALA** [24] is a variant of policy gradient algorithms which we as the training baseline; **ICM** [71] learns a forward and reverse model for predicting state transition and uses the forward model prediction error as intrinsic

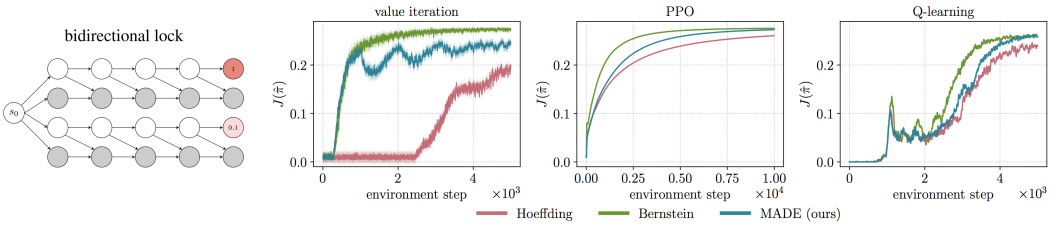

Figure 2: In a stochastic bidirectional lock, the agent starts at s_0 and enters one of the chains based on the selected action. Each chain has a positive reward at the end, H good states, and H dead states. Both actions available to the agent lead it to the dead state, one with probability one and the other with probability $p < 1$. MADE performs better than Hoeffding-style bonus and comparable to Bernstein-style bonus across all three RL algorithms.

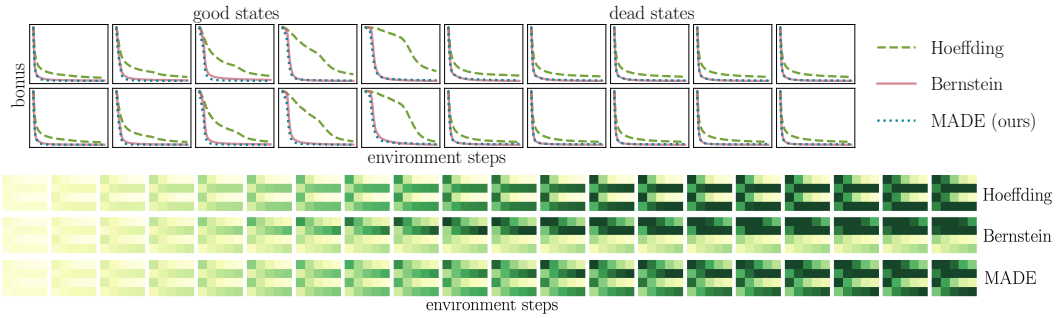

Figure 3: Hoeffding, Bernstein, and MADE intrinsic rewards over iterations in a bidirectional lock.

reward; **RND** [14] works as aforementioned; **RIDE** [76] learns a representation similar to ICM and uses the difference of learned representation along a trajectory as intrinsic reward; **AMIGo** [16] learns a teacher agent to assign intrinsic reward; **BeBold** [103] adopts a regulated difference of novelty measure using RND. In DeepMind Control Suite, we consider **RE3** [79] as a baseline which uses a random encoder for state embedding followed by a k-nearest neighbour bonus for a maximum state coverage objective.

Model-based RL baselines. MADE can be combined with model-based RL algorithms and improve sample efficiency. For baselines: **Dreamer** is a well-known model-based RL algorithm used in DeepMind Control Suite; **Dreamer+RE3** build upon Dreamer plus RE3 bonus.

MADE achieves strong SOTA results on both tasks, greatly improving the sample efficiency of the RL exploration in both model-free and model-based methods. Further details on experiments and exact hyperparameters are provided in Appendix B.

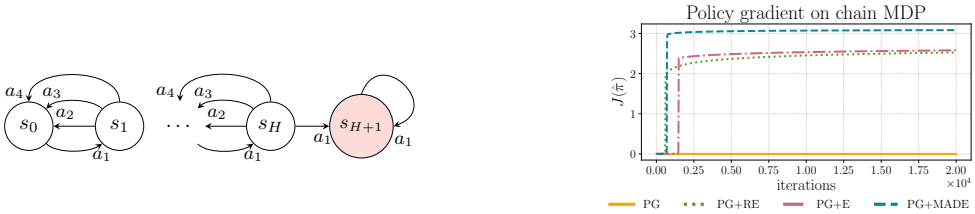

Figure 4: A deterministic chain MDP that suffers from vanishing gradients [1]. We consider a constrained tabular policy parameterization with $\pi(a|s) = \theta_{s,a}$ and $\sum_a \theta_{s,a} = 1$. The agent always starts from s_0 and the only non-zero reward is $r(s_{H+1}, a_1) = 1$.

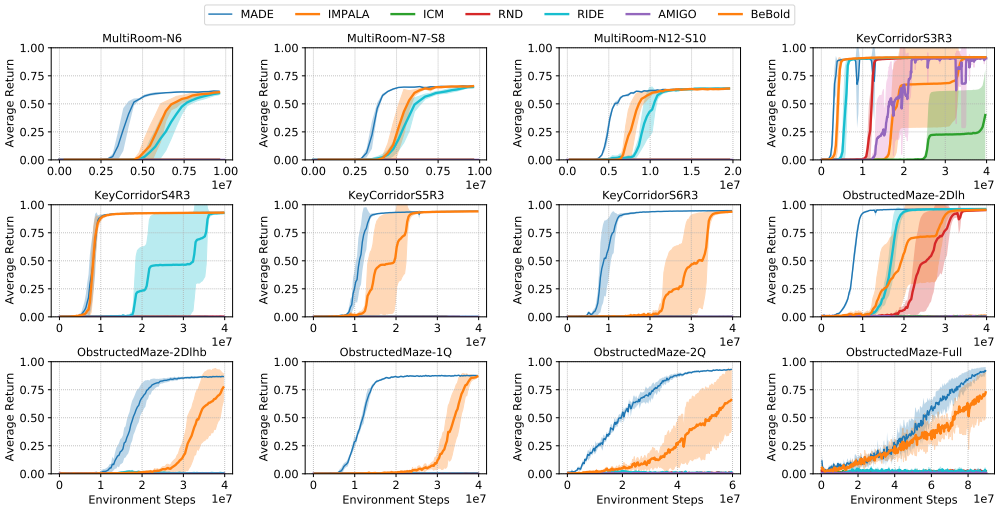

Figure 5: Results for various hard exploration environments from MiniGrid. MADE successfully solves all the environments while all the baselines except for BeBold manage to solve all of them. MADE finds the optimal solution using 2x-5x fewer samples, yields a much better sample efficiency.

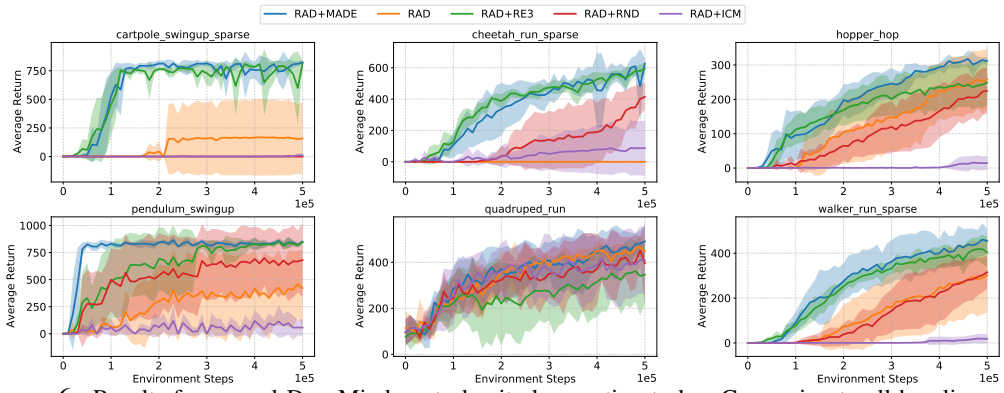

Figure 6: Results for several DeepMind control suite locomotion tasks. Comparing to all baselines, the performance of MADE is consistently better. Sometimes baseline methods even fail to solve the task.

5.1 Model-free RL on MiniGrid

MiniGrid [18] is a widely used benchmark for exploration in RL. Despite having symbolic states and a discrete action space, MiniGrid tasks are quite challenging: the easiest task is **MultiRoom**(MR) in which the agent needs to navigate to the goal by going to different rooms connected by the doors; in **KeyCorridor**(KC), the agent needs to search around different rooms to find the key and then use it to open the door; in **ObstructedMaze**(OM) is a harder version of Kc where the key is hidden in a box and sometimes the door is blocked by an obstruct. In addition to that, the entire environment is procedurally-generated. This adds another layer of difficulty to the problem.

From Figure 5 we can see that MADE manages to solve all the challenging tasks within 90M steps while all the baseline (except BeBold) only solves up to 50% of them. Compared to BeBold, MADE uses significantly fewer (**2x** to **5x** reduction) samples.

5.2 Model-free RL on DeepMind Control

We also test MADE on image-based continuous control tasks of DeepMind Control Suite [88], which is a collection of diverse control tasks such as Pendulum, Hopper, and Acrobot with realistic simulations. Compared to MiniGrid, these tasks are more realistic and complex as they involve stochastic transitions, high-dimensional states, and continuous actions. For baselines, we build our algorithm on top of RAD [52], a strong model-free RL algorithm with a competitive sample efficiency. We compare with ICM, RND, and RE3 [79] (the SOTA algorithm) [3]. Note that we compare MADE to very strong baselines; other algorithms (e.g., DrQ [49], CURL [82], ProtoRL [97], SAC+AE [96]) perform worse.[4] MADE show consistent improvement in sample efficiency: **2.6x** over RAD+RE3, **3.3x** over RAD+RND, **19.7x** over CURL, **15.0x** over DrQ and **3.8x** over RAD.

From Figure 6, we can see that MADE consistently improves sample efficiency compared to all baselines. For these tasks, RND and ICM do not perform well and even fail on `Cartpole-Swingup`. RE3 achieves a comparable performance in two tasks, however, its performance on `Pendulum-Swingup`, `Quadruped-Run`, `Hopper-Hop` and `Walker-Run` is significantly worse than MADE. For example, in `Pendulum-Swingup`, MADE achieves a reward of around 800 in only 30K steps while RE3 requires 300k samples. In `Quadruped-Run`, there is a 150 reward gap between MADE and RE3 and it seems to be still enlarging. These tasks show the strong performance of MADE in model-free RL.

Ablation study. We study how the buffer length affects the performance of our algorithm in some DeepMind Control tasks. Results show that for different tasks the optimal length is slightly different. We empirically found that using a buffer length of 1000 consistently works well across different tasks.

5.3 Model-based RL on DeepMind Control

We also empirically verify the performance of MADE combined with the SOTA model-based RL algorithm Dreamer [32]. We compare MADE with Dreamer and Dreamer combined with RE3 in

[3] We try to implement ICM and RND ourselves. The performance for ICM is slightly worse than what the author reported, but the performance of RND and RE3 is similar.

[4] We compare against them based on the numbers reported in their original papers.

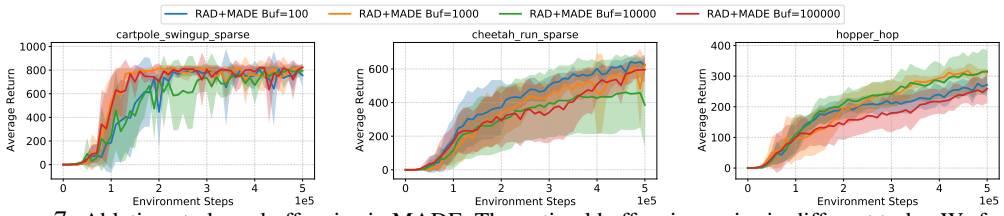

Figure 7: Ablation study on buffer size in MADE. The optimal buffer size varies in different tasks. We found buffer size of 10000 empirically works consistently reasonable.

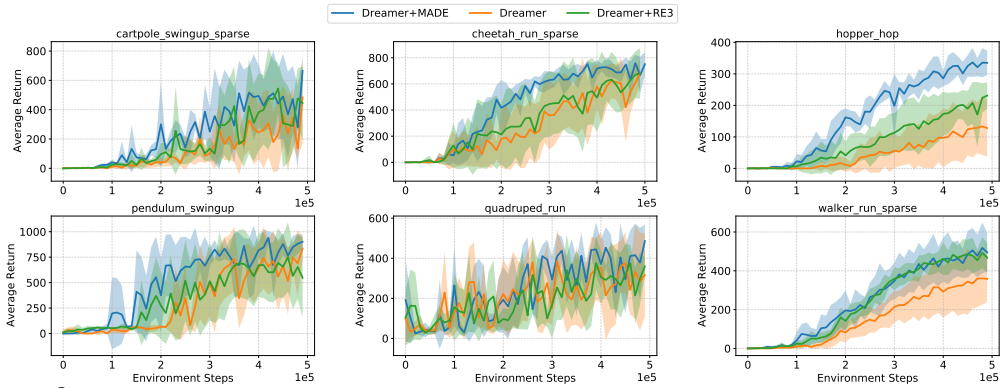

Figure 8: Results for DeepMind control suite locomotion tasks in model-based RL setting. Comparing to all baselines, the performance of MADE is consistently better. Some baseline methods even fail to solve the task.

Figure 8. Results show that MADE has great sample efficiency in maps like `Cheetah-Run-Sparse`, `Hopper-Hop` and `Pendulum-Swingup`. For example, in `Hopper-Hop`, MADE achieves more than 100 higher return than RE3 and 250 higher than Dreamer, achieving a new SOTA result.

6 Related work

Provable optimistic exploration. Most provable exploration strategies are based on optimism in the face of uncertainty (OFU) principle. In tabular setting, model-based exploration algorithms include variants of UCB [45, 11], UCRL [51, 37, 98, 44, 59], and Thompson sampling [94, 4, 77] and value-based methods include optimistic Q-learning [40, 91, 83, 55, 60] and value-iteration with UCB [7, 104, 105, 41]. These methods are recently extended to linear MDP setting leading to a variety of model-based [106, 6, 38, 107], value-based [90, 42], and policy-based algorithms [15, 100, 2]. Going beyond linear function approximation, systematic exploration strategies are developed based on structural assumptions on MDP such as low Bellman rank [39] and block MDP [21]. These methods are either computationally intractable [39, 84, 6, 99, 95, 20, 89] or are only oracle efficient [25, 3]. The recent work [26] provides a sample efficient approach with non-linear policies, however, the algorithm requires maintaining the functional form of all prior policies.

Practical exploration via intrinsic reward. Apart from previously-discussed methods, other works give intrinsic reward based on the difference in (abstraction of) consecutive states [102, 56, 76]. However, this approach is inconsistent: the intrinsic reward does not converge to zero and thus, even with infinite samples, the final policy does not maximize the RL objective. Other intrinsic rewards try to estimate pseudo-counts [9, 87, 14, 13, 70, 8], inspired by provable count-based methods. Though favoring novel states, practically these methods might suffer from *detachment and derailment* [22, 23], and *forgetting* [2]. More recent works propose a combination of different criteria. RIDE [76] learns a representation using curiosity criterion and uses the difference of consecutive states along the trajectory as the bonus. AMIGo [16] learns a teacher agent for assigning rewards for exploration. Go-Explore [22] explicitly decouples the exploration and exploitation stage, yields a more sophisticated algorithm with many hand-tuned hyperparameters.

Maximum entropy exploration. Another line of work encourages exploration via maximizing some type of entropy. One category maximizes policy entropy [62] or relative entropy [1] in addition to the RL objective. Recently, effects of policy entropy regularization have been studied theoretically [67, 30] in policy gradient methods with access to exact gradients, showing better convergence rate by improving optimization landscape [57, 58, 5, 17]. Another category considers maximizing the

entropy of state or state-action occupancy densities such as Shannon entropy [33, 36, 53, 79] or Renyi entropy [101]. Empirically, our approach achieves better performance over entropy-based methods.

Other exploration strategies. Besides intrinsic motivation, other strategies are also fruitful in encouraging the RL agent to visit a wide range of states. One example is exploration by injecting noise to the action action space [54, 68, 35, 69] or parameter space [28, 73]. Another example is the reward-shaping category, in which diverse goals are set to guide exploration [19, 27, 66, 74].

7 Discussion

We introduce a new exploration strategy MADE based on maximizing deviation from explored regions. We show that by simply adding a regularizer to the original RL objective, we get an easy-to-implement intrinsic reward and it can be incorporated with any RL algorithm. We provide a policy computation algorithm for this objective and prove that it converges to a global optimum. In tabular setting, MADE consistently improves the Hoeffding-style bonus and shows competitive performance compared to Bernstein-style bonus, while the latter is impractical to compute beyond tabular. We conduct extensive experiments on MiniGrid, showing a significant reduction of the required sample size by over $5\mathrm{x}$. MADE also performs well in DeepMind Control Suite with both model-free and model-based RL algorithms, achieving SOTA sample efficiency results. One limitation of MADE is that it only uses the naive representations of states (e.g., one-hot representation in tabular case). In fact, the exploration could be much more efficient if MADE is implemented with a more compact representation of states. We leave such analysis to future work.

8 Acknowledgements

The authors are grateful to Andrea Zanette for helpful discussions. The authors thank Alekh Agarwal, Michael Henaff, Sham Kakade, and Wen Sun for providing their code. Paria Rashidinejad is partially supported by the Open Philanthropy Foundation, Intel, and the Leverhulme Trust. Jiantao Jiao is partially supported by NSF grants IIS-1901252, CCF-1909499, and DMS-2023505. Tianjun Zhang is supported by the BAIR Commons at UC-Berkeley and thanks Commons sponsors for their support. In addition to NSF CISE Expeditions Award CCF-1730628, UC Berkeley research is supported by gifts from Alibaba, Amazon Web Services, Ant Financial, CapitalOne, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, Nvidia, Scotiabank, Splunk and VMware.

References

[1] Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *arXiv preprint arXiv:1908.00261*, 2019.

[2] Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. PC-PG: Policy cover directed exploration for provable policy gradient learning. *arXiv preprint arXiv:2007.08459*, 2020.

[3] Alekh Agarwal, Sham Kakade, Akshay Krishnamurthy, and Wen Sun. Flambe: Structural complexity and representation learning of low rank MDPs. *arXiv preprint arXiv:2006.10814*, 2020.

[4] Shipra Agrawal and Randy Jia. Optimistic posterior sampling for reinforcement learning: Worst-case regret bounds. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1184–1194, 2017.

[5] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International Conference on Machine Learning*, pages 151–160. PMLR, 2019.

[6] Alex Ayoub, Zeyu Jia, Csaba Szepesvari, Mengdi Wang, and Lin Yang. Model-based reinforcement learning with value-targeted regression. In *International Conference on Machine Learning*, pages 463–474. PMLR, 2020.

[7] Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, pages 263–272. PMLR, 2017.

[8] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, et al. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038*, 2020.

[9] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

[10] Jalaj Bhandari and Daniel Russo. Global optimality guarantees for policy gradient methods. *arXiv preprint arXiv:1906.01786*, 2019.

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

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[13] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

[14] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[15] Qi Cai, Zhuoran Yang, Chi Jin, and Zhaoran Wang. Provably efficient exploration in policy optimization. In *International Conference on Machine Learning*, pages 1283–1294. PMLR, 2020.

[16] Andres Campero, Roberta Raileanu, Heinrich Küttler, Joshua B Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with AMIGo: Adversarially motivated intrinsic goals. *arXiv preprint arXiv:2006.12122*, 2020.

[17] Shicong Cen, Chen Cheng, Yuxin Chen, Yuting Wei, and Yuejie Chi. Fast global convergence of natural policy gradient methods with entropy regularization. *arXiv preprint arXiv:2007.06558*, 2020.

[18] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for OpenAI Gym. `https://github.com/maximecb/gym-minigrid`, 2018.

[19] Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. CURIOUS: Intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

[20] Kefan Dong, Jiaqi Yang, and Tengyu Ma. Provable model-based nonlinear bandit and reinforcement learning: Shelve optimism, embrace virtual curvature. *arXiv preprint arXiv:2102.04168*, 2021.

[21] Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient RL with rich observations via latent state decoding. In *International Conference on Machine Learning*, pages 1665–1674. PMLR, 2019.

[22] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Go-explore: A new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[23] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return then explore. *arXiv preprint arXiv:2004.12919*, 2020.

[24] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-RL with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[25] Fei Feng, Ruosong Wang, Wotao Yin, Simon S Du, and Lin Yang. Provably efficient exploration for reinforcement learning using unsupervised learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[26] Fei Feng, Wotao Yin, Alekh Agarwal, and Lin F Yang. Provably correct optimization and exploration with non-linear policies. *arXiv preprint arXiv:2103.11559*, 2021.

[27] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

[28] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *International Conference on Learning Representations*, 2018.

[29] Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[30] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169. PMLR, 2019.

[31] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[32] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[33] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691, 2019.

[34] Jiafan He, Dongruo Zhou, and Quanquan Gu. Nearly minimax optimal reinforcement learning for discounted MDPs. *arXiv preprint arXiv:2010.00587*, 2020.

[35] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *arXiv preprint arXiv:1710.02298*, 2017.

[36] Riashat Islam, Zafarali Ahmed, and Doina Precup. Marginalized state distribution entropy regularization in policy optimization. *arXiv preprint arXiv:1912.05128*, 2019.

[37] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4), 2010.

[38] Zeyu Jia, Lin Yang, Csaba Szepesvari, and Mengdi Wang. Model-based reinforcement learning with value-targeted regression. In *Learning for Dynamics and Control*, pages 666–686. PMLR, 2020.

[39] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2017.

[40] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is Q-learning provably efficient? In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4868–4878, 2018.

[41] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pages 4870–4879. PMLR, 2020.

[42] Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

[43] Sham Kakade, Mengdi Wang, and Lin F Yang. Variance reduction methods for sublinear reinforcement learning. *arXiv preprint arXiv:1802.09184*, 2018.

[44] Emilie Kaufmann, Pierre Ménard, Omar Darwiche Domingues, Anders Jonsson, Edouard Leurent, and Michal Valko. Adaptive reward-free exploration. In *Algorithmic Learning Theory*, pages 865–891. PMLR, 2021.

[45] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2):209–232, 2002.

[46] Hyoungseok Kim, Jaekyeom Kim, Yeonwoo Jeong, Sergey Levine, and Hyun Oh Song. Emi: Exploration with mutual information. *arXiv preprint arXiv:1810.01176*, 2018.

[47] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. All else being equal be empowered. In *European Conference on Artificial Life*, pages 744–753. Springer, 2005.

[48] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014. Citeseer, 2000.

[49] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[50] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[51] Tor Lattimore and Marcus Hutter. PAC bounds for discounted MDPs. In *International Conference on Algorithmic Learning Theory*, pages 320–334. Springer, 2012.

[52] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *arXiv preprint arXiv:1910.05396*, 2019.

[53] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[54] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[55] Shuang Liu and Hao Su. Regret bounds for discounted MDPs. *arXiv preprint arXiv:2002.05138*, 2020.

[56] Kenneth Marino, Abhinav Gupta, Rob Fergus, and Arthur Szlam. Hierarchical RL using an ensemble of proprioceptive periodic policies. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=SJz1x20cFQ`.

[57] Jincheng Mei, Chenjun Xiao, Csaba Szepesvari, and Dale Schuurmans. On the global convergence rates of softmax policy gradient methods. In *International Conference on Machine Learning*, pages 6820–6829. PMLR, 2020.

[58] Jincheng Mei, Yue Gao, Bo Dai, Csaba Szepesvari, and Dale Schuurmans. Leveraging non-uniformity in first-order non-convex optimization. *arXiv preprint arXiv:2105.06072*, 2021.

[59] Pierre Ménard, Omar Darwiche Domingues, Anders Jonsson, Emilie Kaufmann, Edouard Leurent, and Michal Valko. Fast active learning for pure exploration in reinforcement learning. *arXiv preprint arXiv:2007.13442*, 2020.

[60] Pierre Menard, Omar Darwiche Domingues, Xuedong Shang, and Michal Valko. UCB momentum Q-learning: Correcting the bias without forgetting. *arXiv preprint arXiv:2103.01312*, 2021.

[61] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pages 6961–6971. PMLR, 2020.

[62] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[63] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 2125–2133, 2015.

[64] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[65] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2772–2782, 2017.

[66] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in Neural Information Processing Systems*, 31:9191–9200, 2018.

[67] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized Markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

[68] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[69] Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

[70] Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

[71] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[72] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019.

[73] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *International Conference on Learning Representations*, 2018.

[74] Vitchyr Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-Fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning*, pages 7783–7792. PMLR, 2020.

[75] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[76] Roberta Raileanu and Tim Rocktäschel. Ride: Rewarding impact-driven exploration for procedurally-generated environments. *arXiv preprint arXiv:2002.12292*, 2020.

[77] Daniel Russo. Worst-case regret bounds for exploration via randomized value functions. *arXiv preprint arXiv:1906.02870*, 2019.

[78] Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment: An introduction. In *Guided Self-Organization: Inception*, pages 67–114. Springer, 2014.

[79] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. *arXiv preprint arXiv:2102.09430*, 2021.

[80] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pages 5779–5788, 2019.

[81] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *nature*, 550(7676):354–359, 2017.

[82] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[83] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888, 2006.

[84] Wen Sun, Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Model-based RL in contextual decision processes: PAC bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pages 2898–2933. PMLR, 2019.

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

[86] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, pages 1057–1063, 1999.

[87] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems*, pages 2753–2762, 2017.

[88] Yuval Tassa, Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Piotr Trochim, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, et al. dm_control: Software and tasks for continuous control. *arXiv preprint arXiv:2006.12983*, 2020.

[89] Ruosong Wang, Russ R Salakhutdinov, and Lin Yang. Reinforcement learning with general value function approximation: Provably efficient approach via bounded Eluder dimension. *Advances in Neural Information Processing Systems*, 33, 2020.

[90] Yining Wang, Ruosong Wang, Simon S Du, and Akshay Krishnamurthy. Optimism in reinforcement learning with generalized linear function approximation. *arXiv preprint arXiv:1912.04136*, 2019.

[91] Yuanhao Wang, Kefan Dong, Xiaoyu Chen, and Liwei Wang. Q-learning with UCB exploration is sample efficient for infinite-horizon MDP. In *International Conference on Learning Representations*, 2019.

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

[93] Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[94] Zhihan Xiong, Ruoqi Shen, and Simon S Du. Randomized exploration is near-optimal for tabular MDP. *arXiv preprint arXiv:2102.09703*, 2021.

[95] Zhuoran Yang, Chi Jin, Zhaoran Wang, Mengdi Wang, and Michael I Jordan. Bridging exploration and general function approximation in reinforcement learning: Provably efficient kernel and neural value iterations. *arXiv preprint arXiv:2011.04622*, 2020.

[96] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[97] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021.

[98] Andrea Zanette and Emma Brunskill. Tighter problem-dependent regret bounds in reinforcement learning without domain knowledge using value function bounds. In *International Conference on Machine Learning*, pages 7304–7312. PMLR, 2019.

[99] Andrea Zanette, Alessandro Lazaric, Mykel Kochenderfer, and Emma Brunskill. Learning near optimal policies with low inherent Bellman error. In *International Conference on Machine Learning*, pages 10978–10989. PMLR, 2020.

[100] Andrea Zanette, Ching-An Cheng, and Alekh Agarwal. Cautiously optimistic policy optimization and exploration with linear function approximation. *arXiv preprint arXiv:2103.12923*, 2021.

[101] Chuheng Zhang, Yuanying Cai, and Longbo Huang Jian Li. Exploration by maximizing Rényi entropy for reward-free RL framework. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[102] Jingwei Zhang, Niklas Wetzel, Nicolai Dorka, Joschka Boedecker, and Wolfram Burgard. Scheduled intrinsic drive: A hierarchical take on intrinsically motivated exploration. *arXiv preprint arXiv:1903.07400*, 2019.

[103] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. BeBold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*, 2020.

[104] Zihan Zhang, Xiangyang Ji, and Simon S Du. Is reinforcement learning more difficult than bandits? a near-optimal algorithm escaping the curse of horizon. *arXiv preprint arXiv:2009.13503*, 2020.

[105] Zihan Zhang, Yuan Zhou, and Xiangyang Ji. Almost optimal model-free reinforcement learning via reference-advantage decomposition. *Advances in Neural Information Processing Systems*, 33, 2020.

[106] Dongruo Zhou, Quanquan Gu, and Csaba Szepesvari. Nearly minimax optimal reinforcement learning for linear mixture Markov decision processes. *arXiv preprint arXiv:2012.08507*, 2020.

[107] Dongruo Zhou, Jiafan He, and Quanquan Gu. Provably efficient reinforcement learning for discounted MDPs with feature mapping. *arXiv preprint arXiv:2006.13165*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We claim to propose efficient exploration criterion that achieves strong results. The experiments are conducted in MiniGrid, and DeepMind Control.
   (b) Did you describe the limitations of your work? [Yes] We discuss limitations in the Discussion section.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not foresee any obvious negative societal impacts from our work, which contributes a general-purpose exploration algorithm.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] We state the full set of assumptions
   (b) Did you include complete proofs of all theoretical results? [Yes] The complete proof is in Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We upload the code in supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We include the training detail in supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We include the error bars for experiments.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use publicly available datasets/tasks.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We don't use data of this sort.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Convergence analysis of Algorithm 1

In this section, we provide a convergence rate analysis for Algorithm 1. Similar to Hazan et al. [33], Algorithm 1 has access to an approximate density oracle and an approximate planner defined below:

- *Visitation density oracle:* We assume access to an approximate density estimator that takes in a policy π and a density approximation error $\epsilon_d \geq 0$ as inputs and returns $\hat{d}^\pi$ such that $\|d^\pi - \hat{d}^\pi\|_\infty \leq \epsilon_d$.
- *Approximate planning oracle:* We assume access to an approximate planner that, given any MDP M and error tolerance $\epsilon_p \geq 0$, returns a policy π such that $J_M(\pi) \geq \max_\pi J_M(\pi) - \epsilon_p$.

A.1 Proof of Theorem 1

We first give the following proposition that captures certain properties of the proposed objective. The proof is postponed to the end of this section.

Proposition 1. *Consider the following regularization for $\lambda > 0$*

$$R_\lambda(d; \{d^{\pi_i}\}_{i=1}^{k-1}) = \sum_{s,a} \sqrt{\frac{d(s,a)+\lambda}{\rho_{cov}(s,a)+\lambda}},$$

with $\tau_k = \tau/k^c$ where $\tau < 1, c > 0$. There exist constants β, B, and ξ that only depend on MDP parameters and λ such that $L_k(d) := J(d) + \tau_k R_\lambda(d; \{d^{\pi_i}\}_{i=1}^{k-1})$ satisfies the following regularity conditions for all $k \geq 1$, an appropriate choice of c, and valid visitation densities d and d′:

(i) $L_k(d)$ is concave in d;

(ii) $L_k(d)$ is β-smooth: $\|\nabla L_k(d) - \nabla L_k(d')\|_\infty \leq \beta \|d - d'\|_\infty, \quad -\beta I \preceq \nabla^2 L_k(d) \preceq \beta I$;

(iii) $L_k(d)$ is B-bounded: $L_k(d) \leq B, \quad \|\nabla L_k(d)\|_\infty \leq B$;

(iv) There exists δ_k such that $\max_d L_{k+1}(d) - L_k(d) \leq \delta_k$ and we have $\sum_{i=0}^k (1-\eta)^i \delta_{k-i} \leq \tau\xi$.

Taking the above proposition as given for the moment, we prove Theorem 1 following steps similar to those of Hazan et al. [33, Theorem 4.1]. By construction of the mixture density $d^{\pi_{\text{mix},k}}$, we have

$$d^{\pi_{\text{mix},k}} = (1-\eta) d^{\pi_{\text{mix},k-1}} + \eta d^{\pi_k}.$$

Combining the above equation with the β-smoothness of $L_k(d)$ yields

$$\begin{aligned} L_k(d^{\pi_{\text{mix},k}}) &= L_k((1-\eta) d^{\pi_{\text{mix},k-1}} + \eta d^{\pi_k}) \\ &\geq L_k(d^{\pi_{\text{mix},k-1}}) + \eta \langle d^{\pi_k} - d^{\pi_{\text{mix},k-1}}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle - \eta^2 \beta \|d^{\pi_k} - d^{\pi_{\text{mix},k-1}}\|_2^2 \\ &\geq L_k(d^{\pi_{\text{mix},k-1}}) + \eta \langle d^{\pi_k} - d^{\pi_{\text{mix},k-1}}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle - 4\eta^2 \beta. \end{aligned} \tag{8}$$

Here the last inequality uses $\|d^{\pi_k} - d^{\pi_{\text{mix},k-1}}\|_2 \leq 2$. By property (ii), we bound $\langle d^{\pi_k}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle$ according to

$$\begin{aligned} \langle d^{\pi_k}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle &\geq \langle d^{\pi_k}, \nabla L_k(\hat{d}^{\pi_{\text{mix},k-1}}) \rangle - \beta \|d^{\pi_{\text{mix},k-1}} - \hat{d}^{\pi_{\text{mix},k-1}}\|_\infty \\ &\geq \langle d^{\pi_k}, \nabla L_k(\hat{d}^{\pi_{\text{mix},k-1}}) \rangle - \beta \epsilon_d, \end{aligned} \tag{9}$$

where in the last step we used the density oracle approximation error. Recall that we defined $r_k = (1-\gamma)\nabla L_k(\hat{d}^{\pi_{\text{mix},k-1}})$. Since π_k returned by the approximate planning oracle is an ϵ_p-optimal policy in M^k, we have $(1-\gamma)^{-1}\langle d^{\pi_k}, r_k \rangle \geq (1-\gamma)^{-1}\langle d^\pi, r_k \rangle - \epsilon_p$ for any policy π, including $\pi^\star$. Therefore,

$$\begin{aligned} \langle d^{\pi_k}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle &\geq \langle d^{\pi^\star}, \nabla L_k(\hat{d}^{\pi_{\text{mix},k-1}}) \rangle - \epsilon_p - \beta \epsilon_d \\ &\geq \langle d^{\pi^\star}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle - \epsilon_p - 2\beta \epsilon_d, \end{aligned} \tag{10}$$

where we used the density oracle approximation error once more in the second step. Going back to inequality (8), we further bound $L_k(d^{\pi_{\text{mix},k}})$ by

$$\begin{aligned} L_k(d^{\pi_{\text{mix},k}}) &\geq L_k(d^{\pi_{\text{mix},k-1}}) + \eta \langle d^{\pi_k} - d^{\pi_{\text{mix},k-1}}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle - 4\eta^2 \beta \\ &\geq L_k(d^{\pi_{\text{mix},k-1}}) + \eta \langle d^{\pi^\star} - d^{\pi_{\text{mix},k-1}}, \nabla L_k(d^{\pi_{\text{mix},k-1}}) \rangle - \eta \epsilon_p - 2\eta\beta\epsilon_d - 4\eta^2\beta \\ &\geq (1-\eta) L_k(d^{\pi_{\text{mix},k-1}}) + \eta L_k(d^{\pi^\star}) - 4\eta^2\beta - \eta\epsilon_p - 2\eta\beta\epsilon_d, \end{aligned}$$

where the last inequality is by concavity of $L_k(d)$. Therefore,

$$L_k(d^{\pi^\star}) - L_k(d^{\pi_{\mathrm{mix},k}}) \le (1-\eta)[L_k(d^{\pi^\star}) - L_k(d^{\pi_{\mathrm{mix},k-1}})] + 2\eta\beta\epsilon_d + \eta\epsilon_p + 4\eta^2\beta.$$

By assumption (iv), we write

$$\begin{aligned}
L_{K+1}(d^{\pi^\star}) - L_{K+1}(d^{\pi_{\mathrm{mix},K}}) &\le L_K(d^{\pi^\star}) - L_K(d^{\pi_{\mathrm{mix},K}}) + 2\delta_K \\
&\le (1-\eta)[L_K(d^{\pi^\star}) - L_K(d^{\pi_{\mathrm{mix},K-1}})] + 2\delta_K + 2\eta\beta\epsilon_d + \eta\epsilon_p + 4\eta^2\beta \\
&\le Be^{-\eta K} + 2\beta\epsilon_d + \epsilon_p + 4\eta\beta + 2\sum_{i=0}^{K}(1-\eta)^i\delta_{K-i} \\
&\le Be^{-\eta K} + 2\beta\epsilon_d + \epsilon_p + 4\eta\beta + 2\tau\xi.
\end{aligned}$$

It is straightforward to check that setting $\eta \le 0.1\epsilon\beta^{-1}, \epsilon_p \le 0.1\epsilon, \epsilon_d \le 0.1\epsilon\beta^{-1}, \tau \le 0.1\epsilon$, and the number of iterations $K \ge \eta^{-1}\log(10B\epsilon^{-1})$ yields the claim of Theorem 1.

Remark 2. *Since the temperature parameter τ_k in Proposition 1 goes to zero as k increases, one can show that the expected value of policy returned by Algorithm 1 converges to the maximum performance $J(\pi^\star)$.*

Proof of Proposition 1. For claim (ii), observe that $\nabla^2 L_k(d)$ is a diagonal matrix whose (s,a) diagonal term is given by

$$(\nabla^2 L_k(d))_{s,a} = \frac{-\tau}{4k^c} \times \frac{1}{(d(s,a)+\lambda)^{3/2}(\rho_{\mathrm{cov}}(s,a)+\lambda)^{1/2}}.$$

The diagonal elements are bounded by $-1/(4\lambda^2) \le (\nabla^2 L_k(d))_{s,a} \le \frac{1}{4\lambda^2} =: \beta$. Furthermore, by Taylor's theorem, one has

$$\|\nabla L_k(d) - \nabla L_k(d')\|_\infty \le \max_{(s,a),\alpha\in[0,1]} (\nabla^2 L_k(\alpha d + (1-\alpha)d'))\|d-d'\|_\infty \le \beta\|d-d'\|_\infty.$$

Claim (i) is immediate from the above calculation as the Hessian $\nabla^2 L_k(d)$ is negative definite. Claim (iii) may be verified by explicit calculation:

$$\sum_{s,a} d(s,a)r(s,a) + \frac{\tau}{k^c}\sum_{s,a}\sqrt{\frac{d(s,a)+\lambda}{\rho_{\mathrm{cov}}(s,a)+\lambda}} \le SA\left(1+\sqrt{\frac{1+\lambda}{\lambda}}\right) =: B.$$

For claim (iv), we have

$$L_{k+1}(d) - L_k(d) \le \sum_{s,a}\frac{\tau}{(k+1)^c}\sqrt{\frac{d(s,a)+\lambda}{\rho_{\mathrm{cov}}(s,a)+\lambda}} \le \frac{SA\tau}{(k+1)^c}\sqrt{\frac{1+\lambda}{\lambda}} =: \delta_k$$

We have

$$\sum_{i=0}^{k}(1-\eta)^i\delta_{k-i} = \tau SA\sqrt{\frac{1+\lambda}{\lambda}}\sum_{i=0}^{k}\frac{(1-\eta)^i}{(k-i+1)^c}.$$

For example, for $c=2$, the above sum is bounded by $\sum_{n=1}^{\infty} 1/n^2 = \pi^2/6$. Thus, one can set $\xi := \frac{\pi^2 SA}{6}\sqrt{\frac{1+\lambda}{\lambda}}$. □

B Experimental details

Source code is included in the supplemental material.

B.1 Bidirectional lock

Environment. For the bidirectional lock environment, one of the locks (randomly chosen) gives a larger reward of 1 and the other lock gives a reward of 0.1. Further details on this environment can be found in the work Agarwal et al. [2].

Exploration bonuses. We consider three exploration bonuses:

- Hoeffding-style bonus is equal to

$$\frac{V_{\max}}{\sqrt{N_k(s,a)}},$$

for every $s \in \mathcal{S}, a \in \mathcal{A}$, where $V_{\max}$ is the maximum possible value in an environment which we set to 1 for bidirectional lock.

- We use a Bernstein-style bonus

$$\sqrt{\frac{\mathrm{Var}_{s' \sim P_k(\cdot|s,a)} V_k(s')}{N_k(s,a)}} + \frac{1}{N_k(s,a)}$$

based on the bonus proposed by He et al. [34]. P_k denotes an empirical estimation of transitions $P_k(s'|s,a) = N_k(s,a,s')/N_k(s,a)$, where $N_k(s,a,s')$ is the number of samples on transiting to s' starting from state s and taking action a.

- MADE's bonus is set to the following in tabular setting:

$$\frac{1}{\sqrt{N_k(s,a)B_k(s,a)}}.$$

Algorithms. Below, we describe details on each tabular algorithm.

- **Value iteration.** We implement discounted value iteration given in [34] with all three bonuses.
- **PPO.** We implement a tabular version of the algorithm in [15], which is based on PPO with bonus. Specifically, the algorithm has the following steps: (1) sampling a new trajectory by running the stochastic policy π_k, (2) updating the empirical transition estimate P_k and exploration bonus, (3) computing Q-function Q_k of π_k over an MDP M_k with empirical transitions P_k and total reward r_k which is a sum of extrinsic reward and exploration bonus, and (4) updating the policy according to $\pi_{k+1}(a|s) \propto \pi_k(a|s)\exp(\alpha_k Q_k(s,a))$, where $\alpha_k = \sqrt{2\log(A)/HK}$ based on Cai et al. [15, Theorem 13.1].
- **Q-learning.** We implement Q-learning with bonus based on the algorithms given by Jin et al. [40].

B.2 Chain MDP

For the chain MDP described in Section 4.2, we run policy gradient for a tabular softmax policy parameterization $\pi(s|a) = \theta_{s,a}$ with the following RL objectives. Since we use a simplex parameterization, we run *projected* gradient ascent.

- **Vanilla PG.** The vanilla version simply considers the standard RL objective $J(\pi_\theta)$. For the gradient $\nabla_\theta J(\pi_\theta)$, see e.g. Agarwal et al. [1, Equation (32)].
- **PG with relative policy entropy regularization.** We use the objective (with the additive constant dropped) given in Agarwal et al. [1, Equation (12)]:

$$L(\pi_\theta) \coloneqq J(\pi_\theta) + \tau_k \sum_{s,a} \log \pi_\theta(a|s).$$

Here, index k denotes the policy gradient step. This form of regularization is more aggressive than the policy entropy regularized objective discussed next. Partial derivatives of the above objective are simply

$$\frac{\partial L(\pi_\theta)}{\partial \theta_{s,a}} = \frac{\partial J(\pi_\theta)}{\partial \theta_{s,a}} + \tau_k \frac{1}{\theta_{s,a}},$$

where the first term is analogous to the vanilla policy gradient.

- **PG with policy entropy regularization.** Policy entropy regularized objective [93, 62, 65, 57] is

$$L(\pi_\theta) \coloneqq J(\pi_\theta) - \tau_k (1-\gamma)^{-1} \mathbb{E}_{(s,a)\sim d_\rho^{\pi_\theta}(\cdot,\cdot)}[\log \pi_\theta(a|s)].$$

The gradient of the regularizer of the above objective is given in Lemma 1.

- **PG with MADE's regularization.** For MADE, we use the following objective

$$L(\pi_\theta) \coloneqq J(\pi_\theta) - \tau_k \sum_{s,a} \sqrt{d^\pi(s,a)}.$$

  The gradient of MADE's regularizer is computed in Lemma 2.

For all regularized objectives, we set $\tau_k = 0.1/\sqrt{k}$.

B.3 MiniGrid

We follow RIDE [16] and use the same hyperparameters for all the baselines. For ICM, RND, IMPALA, RIDE, BeBold and MADE, we use the learning rate 10^{-4}, batch size 32, unroll length 100, RMSProp optimizer with $\epsilon = 0.01$ and momentum 0. For entropy cost hyperparameters, we use 0.0005 for all the baselines except AMIGo. We provide the entropy cost for AMIGo below. We also test different values $\{0.01, 0.02, 0.05, 0.1, 0.5\}$ for the temperature hyperparameter in MADE. The best hyperparameters we found for each method are as follows. For **Bebold**, **RND**, and **MADE** we use intrinsic reward scaling factor of 0.1 for all environments. For **ICM** we use intrinsic reward scaling factor of 0.1 for KeyCorridor environments and 0.5 for the others. Hyperparameters in **RIDE** are exactly the same as **ICM**. For **AMIGo**, we use an entropy cost of 0.0005 for the student agent, and an entropy cost of 0.01 for the teacher agent.

B.4 DeepMind Control Suite

Environment. We use the publicly available environment DeepMind Control Suite [88] without any modification (Figure 9). Following the task design of RE3 [79], we use `Cheetah_Run_Sparse` and `Walker_Run_Sparse`.

Figure 9: Visualization of various tasks in DeepMind Control Suite. DeepMind Control Suite includes image-based control tasks with physics simulation. We mainly experiment on locomotion tasks in this environment.

Model-free RL implementations. For the experiments, we use the baselines of RAD [50], and we conduct a hyperparameter search over certain factors:

- **RND.** We search for the temperature parameter τ_0 over $\{0.001, 0.01, 0.05, 0.1, 0.5, 10.0\}$ and choose the best for each task. Specifically we use $\tau_0 = 0.1$ for `Pendulum_Swingup` and `Cheetah_Run_Sparse`, $\tau_0 = 10$ for `Cartpole_Swingup_Sparse`, and $\tau_0 = 0.05$ for others.
- **ICM.** We search for the temperature parameter τ_0 over $\{0.001, 0.01, 0.05, 0.1, 0.5, 1.0\}$ and choose the best for each task. Specifically we select $\tau_0 = 1.0$ for `Cheetah_Run_Sparse` and $\tau_0 = 0.1$ for the others. For the total loss used in training the networks, to balance the coefficient between forward loss and inverse loss, we follow the convention and use $L_{\text{all}} = 0.2 \cdot L_{\text{forward}} + 0.8 \cdot L_{\text{inverse}}$, where L_{forward} is the loss of predicting the next state given current state-action pair and L_{inverse} is the loss for predicting the action given the current state and the next state.
- **RE3**. We use an initial scaling factor $\tau_0 = 0.05$ (the scaling factor of τ_k at step 0) and decay it afterwards in each step. Note that we use the number of clusters $M = 3$ with a decaying factor on the reward $\rho = \{0.00001, 0.000025\}$. Therefore, the final intrinsic reward scaling factor becomes: $\tau_k = \tau_0 e^{-\rho k}$.
- **MADE.** We search for the temperature parameter τ_0 over $\{0.001, 0.01, 0.05, 0.1, 0.5\}$ and choose the best for each task. Specifically we select $\tau_0 = 0.05$ for `Cartpole_Swingup_Sparse`, `Walker_Run_Sparse` and `Cheetah_Run_Sparse`, $\tau_0 = 0.5$ for `Hopper_Hop` and `Pendulum_Swingup`, and $\tau_0 = 0.001$ for `Quadruped_Run`.

We use the same network architecture for all the algorithms. Specifically, the encoder consists of 4 convolution layers with ReLU activations. There are kernels of size 3×3 with 32 channels for all layers, and stride 1 except for the first layer which has stride 2. The embedding is then followed by a LayerNorm.

Model-based Rl implementation Here we provide implementation details for the model-based RL experiments. We adopt Dreamer as a baseline and build all the algorithms on top of that.

- **RE3.** For RE3, we follow the hyperparameters given in the original paper. We use an initial scaling factor $\tau_0 = 0.1$ without decaying τ_k afterwards. The number of clusters is set to $M = 50$. We use a decaying factor on the reward $\rho = 0$.
- **MADE.** We search for the temperature parameter τ_0 over $\{0.0005, 0.01, 0.05, 0.1, 0.5\}$ and choose the best for each map. Specifically we use 0.5 for `Cartpole_Swingup_Sparse`, `Cheetah_Run_Sparse` and `Hopper_Hop`, 0.01 for `Walker_Run_Sparse` and `Pendulum_Swingup` and 0.0005 for `Quadruped_Run`.

C Gradient computations

In this section we compute the gradients for policy entropy and MADE regularizers used in the chain MDP experiment. Before presenting the lemmas, we define two other visitation densities. The state visitation density $d^\pi : \mathcal{S} \to [0, 1]$ is defined as

$$d^\pi(s) \coloneqq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \, \mathbb{P}_t(s_t = s; \pi),$$

where $\mathbb{P}_t(s_t = s; \pi)$ denotes the probability of visiting s at step t starting at $s_0 \sim \rho(\cdot)$ following policy π. The state-action visitation density starting at (s', a') is denoted by

$$d^\pi_{s',a'}(s, a) \coloneqq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \, \mathbb{P}_t(s_t = s, a_t = a; \pi, s_0 = s', a_0 = a').$$

The following lemma computes the gradient of policy entropy with respect to policy parameters.

Lemma 1. *For a policy π parameterized by θ, the gradient of the policy entropy*

$$R(\pi_\theta) \coloneqq -\mathbb{E}_{(s,a)\sim d_\rho^{\pi_\theta}(\cdot,\cdot)}[\log \pi_\theta(a|s)],$$

with respect to θ is given by

$$\nabla_\theta R(\pi_\theta) = \mathbb{E}_{(s,a)\sim d_\rho^{\pi_\theta}(\cdot,\cdot)} \left[\nabla_\theta \log \pi(a|s) \left(\frac{1}{1-\gamma} \langle d^\pi_{s,a}, -\log \pi \rangle \right) - \log \pi(a|s) \right].$$

Proof. By chain rule, we write

$$\nabla_\theta R(\pi_\theta) = -\sum_{s,a} \nabla_\theta d^\pi(s,a) \log \pi(a|s) + \sum_{s,a} d^\pi(s,a) \nabla_\theta \log \pi(a|s) = -\sum_{s,a} \nabla_\theta d^\pi(s,a) \log \pi(a|s).$$

The second equation uses the fact that $\mathbb{E}_{x\sim p(\cdot)}[\nabla_\theta \log p(x)] = 0$ for any density p and that $d^\pi(s,a) = d^\pi(s)\pi(a|s)$ as laid out below:

$$\sum_{s,a} d^\pi(s,a) \nabla_\theta \log \pi(a|s) = \sum_s d^\pi(s) \sum_a \pi(a|s) \nabla_\theta \log \pi(a|s) = 0.$$

By another application of chain rule, one can write

$$\nabla_\theta d^\pi(s,a) = \nabla_\theta [d^\pi(s)\pi(a|s)] = \nabla_\theta d^\pi(s)\pi(a|s) + d^\pi(s,a) \nabla_\theta \log \pi(a|s).$$

We further simplify $\nabla_\theta R(\pi_\theta)$ according to

$$\begin{aligned} \nabla_\theta R(\pi_\theta) &= -\sum_{s,a} \nabla_\theta d^\pi(s,a) \log \pi(a|s) \\ &= -\sum_{s,a} \nabla_\theta d^\pi(s)\pi(a|s) \log \pi(a|s) - \sum_{s,a} d^\pi(s,a) \nabla_\theta \log \pi(a|s) \log \pi(a|s). \end{aligned}$$

We substitute $\nabla_\theta d^\pi(s)$ based on Zhang et al. [101, Lemma D.1]:

$$\begin{aligned}\nabla_\theta R(\pi_\theta) = &-\frac{1}{1-\gamma}\sum_{s',a'} d^\pi(s',a')\nabla_\theta \log(a'|s')\sum_{s,a} d_{s',a'}(s,a)\log\pi(a|s)\\ &-\sum_{s,a} d^\pi(s,a)\nabla_\theta\log\pi(a|s)\log\pi(a|s)\\ =&\,\mathbb{E}_{(s,a)\sim d_\rho^{\pi_\theta}(\cdot,\cdot)}\left[\nabla_\theta\log\pi(a|s)\left(\frac{1}{1-\gamma}\langle d^\pi_{s,a}, -\log\pi\rangle\right)-\log\pi(a|s)\right],\end{aligned}$$

where $\langle d^\pi_{s,a}, -\log\pi\rangle$ denotes the inner product between vectors $d^\pi_{s,a}$ and $-\log\pi$. This completes the proof. □

The following lemma computes the gradient of MADE regularizer with respect to policy parameters.

Lemma 2. *For a policy π parameterized by θ, the gradient of the regularizer*

$$R(\pi_\theta) \coloneqq \sum_{s,a}\sqrt{d^\pi(s,a)},$$

with respect to θ is given by

$$\nabla_\theta R(\pi_\theta) = \frac{1}{2}\,\mathbb{E}_{(s,a)\sim d^\pi(\cdot,\cdot)}\left[\nabla_\theta\log\pi(a|s)\left(\frac{1}{1-\gamma}\langle d^\pi_{s,a}, \frac{1}{\sqrt{d^\pi}}\rangle + \frac{1}{\sqrt{d^\pi(s,a)}}\right)\right].$$

Proof. The proof is similar to that of Zhang et al. [101, Lemma D.3]. We write $\nabla_\theta d^\pi(s,a) = \nabla_\theta d^\pi(s)\pi(a|s) + d^\pi(s,a)\nabla_\theta\log\pi(a|s)$ and conclude based on Zhang et al. [101, Lemma D.1] that

$$\begin{aligned}\nabla_\theta R(\pi_\theta) =&\frac{1}{2}\sum_{s,a}\frac{\nabla_\theta d^\pi(s,a)}{\sqrt{d^\pi(s,a)}}\\ =&\frac{1}{2(1-\gamma)}\sum_{s',a'} d^\pi(s',a')\nabla_\theta\log\pi(a'|s')\sum_{s,a} d^\pi_{s',a'}(s,a)\frac{1}{\sqrt{d^\pi(s,a)}}\\ &+\frac{1}{2}\sum_{s,a} d^\pi(s,a)\nabla_\theta\log\pi(a|s)\frac{1}{\sqrt{d^\pi(s,a)}}\\ =&\frac{1}{2}\,\mathbb{E}_{(s,a)\sim d^\pi(\cdot,\cdot)}\left[\nabla_\theta\log\pi(a|s)\left(\frac{1}{1-\gamma}\langle d^\pi_{s,a}, \frac{1}{\sqrt{d^\pi}}\rangle + \frac{1}{\sqrt{d^\pi(s,a)}}\right)\right].\end{aligned}$$

□