Task-Agnostic Exploration via Policy Gradient of a Non-Parametric State Entropy Estimate

Anonymous Authors[1]

Abstract

In a reward-free environment, what is a suitable intrinsic objective for an agent to pursue so that it can learn an optimal task-agnostic exploration policy? In this paper, we argue that the entropy of the state distribution induced by limited-horizon trajectories is a sensible target. Especially, we present a novel and practical policy-search algorithm, Maximum Entropy POLicy optimization (MEPOL), to learn a policy that maximizes a non-parametric, k-nearest neighbors estimate of the state distribution entropy. In contrast to known methods, MEPOL is completely model-free as it requires neither to estimate the state distribution of any policy nor to model transition dynamics. Then, we empirically show that MEPOL allows learning a maximum-entropy exploration policy in high-dimensional, continuous-control domains, and how this policy facilitates learning a variety of meaningful reward-based tasks downstream.

1. Introduction

In recent years, Reinforcement Learning (RL) (Sutton & Barto, 2018) has achieved outstanding results in remarkable tasks, such as Atari games (Mnih et al., 2015), Go (Silver et al., 2016), Dota 2 (Berner et al., 2019), and dexterous manipulation (Andrychowicz et al., 2020). To accomplish these feats, the learning process usually requires a considerable amount of human supervision, especially a hand-crafted reward function (Hadfield-Menell et al., 2017), while the outcome rarely generalizes beyond a single task (Cobbe et al., 2019). This barely mirrors human-like learning, which is far less dependent on exogenous guidance and exceptionally general. Notably, an infant would go through an intrinsically-driven, nearly exhaustive, exploration of the environment in an early stage, without knowing much about the tasks she/he will face. Still, this same unsupervised process will be consequential to solve those complex, externally-driven tasks, when they will eventually arise. In this perspective, what is a suitable *task-agnostic* exploration objective to set for the agent in an *unsupervised* phase, so that the acquired knowledge would facilitate learning a variety of reward-based tasks afterwards?

Lately, several works have addressed this question in different directions. In (Bechtle et al., 2019; Zheng et al., 2019), authors investigate how to embed task-agnostic knowledge into a transferable meta-reward function. Other works (Jin et al., 2020; Tarbouriech et al., 2020) consider the active estimation of the environment dynamics as an unsupervised objective. Another promising approach, which is the one we focus in this paper, is to incorporate the unsupervised knowledge into a *task-agnostic exploration policy*, obtained by maximizing some entropic measure over the state space (Hazan et al., 2019; Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020; Lee et al., 2019). Notably, an exploration policy is both easier to transfer than a transition model, which would be hardly robust to changes in the environment, and more ready to use than a meta-reward function, which would still require optimizing a policy as an intermediate step. An ideal maximum-entropy policy, thus inducing a uniform distribution over states, is an extremely general starting point to solve any (unknown) subsequent goal-reaching task, as it minimizes the so-called worst-case regret (Gupta et al., 2018, Lemma 1). In addition, by providing an efficient estimation of any, possibly sparse, reward function, it significantly reduces the burden on reward design. In tabular settings, (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020) propose theoretically-grounded methods for learning an exploration policy that maximizes the entropy of the asymptotic state distribution, while (Mutti & Restelli, 2020) concurrently considers the minimization of the mixing time as a secondary objective. In (Hazan et al., 2019), authors present a principled algorithm (MaxEnt) to optimize the entropy of the discounted state distribution of a tabular policy, and a theoretically-relaxed implementation to deal with function approximation. Similarly, (Lee et al., 2019) designs a method (SMM) to maximize the entropy of the finite-horizon state distribution. Both SMM and MaxEnt learn a maximum-entropy mixture of policies following this

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

iterative procedure: first, they estimate the state distribution induced by the current mixture to define an intrinsic reward, then, they learn a policy that optimizes this reward to be added to the mixture. Unfortunately, the literature approaches to state entropy maximization either consider impractical infinite-horizon settings (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), or output a mixture of policies that would be inadequate for non-episodic tasks (Hazan et al., 2019; Lee et al., 2019). In addition, they would still require a full model of the transition dynamics (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), or a state distribution density estimation (Hazan et al., 2019; Lee et al., 2019), which hardly scale to complex domains.

In this paper, we present a novel policy-search algorithm (Deisenroth et al., 2013), to deal with task-agnostic exploration via state entropy maximization over a *finite-horizon*, which gracefully scales to continuous, high-dimensional domains. The algorithm, which we call *Maximum Entropy POLicy optimization* (MEPOL), allows learning a *single* maximum-entropy parameterized policy from mere interactions with the environment, combining non-parametric state entropy estimation and function approximation. It is completely *model-free* as it requires neither to model the environment transition dynamics nor to directly estimate the state distribution of any policy. Notably, the entropy of continuous distributions can be speculated by looking at how random samples drawn from them laid out over the support surface (Beirlant et al., 1997). Intuitively, samples from a high entropy distribution would evenly cover the surface, while samples drawn from low entropy distributions would concentrate over narrow regions. Backed by this intuition, MEPOL relies on a k-nearest neighbors entropy estimator (Singh et al., 2003) to asses the quality of a given policy from a batch of interactions. Hence, it searches for a policy that maximizes this entropy index within a parametric policy space. To do so, it combines ideas from two successful, state-of-the-art policy-search methods: TRPO (Schulman et al., 2015), as it performs iterative optimizations of the entropy index within trust regions around the current policies, and POIS (Metelli et al., 2018), as these optimizations are performed offline via importance sampling. This recipe allows MEPOL to learn a maximum-entropy task-agnostic exploration policy while showing stable behavior during optimization.

The paper is organized as follows. First, we report the basic background (Section 2) and some relevant theoretical properties (Section 3) that will be instrumental to subsequent sections. Then, we present the task-agnostic exploration objective (Section 4), and a learning algorithm, MEPOL, to optimize it (Section 5), which is empirically evaluated in Section 6. In Section 7, we discuss related work. The proofs of the theorems are reported in Appendix A.

2. Preliminaries

In this section, we report the essential background and notation we will use throughout the paper.

Markov Decision Processes A discrete-time Markov Decision Process (MDP) (Puterman, 2014) is defined by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, d_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and the action space respectively, $P(s'|s,a)$ is a Markovian transition model that defines the conditional probability of the next state s' given the current state s and action a, $R(s)$ is the expected immediate reward when arriving in state s, and d_0 is the initial state distribution. A trajectory $\tau \in \mathcal{T}$ is a sequence of state-action pairs $\tau = (s_0, a_0, s_1, a_1, \ldots)$. A policy $\pi(a|s)$ defines the probability of taking action a given the current state s. We denote by Π the set of all stationary Markovian policies. A policy π that interacts with an MDP, induces a t-step state distribution defined as (let $d_0^\pi = d_0$):

$$d_t^\pi(s) = Pr(s_t = s|\pi) = \int_{\mathcal{T}} Pr(\tau|\pi, s_t = s)\,\mathrm{d}\tau,$$
$$d_t^\pi(s) = \int_{\mathcal{S}} d_{t-1}^\pi(s') \int_{\mathcal{A}} \pi(a|s') P(s|s',a)\,\mathrm{d}a\,\mathrm{d}s', \quad (1)$$

for every $t > 0$. If the MDP is ergodic, it admits a unique steady-state distribution which is $\lim_{t\to\infty} d_t^\pi(s) = d^\pi(s)$. The mixing time t_{mix} describes how fast the state distribution d_t^π converges to its limit, given a mixing threshold ϵ:

$$t_{\text{mix}} = \big\{t \in \mathbb{N} : \sup_{s\in\mathcal{S}} \big|d_t^\pi(s) - d_{t-1}^\pi(s)\big| \le \epsilon\big\}.$$

Differential Entropy Let $f(x)$ be a probability density function of a random vector $\boldsymbol{X}$ taking values in $\mathbb{R}^p$, then its differential entropy (Shannon, 1948) is defined as:

$$H(f) = -\int f(x) \ln f(x)\,\mathrm{d}x.$$

When the distribution f is not available, this quantity can be estimated given a realization of $\boldsymbol{X} = \{x_i\}_{i=1}^N$ (Beirlant et al., 1997). In particular, to deal with high-dimensional data, we can turn to non-parametric, k-Nearest Neighbors (k-NN) entropy estimators of the form (Singh et al., 2003):

$$\widehat{H}_k(f) = -\frac{1}{N}\sum_{i=1}^N \ln \frac{k}{N V_i^k} + \ln k - \Psi(k), \quad (2)$$

where Ψ is the digamma function, $\ln k - \Psi(k)$ is a bias correction term, V_i^k is the volume of the hyper-sphere of radius $R_i = |x_i - x_i^{k\text{-NN}}|$, which is the Euclidean distance between x_i an its k-nearest neighbor $x_i^{k\text{-NN}}$, so that:

$$V_i^k = \frac{\big|x_i - x_i^{k\text{-NN}}\big|^p \cdot \boldsymbol{\pi}^{p/2}}{\Gamma(\frac{p}{2}+1)},$$

where Γ is the gamma function, and p the dimensions of $\boldsymbol{X}$. The estimator (2) is known to be asymptotically unbiased and consistent (Singh et al., 2003). When the target distribution f' differs from the sampling distribution f, we can provide an estimate of $H(f')$ by means of an Importance-Weighted (IW) k-NN estimator (Ajgl & Šimandl, 2011):

$$\widehat{H}_k(f'|f) = -\sum_{i=1}^{N} \frac{W_i}{k} \ln \frac{W_i}{V_i^k} + \ln k - \Psi(k), \qquad (3)$$

where $W_i = \sum_{j \in \mathcal{N}_i^k} w_j$, such that $\mathcal{N}_i^k$ is the set of indices of the k-NN of x_i, and w_j are the normalized importance weights of samples x_j, which are defined as:

$$w_j = \frac{f'(x_j)/f(x_j)}{\sum_{n=1}^{N} f'(x_n)/f(x_n)}.$$

As a by-product, we have access to a non-parametric IW k-NN estimate of the Kullback-Leibler (KL) divergence, given by (Ajgl & Šimandl, 2011):

$$\widehat{D}_{KL}(f||f') = \frac{1}{N} \sum_{i=1}^{N} \ln \frac{k/N}{\sum_{j \in \mathcal{N}_i^k} w_j}. \qquad (4)$$

Note that, when $f' = f$, $w_j = 1/N$, the estimator (3) is equivalent to (2), while $\widehat{D}_{KL}(f||f')$ is zero.

3. Analysis of the Importance-Weighted Entropy Estimator

In this section, we present a theoretical analysis over the quality of the estimation provided by (3). Especially, we provide a novel detailed proof of the bias, and a new characterization of its variance. In a similar flavor as in (Singh et al., 2003, Theorem 8) for the estimator (2), we can prove the following theorem.

Theorem 3.1. (Ajgl & Šimandl, 2011, Sec. 4.1) *Let f be a sampling distribution, f' a target distribution. The estimator $\widehat{H}_k(f'|f)$ is asymptotically unbiased for any choice of k.*

Therefore, given a sufficiently large batch of samples from an unknown distribution f, we can get an unbiased estimate of the entropy of any distribution f', irrespective of the form of f and f'. However, if the distance between the two grows too large, a high variance might negatively affect the estimation.

Theorem 3.2. *Let f be a sampling distribution, f' a target distribution. The asymptotic variance of the estimator $\widehat{H}_k(f'|f)$ is given by:*

$$\lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\left[\widehat{H}_k(f'|f)\right] = \frac{1}{N}\Bigg(\operatorname*{Var}_{x\sim f}\left[\overline{w}\ln\overline{w}\right] + \operatorname*{Var}_{x\sim f}\left[\overline{w}\ln R^p\right] + \left(\ln C\right)^2 \operatorname*{Var}_{x\sim f}\left[\overline{w}\right]\Bigg),$$

where $\overline{w} = \frac{f'(x)}{f(x)}$, and $C = \frac{N\pi^{p/2}}{k\Gamma(p/2+1)}$ is a constant.

4. A Task-Agnostic Exploration Objective

In this section, we define a learning objective for task-agnostic exploration, which is a fully unsupervised phase that potentially precedes a set of diverse goal-based RL phases. First, we make a common regularity assumption on the class of the considered MDPs, which allows us to exclude the presence of unsafe behaviors or dangerous states when interacting with the environment.

Assumption 4.1. *For any policy $\pi \in \Pi$, the corresponding Markov chain P^π is ergodic.*

Then, following a common thread in maximum-entropy exploration (Hazan et al., 2019; Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), and particularly (Lee et al., 2019), which focuses on a finite-horizon setting as we do, we define the *task-agnostic exploration* problem:

$$\operatorname*{maximize}_{\pi\in\Pi} \ \mathcal{F}_{\text{TAE}}(\pi) = H\left(\frac{1}{T}\sum_{t=1}^{T} d_t^\pi\right), \qquad (5)$$

where $\bar{d}_T = \frac{1}{T}\sum_{t=1}^{T} d_t^\pi$ is the finite-horizon *average state distribution*. An optimal policy w.r.t. the introduced objective favors a maximal coverage of the state space into the finite-horizon T, irrespective of the state-visitation order. Notably, the exploration horizon T has not to be intended as a given trajectory length, but rather as a parameter of the unsupervised exploration phase which allows to tradeoff exploration quality (i.e., state-space coverage) with exploration efficiency (i.e., mixing properties).

As the thoughtful reader might realize, optimizing Objective (5) is not an easy task. Known approaches would require either to estimate the transition model in order to obtain average state distributions (Tarbouriech & Lazaric, 2019; Mutti & Restelli, 2020), or to directly estimate these distributions through a density model (Hazan et al., 2019; Lee et al., 2019). In contrast to the literature, we turn to non-parametric entropy estimation without explicit state distributions modeling, deriving a more practical policy-search approach that we present in the following section.

5. The Algorithm

In this section, we present a model-free policy-search algorithm, called *Maximum Entropy POLicy optimization* (MEPOL), to deal with the task-agnostic exploration problem (5) in continuous, high-dimensional domains. MEPOL searches for a policy that maximizes the performance index $\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}))$ within a parametric space of stochastic differentiable policies $\Pi_\Theta = \{\pi_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^q\}$. The performance index is given by the non-parametric entropy estimator (2) where we replace f with the average state distribution $\bar{d}_T(\cdot|\pi_{\boldsymbol{\theta}}) = \bar{d}_T(\boldsymbol{\theta})$. The approach combines the ideas of two successful policy-search algorithms, TRPO (Schulman

Algorithm 1 MEPOL

Input: exploration horizon T, sample-size N, trust-region threshold δ, learning rate α, nearest neighbors k
initialize $\boldsymbol{\theta}$
for epoch = $1, 2, \ldots,$ until convergence **do**
  draw a batch of $\lceil \frac{N}{T} \rceil$ trajectories of length T with $\pi_{\boldsymbol{\theta}}$
  build a dataset of particles $\mathcal{D}_\tau = \{(\tau_i^t, s_i)\}_{i=1}^N$
  $\boldsymbol{\theta}' =$ *IS-Optimizer*$(\mathcal{D}_\tau, \boldsymbol{\theta})$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$
end for
Output: task-agnostic exploration policy $\pi_{\boldsymbol{\theta}}$

IS-Optimizer

Input: dataset of particles $\mathcal{D}_\tau$, sampling parameters $\boldsymbol{\theta}$
initialize $h = 0$ and $\boldsymbol{\theta}_h = \boldsymbol{\theta}$
while $D_{KL}(\bar{d}_T(\boldsymbol{\theta}_0)||\bar{d}_T(\boldsymbol{\theta}_h)) \leq \delta$ **do**
  compute a gradient step:
    $\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}_h} \widehat{H}_k\big(\bar{d}_T(\boldsymbol{\theta}_h)|\bar{d}_T(\boldsymbol{\theta}_0)\big)$
  $h \leftarrow h + 1$
end while
Output: parameters $\boldsymbol{\theta}_h$

et al., 2015) and POIS (Metelli et al., 2018), as it is reported in the following paragraphs. Algorithm 1 provides the pseudocode for MEPOL.

Trust-Region Entropy Maximization The algorithm is designed as a sequence of entropy index maximizations, called epochs, within a trust-region around the current policy $\pi_{\boldsymbol{\theta}}$ (Schulman et al., 2015). First, we select an exploration horizon T and an estimator parameter $k \in \mathbb{N}$. Then, at each epoch, a batch of trajectories of length T is sampled from the environment with $\pi_{\boldsymbol{\theta}}$, so as to take a total of N samples. By considering each state encountered in these trajectories as an unweighted particle, we have $\mathcal{D} = \{s_i\}_{i=1}^N$ where $s_i \sim \bar{d}_T(\boldsymbol{\theta})$. Then, given a trust-region threshold δ, we aim to solve the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}' \in \Theta}{\text{maximize}} \quad & \widehat{H}_k\big(\bar{d}_T(\boldsymbol{\theta}')\big) \\ \text{subject to} \quad & D_{KL}\big(\bar{d}_T(\boldsymbol{\theta})\big|\big|\bar{d}_T(\boldsymbol{\theta}')\big) \leq \delta. \end{aligned} \tag{6}$$

The idea is to optimize Problem (6) via Importance Sampling (IS) (Owen, 2013), in a fully off-policy manner partially inspired by (Metelli et al., 2018), exploiting the IW entropy estimator (3) to calculate the objective and the KL estimator (4) to compute the trust-region constraint. We detail the off-policy optimization in the following paragraph.

Importance Sampling Optimization We first expand the set of particles $\mathcal{D}$ by introducing $\mathcal{D}_\tau = \{(\tau_i^t, s_i)\}_{i=1}^N$, where $\tau_i^t = (s_i^0, \ldots, s_i^t = s_i)$ is the portion of the trajectory that leads to state s_i. In this way, for any policy $\pi_{\boldsymbol{\theta}'}$, we can associate to each particle its normalized importance weight:

$$\overline{w}_i = \frac{Pr(\tau_i^t|\pi_{\boldsymbol{\theta}'})}{Pr(\tau_i^t|\pi_{\boldsymbol{\theta}})} = \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_i^z|s_i^z)}{\pi_{\boldsymbol{\theta}}(a_i^z|s_i^z)}, \quad w_i = \frac{\overline{w}_i}{\sum_{n=0}^N \overline{w}_n}.$$

Then, having set a constant learning rate α and the initial parameters $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$, we consider a gradient ascent optimization of the IW entropy estimator (3),

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha \nabla_{\boldsymbol{\theta}_h} \widehat{H}_k\big(\bar{d}_T(\boldsymbol{\theta}_h)|\bar{d}_T(\boldsymbol{\theta}_0)\big), \tag{7}$$

until the trust-region boundary is reached, i.e., when it holds:

$$\widehat{D}_{KL}\big(\bar{d}_T(\boldsymbol{\theta}_0)\big|\big|\bar{d}_T(\boldsymbol{\theta}_{h+1})\big) > \delta.$$

The following theorem provides the expression for the gradient of the IW entropy estimator in Equation (7).

Theorem 5.1. *Let $\pi_{\boldsymbol{\theta}}$ be the current policy and $\pi_{\boldsymbol{\theta}'}$ a target policy. The gradient of the IW estimator $\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta}))$ w.r.t. $\boldsymbol{\theta}'$ is given by:*

$$\nabla_{\boldsymbol{\theta}'} \widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta})) = -\sum_{i=0}^N \frac{\nabla_{\boldsymbol{\theta}'} W_i}{k} \left(V_i^k + \ln \frac{W_i}{V_i^k} \right),$$

where:

$$\nabla_{\boldsymbol{\theta}'} W_i = \sum_{j \in \mathcal{N}_i^k} w_j \times \Bigg(\sum_{z=0}^t \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^N \prod_{z=0}^t \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)} \sum_{z=0}^t \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\sum_{n=1}^N \prod_{z=0}^t \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}} \Bigg).$$

6. Empirical Analysis

In this section, we present a comprehensive empirical analysis, which is organized as follows:

6.1) We illustrate that MEPOL allows learning a maximum-entropy policy in a variety of continuous domains, outperforming the current state of the art (MaxEnt);
6.2) We illustrate how the exploration horizon T, over which the policy is optimized, maximally impacts the trade-off between state entropy and mixing time;
6.3) We reveal the significant benefit of initializing an RL algorithm (TRPO) with a MEPOL policy to solve numerous challenging continuous control tasks.

A thorough description of the experimental set-up, along with some additional results and visualizations, are provided in Appendix B.

6.1. Task-Agnostic Exploration Learning

In this section, we consider the ability of MEPOL to learn a task-agnostic exploration policy according to the proposed objective (5). Such a policy is evaluated in terms of its induced entropy value $\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}))$, which we henceforth refer as *entropy index*. We chose k to optimize the performance of the estimator, albeit experiencing little to none sensitivity

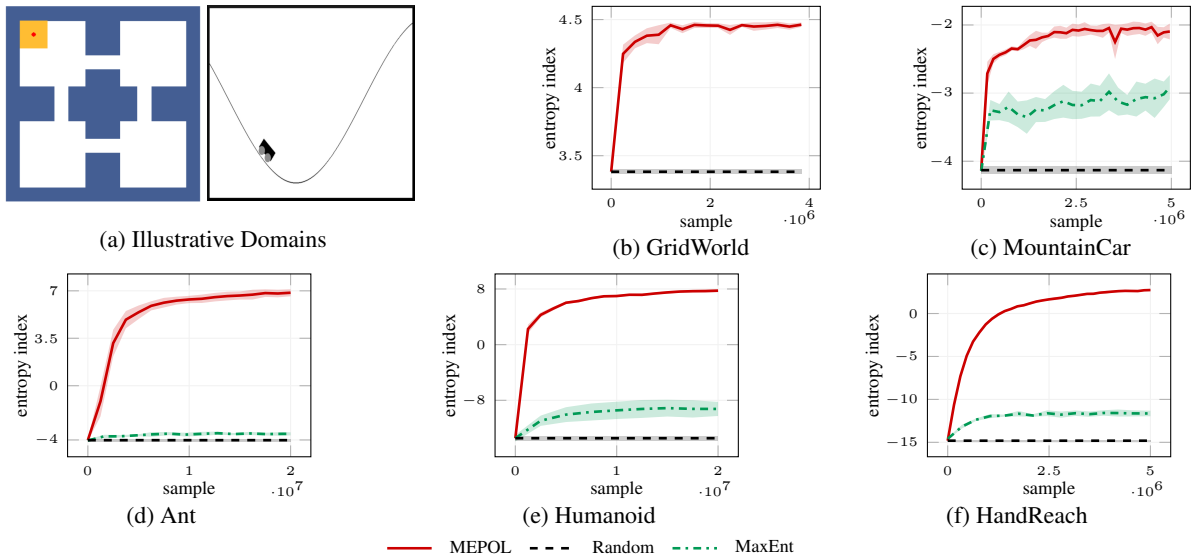


Figure 1. Comparison of the entropy index as a function of training samples achieved by MEPOL, MaxEnt, and a random policy. (95% c.i. over 8 runs. MEPOL parameters: k: 4 (c, d, e, f), 50 (b); T: 400 (c), 500 (d, e, f), 1200 (b). MaxEnt epochs: 20 (c), 30 (d, e, f)).

to this parameter (Appendix B.3.4). In any considered domain, we picked a specific T according to the time horizon we aimed to test in the subsequent goal-based setting (Section 6.3). This choice is not relevant in the policy optimization process, while we discuss how it affects the properties of the optimal policy in the next section. Note that, in all the experiments, we adopt a neural network to represent the parametric policy $\pi_{\boldsymbol{\theta}}$ (see Appendix B.2). We compare our algorithm with MaxEnt (Hazan et al., 2019). To this end, we considered the practical implementation of their algorithm to deal with continuous, non-discretized domains,[1] in which we employ TRPO instead of SAC (Haarnoja et al., 2018) for the RL component (see Appendix B.3.3). Note that MaxEnt learns a mixture of policies rather than a single policy. To measure its entropy index, we stick with the original implementation by generating a batch as follows: for each step of a trajectory, we sample a policy from the mixture and we take an action with it. We also investigated SMM (Lee et al., 2019) as a potential comparison. We do not report its results here for two reasons: we cannot achieve significant performance w.r.t. the random baseline, the difference with MaxEnt is merely in the implementation.

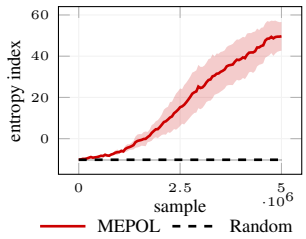
First, we evaluate task-agnostic exploration learning over two continuous illustrative domains: GridWorld (2D states, 2D actions) and MountainCar (2D, 1D), which we depict in Figure 1a. In these two domains, MEPOL successfully learns a policy that evenly covers the state space in a single batch of trajectories (state-visitation heatmaps are reported in Appendix B.3.5), while showcasing minimal variance across different runs (Figure 1b, 1c). Notably, it significantly outperforms MaxEnt in the MountainCar domain.[2] Then, we consider a set of continuous control, high-dimensional environments from the Mujoco suite (Todorov et al., 2012): Ant (29D, 8D), Humanoid (47D, 20D), HandReach (63D, 20D). While we learn a policy that maps full state representations to actions, we maximize the entropy index over a subset of the state sapce dimensions: 7D for Ant (3D position and 4D torso orientation), 24D for Humanoid (3D position, 4D body orientation, and all the joint angles), 24D for HandReach (full set of joint angles). Note that this selection of features is not dictated by the inability of MEPOL to cope with even more dimensions (see Figure 2b), but to obtain reliable and visually interpretable behaviors (see Appendix B.3.1). As we report in Figure 1d, 1e, 1f, MEPOL is able to learn policies with striking entropy values in all the environments. As a by-product, it unlocks several meaningful high-level skills during the process, such as jumping, rotating, navigation (Ant), crawling, standing up (Humanoid), and basic coordination (Humanoid, HandReach). Most importantly, the learning process is not negatively affected by the increasing number of dimensions, which is, instead, a well-known weakness of approaches based on explicit density estimation to compute the entropy (Beirlant et al., 1997). This issue is documented by the poor results of MaxEnt, which struggles to match the performance of MEPOL in the considered domains, as it prematurely converges to a low-entropy mixture.

[1] https://github.com/abbyvansoest/maxent/tree/master/humanoid

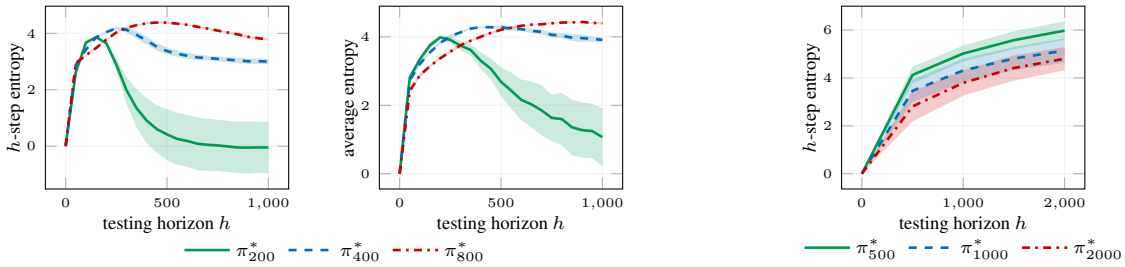
[2] We avoid the comparison in GridWorld, since the environment resulted particularly averse to MaxEnt.

	MountainCar	Ant	Humanoid
samples	$5 \cdot 10^6$	$2 \cdot 10^7$	$2 \cdot 10^7$
MEPOL	**4.31 ± 0.04**	**3.67 ± 0.05**	**1.92 ± 0.08**
MaxEnt	3.36 ± 0.4	1.92 ± 0.05	0.96 ± 0.06
Random	1.98 ± 0.05	1.86 ± 0.06	0.84 ± 0.04

(a) Comparison of the entropy over the 2D-discretized states achieved by MEPOL, MaxEnt, and a random policy (95% c.i. over 8 runs).



(b) 200D-GridWorld



(c) GridWorld (d) Humanoid

Figure 2. Comparison of the entropy index over an extended (200D, 200D) GridWorld domain (b). Comparison of the h-step entropy ($H(d_h^\pi)$) and average entropy ($H(\bar{d}_h)$) achieved by a set of policies trained over different horizons T as a function of the testing horizon h (c, d). (95% c.i. over 8 runs).

On MaxEnt Results One might realize that the performance reported for the MaxEnt algorithm appears to be much lower than the one presented in (Hazan et al., 2019). In this regard, some aspects need to be considered. First, their objective is different, as they focus on the entropy of discounted stationary distributions instead of $\bar{d}_T$. However, in the practical implementation, they consider undiscounted, finite-length trajectories as we do. Secondly, their results are computed over all samples collected during the learning process, while we measure the entropy over a single batch. Lastly, one could argue that an evaluation over the same measure (k-NN entropy estimate) that our method explicitly optimize is unfair. Nevertheless, even evaluating over the entropy of the 2D-discretized state space, which is the measure considered in (Hazan et al., 2019), leads to similar results (as reported in Figure 2a).

6.2. Impact of the Exploration Horizon Parameter

In this section, we discuss how choosing an exploration horizon T affects the properties of the learned policy. First, it is useful to distinguish between a *training horizon* T, which is an input parameter to MEPOL, and a *testing horizon* h on which the policy is evaluated. Especially, it is of particular interest to consider how an exploratory policy trained over T-steps fares in exploring the environment for a mismatching number of steps h. To this end, we carried out a set of experiments in the aforementioned GridWorld and Humanoid domains. We denote by π^*_T a policy obtained by executing MEPOL with a training horizon T and we consider the entropy of the h-step state distribution induced by π^*_T. Figure 2c (left), referring to the GridWorld experiment, shows that a policy trained over a shorter T might hit a peak in the entropy measure earlier (fast mixing), but other policies achieve higher entropy values at their optimum (highly exploring).[3] It is worth noting that the policy trained over 200-steps becomes overzealous when the testing horizon extends to higher values, while derailing towards a poor h-step entropy. In such a short horizon, the learned policy cannot evenly cover the four rooms and it overfits over easy-to-reach locations. Unsurprisingly, also the average state entropy over h-steps ($\bar{d}_h$), which is the actual objective we aim to maximize in task-agnostic exploration, is negatively affected, as we report in Figure 2c (right). This result points out the importance of properly choosing the training horizon in accordance with the downstream-task horizon the policy will eventually face. However, in other cases a policy learned over T-steps might gracefully generalize to longer horizons, as confirmed by the Humanoid experiment (Figure 2d). The environment is free of walls and hallways that can limit the agent's motion, so there is no specific incentive to overfit an exploration behavior over a shorter T.

6.3. Goal-Based Reinforcement Learning

In this section, we illustrate how a learning agent can benefit from an exploration policy learned by MEPOL when dealing with a variety of goal-based RL tasks. Especially, we compare the performance achieved by TRPO (Schulman et al.,

[3] The trade-off between entropy of the exploration and its mixing properties has been already substantiated for steady-state distributions in (Mutti & Restelli, 2020).

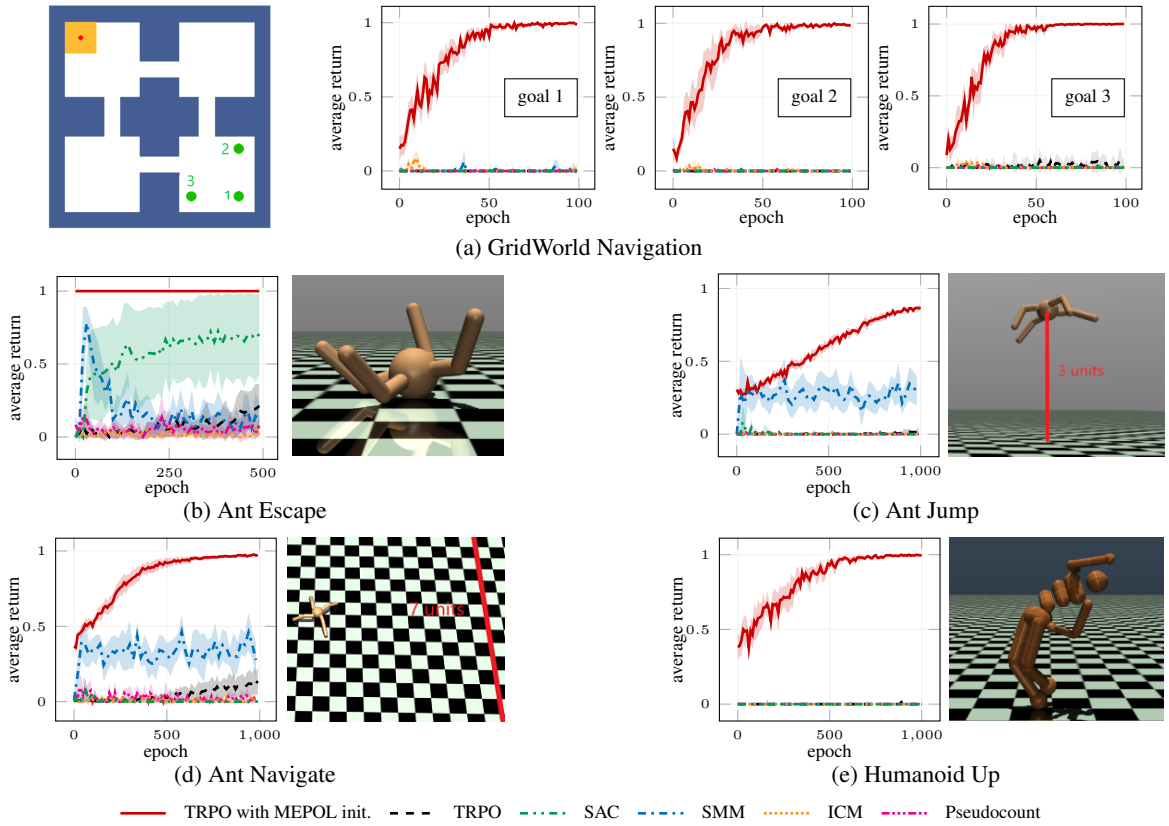


Figure 3. Comparison of the average return as a function of learning epochs achieved by TRPO with MEPOL initialization, TRPO, SAC, SMM, ICM, and Pseudocount over a set of sparse-reward RL tasks. For each task, we report a visual representation and learning curves. (95% c.i. over 8 runs).

2015) initialized with a MEPOL policy (the one we learned in Section 6.1) w.r.t. a set of significant baselines[4] that learn from scratch, i.e., starting from a randomly initialized policy. These baselines are: TRPO, SAC (Haarnoja et al., 2018), which promotes exploration over actions, SMM (Lee et al., 2019), which has an intrinsic reward related to the state-space entropy, ICM (Pathak et al., 2017), which favors exploration by fostering prediction errors, and Pseudocount (Bellemare et al., 2016), which assigns high rewards to rarely visited states. The algorithms are evaluated in terms of average return on a series of sparse-reward RL tasks defined over GridWorld, Ant, and Humanoid environments we considered in the previous sections.

In GridWorld, we test three navigation tasks with different goal locations (see Figure 3a). The reward is 1 in the states having Euclidean distance to the goal lower than 0.1. For the Ant environment, we define three, incrementally challenging, tasks: Escape, Jump, Navigate. In the first, the Ant starts from an upside-down position and it receives a reward of 1 whenever it rotates to a straight position (Figure 3b). In Jump, the agent gets a reward of 1 whenever it jumps higher than three units from the ground (Figure 3c). In Navigate, the reward is 1 in all the states further than 7 units from the initial location (Figure 3d). Finally, in Humanoid Up, the agent initially lies on the ground and it receives a reward of 1 when it is able to stand-up (Figure 3e). In all the considered tasks, the reward is zero anywhere except for the goal states, an episode ends when a goal is reached.

As we show in Figure 3, the MEPOL initialization leads to a striking performance across the board, while the tasks resulted extremely hard to learn from scratch. In some cases (Figure 3b), MEPOL allows for zero-shot policy optimization, as the optimal behavior has been already learned in the unsupervised exploration stage. In other tasks (e.g., Figure 3a), the MEPOL-initialized policy has lower return, but it permits for lighting fast adaptation w.r.t. random initialization. Note that, to match the tasks' higher-level of abstraction, in Ant Navigate and Humanoid Up we employed MEPOL initialization learned by maximizing the entropy over mere spatial coordinates (x-y in Ant, x-y-z in Humanoid). However, also the exact policies learned in Section 6.1 fares remarkably well in those scenarios (see Appendix B.4.3), albeit experiencing slower convergence.

[4]In view of previous results in task-agnostic exploration learning, we do not compare with TRPO initialized with a MaxEnt policy, which is clearly dominated by our approach.

7. Related Work

Our work falls into the category of unsupervised reinforcement learning. Especially, we address the problem of task-agnostic exploration, i.e., how to learn an unsupervised exploration policy that generalizes towards a wide range of tasks. This line of work relates to intrinsically-motivated learning, state-reaching approaches, and, to some extent, meta-RL. In this section, we provide a non-exhaustive review of previous work in these areas.

In the context of (supervised) reinforcement learning, many works have drawn inspiration from intrinsic motivation (Chentanez et al., 2005; Oudeyer et al., 2007) to design *exploration bonuses* that help the agent overcoming the exploration barrier in sparsely defined tasks. Some of them, initially (Schmidhuber, 1991) and more recently (Pathak et al., 2017; Burda et al., 2019a), promotes exploration by fostering prediction errors of an environment model, which is concurrently improved during the learning process. Another approach (e.g., Lopes et al., 2012; Houthooft et al., 2016) considers exploration strategies that maximizes the information gain, rather than prediction errors, on the agent's belief about the environment. Other works devise intrinsic rewards that are somewhat proportional to the novelty of a state, so that the agent is constantly pushed towards new portions of the state space. Notable instances are count-based methods (e.g., Bellemare et al., 2016; Tang et al., 2017), and random distillation (Burda et al., 2019b). Lately, Lee et al. (2019) propose an exploration bonus that is explicitly related to the state distribution entropy, while they present an interesting unifying perspective of other intrinsic bonuses in the view of entropy maximization.

In the intrinsic motivation literature, other works tackle the problem of learning a set of useful skills in an unsupervised setting (Gregor et al., 2017; Achiam et al., 2018; Eysenbach et al., 2019). The common thread is to combine variational inference and intrinsic motivation to maximize some information theoretic measure, which is usually closely related to *empowerment* (Salge et al., 2014; Mohamed & Rezende, 2015), i.e., the ability of the agent to control its own environment. In (Gregor et al., 2017), the proposed method learns a set of diverse skills by maximizing the mutual information between the skills and their terminations states. Eysenbach et al. (2019) consider a mutual information objective as well, but computed over all the states visited by the learned skills, instead of just termination states.

Another relevant line of research focus on *state-reaching objectives*. In tabular settings, Lim and Auer (Lim & Auer, 2012) cast the problem in its most comprehensive form, proposing a principled method to incrementally solve any possible state-reaching task, from the easiest to the hardest. In (Gajane et al., 2019), the same method is extended to non-stationary environments. Other works, originally (Bonarini et al., 2006) and lately (Ecoffet et al., 2019), provide algorithms to address a relaxed version of the state-reaching problem. In their case, the goal is to learn reaching policies to return to some sort of promising states (instead of all the states), then to seek for novel states from there on. In complex domains, Pong et al. (2019) consider learning maximum-entropy goal generation to provide targets to the state-reaching component. Notably, they relates this procedure to maximum-entropy exploration.

Meta-learning (Schmidhuber, 1987) has been successfully applied to address generalization in RL (e.g., Finn et al., 2017). In this set-up, the agent faces numerous tasks in order to meta-learn a general model, which, then, can be quickly adapted to solve unseen tasks in a few shots. These methods generally relies on reward functions to train the meta-model. Instead, Gupta et al. (2018) considers an unsupervised meta-RL set-up, which has some connections with our work. However, their focus is directed on learning fast adaptation rather than exploration, which is delegated to a method (Eysenbach et al., 2019) we already mentioned.

8. Conclusions

In this paper, we addressed task-agnostic exploration in environments with non-existent rewards by pursuing state entropy maximization. We presented a practical policy-search algorithm, MEPOL, to learn an optimal task-agnostic exploration policy in continuous, high-dimensional domains. We empirically showed that MEPOL achieves outstanding performance in terms of state entropy maximization, and that the learned policy paves the way for solving several reward-based tasks downstream. Natural extensions of MEPOL might target vision-based domains, which would require a meaningful distance metric between images, and explicitly accounting for the uncertainty of the entropy estimator in the off-policy component. Promising future directions includes addressing task-agnostic exploration in unsupervised meta-learning set-ups, and contemplating the use of a non-parametric state entropy regularization in reward-based policy optimization. We believe that this work can shed some light on the potential of state entropy maximization approaches to perform task-agnostic exploration.

References

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Ajgl, J. and Šimandl, M. Differential entropy estimation by particles. *IFAC*, 2011.

Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M.,

Powell, G., Ray, A., et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

Bechtle, S., Molchanov, A., Chebotar, Y., Grefenstette, E., Righetti, L., Sukhatme, G., and Meier, F. Meta-learning via learned loss. *arXiv preprint arXiv:1906.05374*, 2019.

Beirlant, J., Dudewicz, E. J., Györfi, L., and Van der Meulen, E. C. Nonparametric entropy estimation: An overview. *International Journal of Mathematical and Statistical Sciences*, 1997.

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

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Bonarini, A., Lazaric, A., Restelli, M., and Vitali, P. Self-development framework for reinforcement learning agents. In *Proceedings of the International Conference on Development and Learning*, pp. 355–362, 2006.

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

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

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

Chentanez, N., Barto, A. G., and Singh, S. P. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1281–1288, 2005.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1282–1289, 2019.

Deisenroth, M. P., Neumann, G., Peters, J., et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the International Conference on Machine Learning*, pp. 1329–1338, 2016.

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

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

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

Gajane, P., Ortner, R., Auer, P., and Szepesvari, C. Autonomous exploration for navigating in non-stationary cmps. *arXiv preprint arXiv:1910.08446*, 2019.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. In *Proceedings of the International Conference on Learning Representations*, 2017.

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

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

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S. J., and Dragan, A. Inverse reward design. In *Advances in Neural Information Processing Systems*, pp. 6765–6774, 2017.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *Proceedings of the International Conference on Machine Learning*, pp. 2681–2691, 2019.

Hodges, J. L. and Le Cam, L. The poisson approximation to the poisson binomial distribution. *The Annals of Mathematical Statistics*, 1960.

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

Jin, C., Krishnamurthy, A., Simchowitz, M., and Yu, T. Reward-free exploration for reinforcement learning. *arXiv preprint arXiv:2002.02794*, 2020.

Lee, L., Eysenbach, B., Parisotto, E., Xing, E., Levine, S., and Salakhutdinov, R. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

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

Lopes, M., Lang, T., Toussaint, M., and Oudeyer, P.-Y. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pp. 206–214, 2012.

Metelli, A. M., Papini, M., Faccio, F., and Restelli, M. Policy optimization via importance sampling. In *Advances in Neural Information Processing Systems*, pp. 5442–5454, 2018.

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

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

Mutti, M. and Restelli, M. An intrinsically-motivated approach for learning highly exploring and fast mixing policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2): 265–286, 2007.

Owen, A. B. Monte carlo theory, methods and examples. *Monte Carlo Theory, Methods and Examples. Art Owen*, 2013.

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

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

Pong, V. H., Dalal, M., Lin, S., Nair, A., Bahl, S., and Levine, S. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

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

Salge, C., Glackin, C., and Polani, D. Empowerment–an introduction. In *Guided Self-Organization: Inception*, pp. 67–114. 2014.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Schmidhuber, J. 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*, pp. 222–227, 1991.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *Proceedings of the International Conference on Machine Learning*, pp. 1889–1897, 2015.

Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal*, pp. 379–423, 1948.

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

Singh, H., Misra, N., Hnizdo, V., Fedorowicz, A., and Demchuk, E. Nearest neighbor estimates of entropy. *American Journal of Mathematical and Management Sciences*, 2003.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2753–2762, 2017.

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

Tarbouriech, J., Shekhar, S., Pirotta, M., Ghavamzadeh, M., and Lazaric, A. Active model estimation in markov decision processes. *arXiv preprint arXiv:2003.03297*, 2020.

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

Zheng, Z., Oh, J., Hessel, M., Xu, Z., Kroiss, M., van Hasselt, H., Silver, D., and Singh, S. What can learned intrinsic rewards capture? *arXiv preprint arXiv:1912.05500*, 2019.

A. Proofs

Theorem 3.1. (Ajgl & Šimandl, 2011, Sec. 4.1) *Let f be a sampling distribution, f' a target distribution. The estimator $\widehat{H}_k(f'|f)$ is asymptotically unbiased for any choice of k.*

Proof. The proof follow the sketch reported in (Ajgl & Šimandl, 2011, Section 4.1). First, We consider the estimator $\widehat{G}_k(f'|f) = \widehat{H}_k(f'|f) - \ln k + \Psi(k)$, that is:

$$\widehat{G}_k(f'|f) = \sum_{i=1}^{N} \frac{W_i}{k} \ln \frac{V_i^k}{W_i}. \tag{8}$$

By considering its expectation w.r.t. the sampling distribution we get:

$$\mathbb{E}_{x\sim f}[\widehat{G}_k(f'|f)] = \mathbb{E}_{x\sim f}\left[\sum_{i=1}^{N} \frac{W_i}{k} \ln \frac{1}{W_i} \frac{R_i^p p i^{p/2}}{\Gamma(\frac{p}{2}+1)}\right],$$

where, for the sake of clarity, we will replace each logarithmic term as:

$$T_i = \ln \frac{1}{\sum_{j\in\mathcal{N}_i^k} w_j} \frac{R_i^p p i^{p/2}}{\Gamma(\frac{p}{2}+1)}. \tag{9}$$

Since we are interested in the asymptotic mean, we can notice that for $N \to \infty$ we have $W_i/k \to \overline{w}_i/N$, where $\overline{w}_i = f'(x)/f(x)$ are the unnormalized importance weights (Ajgl & Šimandl, 2011, Section 4.1). Thus, for $N \to \infty$, we can see that:

$$\mathbb{E}_{x\sim f}\big[\widehat{G}_k(f'|f)\big] = \mathbb{E}_{x\sim f}\left[\sum_{i=1}^{N} \frac{\overline{w}_i}{N} T_i\right] = \mathbb{E}_{x\sim f'}\left[\frac{1}{N}\sum_{i=1}^{N} T_i\right],$$

where the random variables $T_1, T_2, \ldots, T_N$ are identically distributed, so that:

$$\mathbb{E}_{x\sim f}\big[\widehat{G}_k(f'|f)\big] = \mathbb{E}_{x\sim f'}\big[T_1\big].$$

Thus, to compute the expectation, we have to characterize the following probability for any real number r and any $x \sim f'$:

$$Pr\big[T_1 > r | X_1 = x\big] = Pr\big[R_1 > \rho_r | X_1 = x\big],$$

where we have:

$$\rho_r = \left[\frac{W_i \cdot \Gamma(\frac{p}{2}+1) \cdot e^r}{p i^{p/2}}\right]^{\frac{1}{p}}.$$

We can rewrite the probability as a binomial:

$$Pr\big[R_1 > \rho_r | X_1 = x\big] = \sum_{i=0}^{k-1} \binom{N-1}{i} \big[Pr(S_{\rho_r,x})\big]^i \big[1 - Pr(S_{\rho_r,x})\big]^{N-1-i},$$

where $P(S_{\rho_r,x})$ is the probability of x lying into the sphere of radius ρ_r, denoted as $S_{\rho_r,x}$. Then, we employ the Poisson Approximation (Hodges & Le Cam, 1960) to this binomial distribution, reducing it to a poisson distribution having parameter:

$$\begin{aligned}\lim_{N\to\infty}\big[N Pr(S_{\rho_r,x})\big] &= \lim_{N\to\infty}\left[N f(x) \frac{p i^{p/2} \cdot \rho_r^p}{\Gamma(\frac{p}{2}+1)}\right] \\ &= \lim_{N\to\infty}\big[N W_i f(x) e^r\big] = \overline{w}_i f(x) k e^r = f'(x) k e^r.\end{aligned}$$

Therefore, we get:

$$\lim_{N\to\infty} Pr\big[T_1 > r | X_1 = x\big] = \sum_{i=0}^{k-1} \frac{\big[k f'(x) e^r\big]^i}{i!} e^{-k f'(x) e^r} = Pr\big[T_x > r\big],$$

such that the random variable T_x has the pdf:

$$h_{T_x}(y) = \frac{\left[kf'(x)e^r\right]^k}{(k-1)!} e^{-kf'(x)e^r}, -\infty < y < \infty.$$

Finally, we can compute the expectation following the same steps reported in (Singh et al., 2003, Theorem 8):

$$\begin{aligned}
\lim_{N\to\infty} \mathbb{E}\big[T_1|X_1 = x\big] &= \int_{-\infty}^{\infty} y \frac{\left[kf'(x)e^r\right]^k}{(k-1)!} e^{-kf'(x)e^r}\,\mathrm{d}y \\
&= \int_0^{\infty} \big[\ln z - \ln k - \ln f'(x)\big] \frac{\left[z^{k-1}\right]}{(k-1)!} e^{-z}\,\mathrm{d}z \\
&= \frac{1}{\Gamma(k)} \int_0^{\infty} \big[\ln(z) z^{k-1} e^{-z}\big]\,\mathrm{d}z - \ln k - \ln f'(x) \\
&= \Psi(k) - \ln k - \ln f'(x),
\end{aligned}$$

which for a generic x it yields:

$$\lim_{N\to\infty} \mathbb{E}_{x\sim f'}\big[T_1\big] = H(f') - \ln k + \Psi(k) = \lim_{N\to\infty} \mathbb{E}_{x\sim f}\big[\widehat{G}_k(f'|f)\big].$$

□

Theorem 3.2. *Let f be a sampling distribution, f' a target distribution. The asymptotic variance of the estimator $\widehat{H}_k(f'|f)$ is given by:*

$$\begin{aligned}
\lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\big[\widehat{H}_k(f'|f)\big] = \frac{1}{N}\Bigg(&\operatorname*{Var}_{x\sim f}\big[\overline{w}\ln\overline{w}\big] \\
&+ \operatorname*{Var}_{x\sim f}\big[\overline{w}\ln R^p\big] + \big(\ln C\big)^2 \operatorname*{Var}_{x\sim f}\big[\overline{w}\big]\Bigg),
\end{aligned}$$

where $\overline{w} = \frac{f'(x)}{f(x)}$, and $C = \frac{N\pi^{p/2}}{k\Gamma(p/2+1)}$ is a constant.

Proof. We consider the limit of the variance of $\widehat{H}_k(f'|f)$, we have:

$$\lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\big[\widehat{H}_k(f'|f)\big] = \lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\big[\widehat{G}_k(f'|f)\big] = \lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\left[\sum_{i=1}^{N} \frac{W_i}{k} T_i\right] = \lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\left[\frac{1}{N}\sum_{i=1}^{N} \overline{w}_i T_i\right],$$

where $\widehat{G}_k(f'|f)$ is the estimator without the bias correcting term (8), and T_i are the logarithmic terms (9). Then, since the distribution of the random vector $\big(\overline{w}_1T_1, \overline{w}_2T_2, \ldots, \overline{w}_NT_N\big)$ is the same as any permutation of it (Singh et al., 2003):

$$\operatorname*{Var}_{x\sim f}\left[\frac{1}{N}\sum_{i=1}^{N} \overline{w}_i T_i\right] = \frac{\operatorname{Var}_{x\sim f}\big[\overline{w}_1T_1\big]}{N} + \frac{N(N-1)}{N^2} \operatorname{Cov}\big(\overline{w}_1T_1, \overline{w}_2T_2\big).$$

Assuming that, for $N \to \infty$, the term $\operatorname{Cov}(\overline{w}_1T_1, \overline{w}_2T_2) \to 0$ as its non-IW counterpart (Singh et al., 2003, Theorem 11), we are interested on the first term $\operatorname{Var}_{x\sim f}\big[\overline{w}_1T_1\big]$. Especially, we can derive:

$$\operatorname*{Var}_{x\sim f}\big[\overline{w}_1T_1\big] = \operatorname*{Var}_{x\sim f}\left[\overline{w}_1 \ln \frac{1}{\overline{w}_1} \frac{N}{k} \frac{R_1^p pi^{p/2}}{\Gamma(\frac{p}{2}+1)}\right] = \operatorname*{Var}_{x\sim f}\left[-\overline{w}_1 \ln \overline{w}_1 + \overline{w}_1 \ln R_1^p + \overline{w}_1 \ln \frac{N}{k} \frac{pi^{p/2}}{\Gamma(\frac{p}{2}+1)}\right],$$

where in the following, we will substitute $C = \frac{Npi^{p/2}}{k\Gamma(p/2+1)}$. Then, we can write the second momentum as:

$$\begin{aligned}
\mathbb{E}_{x\sim f}\Big[\big(\overline{w}_1T_1\big)^2\Big] = \mathbb{E}_{x\sim f}\Big[&\big(\overline{w}_1 \ln \overline{w}_1\big)^2 + \big(\overline{w}_1 \ln R_1^p\big)^2 + \big(\overline{w}_1 \ln C\big)^2 \\
&- 2\overline{w}_1^2 \ln \overline{w}_1 \ln R_1^p - 2\overline{w}_1^2 \ln \overline{w}_1 \ln C + 2\overline{w}_1^2 \ln R_1^p \ln C\Big],
\end{aligned}$$

while the squared expected value is:

$$\begin{aligned}\Big(\mathbb{E}_{x\sim f}\big[\overline{w}_1 T_1\big]\Big)^2 &= \Big(-\mathbb{E}_{x\sim f}\big[\overline{w}_1 \ln \overline{w}_1\big] + \mathbb{E}_{x\sim f}\big[\overline{w}_1 \ln R_1^p\big] + \mathbb{E}_{x\sim f}\big[\overline{w}_1 \ln C\big]\Big)^2 \\ &= \Big(\mathbb{E}_{x\sim f}\big[\overline{w}_1 \ln \overline{w}_1\big]\Big)^2 + \Big(\mathbb{E}_{x\sim f}\big[\overline{w}_1 R_1^p\big]\Big)^2 + \Big(\mathbb{E}_{x\sim f}\big[\overline{w}_1 \ln C\big]\Big)^2 \\ &\quad - 2\mathbb{E}_{x\sim f}\Big[\overline{w}_1^2 \ln \overline{w}_1 \ln R_1^p\Big] - 2\mathbb{E}_{x\sim f}\Big[\overline{w}_1^2 \ln \overline{w}_1 \ln C\Big] + 2\mathbb{E}_{x\sim f}\Big[\overline{w}_1^2 \ln C \ln R_1^p\Big].\end{aligned}$$

Thus, we have:

$$\begin{aligned}\operatorname*{Var}_{x\sim f}\big[\overline{w}_1 T_1\big] &= \mathbb{E}_{x\sim f}\Big[\big(\overline{w}_1 T_1\big)^2\Big] - \Big(\mathbb{E}_{x\sim f}\big[\overline{w}_1 T_1\big]\Big)^2 \\ &= \operatorname*{Var}_{x\sim f}\Big[\overline{w}_1 \ln \overline{w}_1\Big] + \operatorname*{Var}_{x\sim f}\Big[\overline{w}_1 \ln R_1^p\Big] + \operatorname*{Var}_{x\sim f}\Big[\overline{w}_1 \ln C\Big] \\ &= \operatorname*{Var}_{x\sim f}\Big[\overline{w}_1 \ln \overline{w}_1\Big] + \operatorname*{Var}_{x\sim f}\Big[\overline{w}_1 \ln R_1^p\Big] + \big(\ln C\big)^2 \operatorname*{Var}_{x\sim f}\Big[\overline{w}_1\Big].\end{aligned}$$

Summing it up, we can write the asymptotic order of the variance as:

$$\lim_{N\to\infty} \operatorname*{Var}_{x\sim f}\big[\widehat{H}(f'|f)\big] = \frac{\operatorname{Var}_{x\sim f}\big[\overline{w}(x)\ln\overline{w}(x)\big] + \operatorname{Var}_{x\sim f}\big[\overline{w}(x)\ln R(x)^p\big] + \big(\ln C\big)^2 \operatorname{Var}_{x\sim f}\big[\overline{w}(x)\big]}{N}$$

□

Theorem 5.1. *Let $\pi_{\boldsymbol{\theta}}$ be the current policy and $\pi_{\boldsymbol{\theta}'}$ a target policy. The gradient of the IW estimator $\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta}))$ w.r.t. $\boldsymbol{\theta}'$ is given by:*

$$\nabla_{\boldsymbol{\theta}'}\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta})) = -\sum_{i=0}^{N}\frac{\nabla_{\boldsymbol{\theta}'}W_i}{k}\Big(V_i^k + \ln\frac{W_i}{V_i^k}\Big),$$

where:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}'}W_i = \sum_{j\in\mathcal{N}_i^k} w_j \times \Bigg(&\sum_{z=0}^{t}\nabla_{\boldsymbol{\theta}'}\ln\pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) \\ &- \frac{\sum_{n=1}^{N}\prod_{z=0}^{t}\frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}\sum_{z=0}^{t}\nabla_{\boldsymbol{\theta}'}\ln\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\sum_{n=1}^{N}\prod_{z=0}^{t}\frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}}\Bigg).\end{aligned}$$

Proof. We consider the IW entropy estimator (3), that is

$$\widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta})) = -\sum_{i=1}^{N}\frac{W_i}{k}\ln\frac{W_i}{V_i^k} + \ln k - \Psi(k), \tag{10}$$

where:

$$W_i = \sum_{j\in\mathcal{N}_i^k} w_j = \sum_{j\in\mathcal{N}_i^k}\frac{\prod_{z=0}^{t}\frac{\pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z)}{\pi_{\boldsymbol{\theta}}(a_j^z|s_j^z)}}{\sum_{n=1}^{N}\prod_{z=0}^{t}\frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}}.$$

Then, by differentiating Equation (10) w.r.t. $\boldsymbol{\theta}'$, we have:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}'} \widehat{H}_k(\bar{d}_T(\boldsymbol{\theta}')|\bar{d}_T(\boldsymbol{\theta})) &= -\sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}'} \left(\frac{\sum_{j\in\mathcal{N}_i^k} w_j}{k} \ln \frac{\sum_{j\in\mathcal{N}_i^k} w_j}{V_i^k} + \ln k - \psi(k) \right) \\
&= -\sum_{i=1}^{N} \left(\frac{\sum_{j\in\mathcal{N}_i^k} \nabla_{\boldsymbol{\theta}'} w_j}{k} \ln \frac{\sum_{j\in\mathcal{N}_i^k} w_j}{V_i^k} + \frac{\sum_{j\in\mathcal{N}_i^k} w_j}{k} \frac{V_i^k}{\sum_{j\in\mathcal{N}_i^k} w_j} \sum_{j\in\mathcal{N}_i^k} \nabla_{\boldsymbol{\theta}'} w_j \right) \\
&= -\sum_{i=1}^{N} \frac{\sum_{j\in\mathcal{N}_i^k} \nabla_{\boldsymbol{\theta}'} w_j}{k} \left(V_i^k + \ln \frac{\sum_{j\in\mathcal{N}_i^k} w_j}{V_i^k} \right). \qquad (11)
\end{aligned}$$

Finally, we consider the expression of $\nabla_{\boldsymbol{\theta}'} w_j$ in Equation (11) to conclude the proof:

$$\begin{aligned}
\nabla_{\theta} w_j &= w_j \nabla_{\boldsymbol{\theta}'} \ln w_j \\
&= w_j \nabla_{\boldsymbol{\theta}'} \left(\ln \prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z)}{\pi_{\boldsymbol{\theta}}(a_j^z|s_j^z)} - \ln \sum_{n=1}^{N} \overbrace{\prod_{z=0}^{t} \frac{\pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)}{\pi_{\boldsymbol{\theta}}(a_n^z|s_n^z)}}^{\boldsymbol{Prod}_n} \right) \\
&= w_j \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N} \nabla_{\boldsymbol{\theta}'} \boldsymbol{Prod}_n}{\sum_{n=1}^{N} \boldsymbol{Prod}_n} \right) \\
&= w_j \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N} \boldsymbol{Prod}_n \nabla_{\boldsymbol{\theta}'} \ln \left(\boldsymbol{Prod}_n\right)}{\sum_{n=1}^{N} \boldsymbol{Prod}_n} \right) \\
&= w_j \left(\sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_j^z|s_j^z) - \frac{\sum_{n=1}^{N} \left(\boldsymbol{Prod}_n \sum_{z=0}^{t} \nabla_{\boldsymbol{\theta}'} \ln \pi_{\boldsymbol{\theta}'}(a_n^z|s_n^z)\right)}{\sum_{n=1}^{N} \boldsymbol{Prod}_n} \right).
\end{aligned}$$

□

B. Empirical Analysis: Further Details

B.1. Environments

For all the environments, we use off-the-shelf implementations from the OpenAI gym library (Brockman et al., 2016) with the exception of GridWorld, which we coded from scratch and we describe in the next paragraph. We also slightly modified the MountainCar environment by adding a wall on top of the right mountain to make the environment non-episodic.

In GridWorld, the agent can navigate a map composed of four rooms connected by four hallways, as represented in Figure 4. At each step the agent can choose how much to move on the x and y axes. The maximum continuous absolute change in position along any of the axes is 0.2. Each room is a space of 5 by 5 units, thus, the agent needs around 50 steps to move, along a straight line, from one side of the room to the other. Any action that leads the agent to collide with a wall is ignored and the agent remains in the previous state (position).

B.2. Class of Policies

In all the experiments, the policy is a Gaussian distribution with diagonal covariance matrix. It takes as input the environment state features and outputs an action vector $a \sim \mathcal{N}(\mu, \sigma^2)$. The mean μ is state-dependent and is the downstream output of a densely connected neural network. The standard deviation is state-independent and it is represented by a separated trainable vector. The dimension of μ, σ and a vectors is equal to the action-space dimension of the environment.

B.3. Task-Agnostic Exploration Learning

B.3.1. Continuous Control Set-Up

Here we further comment the experimental set-up for continuous control domains that we have briefly described in Section 6.1, especially concerning how we select the set of features on which the entropy index is maximized. First, in the

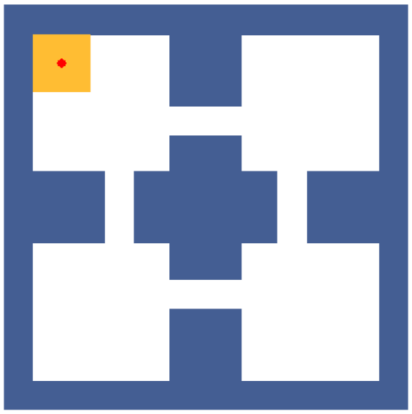


Figure 4. Visual representation of the GridWorld environment. The agent, represented with the red circle, starts at each episode in a random position inside the area highlighted in yellow.

Ant domain, we maximize the entropy over a 7D space of spatial coordinates (3D) and torso orientation (4D), excluding joint angles and velocities. This is to obtain an intermediate hurdle, in terms of dimensionality, w.r.t. the smaller GridWorld and MountainCar, and the most complex Humanoid and HandReach. Instead, in the latter two domains, we essentially maximize the entropy over the full state space excluding velocities and external forces, so that we have a 24D space both in Humanoid (3D position, 4D body orientation, and all the joint angles) and HandReach (all the joint angles). We noted that including external forces does not affect the entropy maximization in a meaningful way, as they resulted always zero during training. The reason why we also discarded velocities from the entropy computation is twofold. First, we noted that velocity-related features are quite unstable and spiky, so that they can be harmful without normalization, which we avoided. Secondly, we think that maximizing the entropy over the largest set of features is not necessarily a good idea when targeting generalization (see B.4.3), especially if improving the entropy over some features reduces the entropy over some other (as in the case of positions/angles and velocities).

B.3.2. MEPOL

For what concerns the k-nearest neighbors computation we use the *neighbors* package from the *scikit-learn* library (Pedregosa et al., 2011), which provides efficient algorithms and data structures to first compute and then query nearest neighbors. Before starting the main loop of the algorithm we perform some training steps to force the policy to output a zero μ vector. This is instrumental to obtain a common starting point across all the seeds and can be removed without affecting the algorithm behavior.

As outlined in the pseudcode of Algorithm 1, in each epoch a dataset of particles $\mathcal{D}_\tau = \{(\tau_i^t, s_i)\}_{i=1}^N$ is gathered for the given time horizon T. We call N_{traj} the number of trajectories, or batch size, used to build the dataset, so that $N = N_{traj} * T$.

Table 1. MEPOL Parameters

	MountainCar	GridWorld	Ant	Humanoid	HandReach
Number of epochs	650	200	2000	2000	2000
Horizon (T)	400	1200	500	500	50
Batch Size (N_{traj})	20	20	20	20	50
Kl threshold (δ)	15.0	15.0	15.0	15.0	15.0
Learning rate (α)	10^{-4}	10^{-5}	10^{-5}	10^{-5}	10^{-5}
Max iters	30	30	30	30	30
Number of neighbors (k)	4	50	4	4	4
Policy hidden layer sizes	(300,300)	(300,300)	(400,300)	(400,300)	(400,300)
Policy hidden layer act. function	ReLU	ReLU	ReLU	ReLU	ReLU
Number of seeds	8	8	8	8	8

B.3.3. MAXENT

As already outlined in Section 6.1, we use the original MaxEnt implementation to deal with continuous domains (https://github.com/abbyvansoest/maxent/tree/master/humanoid). We adopt this implementation also for the Ant and MountainCar environments, which were originally presented only as discretized domains (https://github.com/abbyvansoest/maxent_base and https://github.com/abbyvansoest/maxent/tree/master/ant). This allowed us not only to work in a truly continuous setting, as we do in MEPOL, but also to obtain better results than the discretized version. We were able to further enhance the MaxEnt performance by replacing SAC with TRPO.

In the tables below, you can find all the parameters and corresponding ranges (or sets) of values over which we searched for their optimal values. The TRPO parameters are reported employing the same notation as in (Duan et al., 2016). To estimate the density of the mixture, in Ant and Humanoid, we use a higher time horizon (T_d) than the optimized one (T). The reason why we do this is that, otherwise, we were not able to obtain reliable density estimations. The batch size used for the density estimation is denoted as N_{traj_d}. We also report the number of neighbors (k), which does not affect the learning process of MaxEnt, that we used to calculate the entropy of the mixture in the plots of Section 6.1. The entropy is computed over the same time horizon T and the same batch size N_{traj} used in MEPOL. Note that the horizon reported for TRPO is the same as the objective horizon T. The neural policy architectures are not reported as they are the same as in Section B.3.2.

Table 2. MaxEnt Parameters - MountainCar

	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	60	-
Density Horizon (T_d)	400	-
Density Batch Size (N_{traj_d})	100	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	0.1	[0.1, 2.0]
PCA	No	Yes, No
	TRPO	
Num. Iter.	50	40, 50, 60
Horizon	400	-
Sim. steps per Iter.	4000	-
δ_{KL}	0.1	0.1, 0.01, 0.001
Discount (γ)	0.99	-
Number of seeds	8	-

Table 3. MaxEnt Parameters - Ant

	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	30	-
Density Horizon (T_d)	10000	-
Density Batch Size (N_{traj_d})	10	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	0.2	[0.1, 2.0]
PCA	Yes (3 components)	Yes, No
	TRPO	
Num. Iter.	300	300, 500
Horizon	500	-
Sim. steps per Iter.	5000	-
δ_{KL}	0.1	0.1, 0.008
Discount (γ)	0.99	-
Number of seeds	8	-

Table 4. MaxEnt Parameters - Humanoid

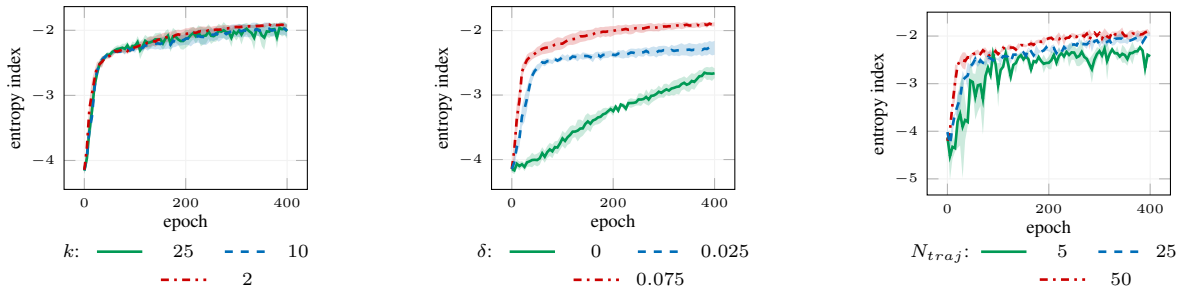
	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	30	-
Density Horizon (T_d)	50000	-
Density Batch Size (N_{traj_d})	20	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	1.0	[0.1, 2.0]
PCA	No	Yes, No
	TRPO	
Num. Iter.	300	200, 300
Horizon	500	-
Sim. steps per Iter.	5000	-
δ_{KL}	0.1	0.1, 0.008
Discount (γ)	0.99	-
Number of seeds	8	-

Table 5. MaxEnt Parameters - HandReach

	Value	Search In
Number of neighbors (k)	4	-
Mixture Size	30	-
Density Horizon (T_d)	10000	-
Density Batch Size (N_{traj_d})	20	-
KDE Kernel	Epanechnikov	Gaussian, Epanechnikov
KDE Bandwidth	1.0	[0.1, 5.0]
PCA	No	Yes, No
	TRPO	
Num. Iter.	300	200, 300
Horizon	50	-
Sim. steps per Iter.	500	-
δ_{KL}	0.1	-
Discount (γ)	0.99	-
Number of seeds	8	-

B.3.4. Parameters Sensitivity

In Figure 5, we show how the selection of the main parameters of MEPOL impacts on the learning process. To this end, we consider a set of experiments in the illustrative MountainCar domain, where we vary one parameter at a time to inspect the change in entropy index. As we can notice, the algorithm shows little sensitivity to the number of neighbors (k) considered in the entropy estimation. Allowing off-policy updates through an higher KL threshold δ positively impacts the learning efficiency. Furthermore, MEPOL displays a good behavior even when we limit the batch-size.



(a) k-NN (b) KL threshold (c) Batch-size

Figure 5. Comparison of the entropy index as a function of the learning epochs for MEPOL with different set of parameters on the MountainCar domain. (95% c.i. over 8 runs, $T = 400$ (a,b,c), $k = 4$ (b,c), $\delta = 0.05$ (a,c), $N_{traj} = 100$ (a,b)).

B.3.5. State-Visitation Heatmaps

In Figure 6, and Figure 7, we report the MEPOL state-coverage evolution over GridWorld and MountainCar domains. In Figure 8, you can see the state-coverage evolution in the Ant domain over a 12 by 12 units space, which is centered in the Ant starting position. The well-defined borders in the heatmap are due to trajectories going out of bounds. These heatmaps were created while running the experiments presented in Section 6.1 by discretizing the continuous state-space.

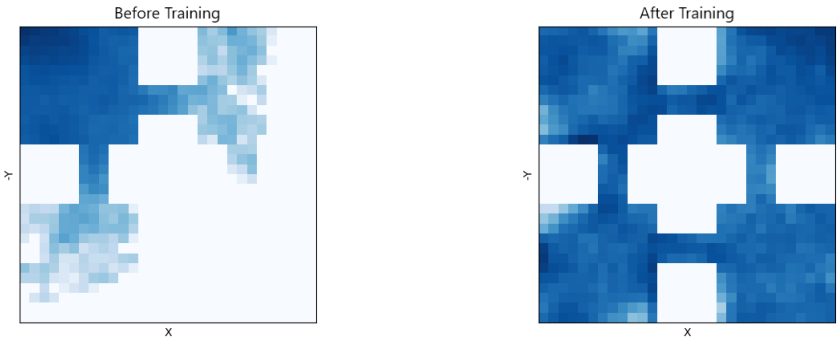


Figure 6. MEPOL log-probability state visitation evolution in the GridWorld domain created by running the policy for $N_{traj} = 100$ trajectories in a time horizon of $T = 1200$.

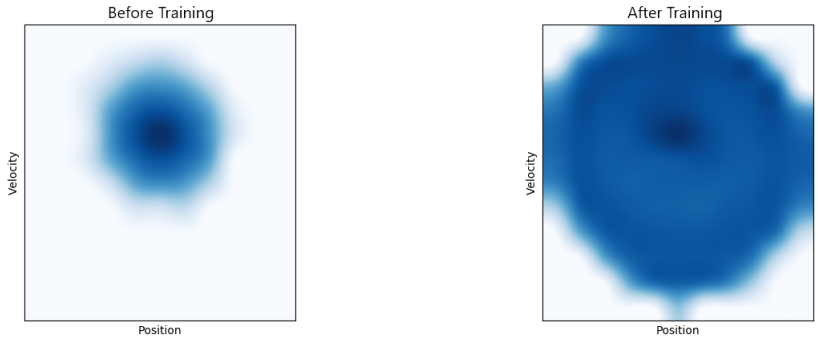


Figure 7. MEPOL log-probability state visitation evolution in the MountainCar domain created by running the policy for $N_{traj} = 100$ trajectories in a time horizon of $T = 400$.

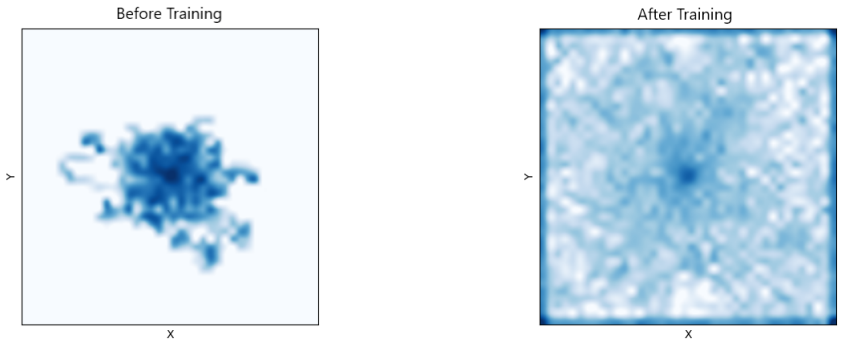
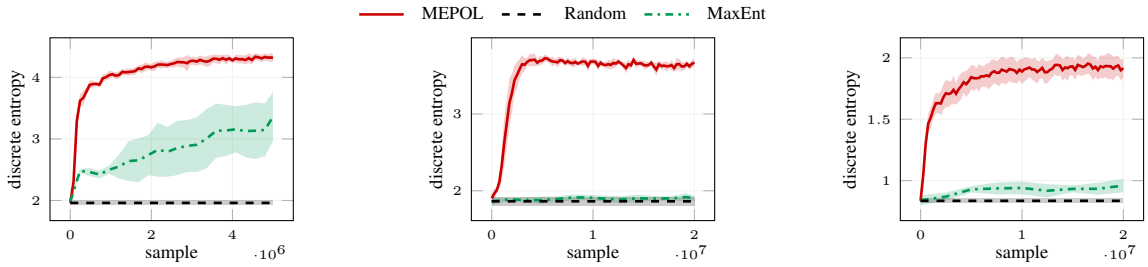


Figure 8. MEPOL log-probability (x, y) state visitation evolution in the Ant domain created by running the policy for $N_{traj} = 100$ trajectories in a time horizon of $T = 500$.

B.3.6. Discrete Entropy

In Figure 9, we report the plots for the evaluation of the entropy on the 2D-discretized state-space from which we have taken the values reported in Figure 2a. In Ant and Humanoid, the considered state-space is the 2D, discretized, agent's position (x, y). These plots were created while running the experiments in Section 6.1.



(a) MountainCar (b) Ant (c) Humanoid

Figure 9. Comparison of the entropy computed on the 2D-discretized state-space as a function of training samples achieved by MEPOL, MaxEnt, and a random policy in the MountainCar, Ant and Humanoid domains in the setting presented in Section 6.1 (95% c.i. over 8 runs).

B.4. Goal-Based Reinforcement Learning

B.4.1. Algorithms

We use the TRPO implementation from OpenAI's SpinningUp library https://github.com/openai/spinningup. For SAC, we adopt the codebase from https://github.com/microsoft/oac-explore. We use the original SMM codebase https://github.com/RLAgent/state-marginal-matching, which provides also an implementation of ICM, and Pseudocount.

B.4.2. Parameters Detail

In Table 6, we report the TRPO parameters used in the tasks, following the same notation in (Duan et al., 2016). Both the basic agent and the MEPOL agent use the same parameters. In Table 7, we report the SAC parameters, following the notation in (Haarnoja et al., 2018). In SMM we use 4 skills. In SMM, ICM, and Pseudocount we equally weight the extrinsic and the intrinsic components, as we haven't seen any improvement doing otherwise. Note that SMM, ICM, and Pseudocount are built on top of SAC for which we adopt the same parameters as in Table 7. The neural policy architectures are not reported as they are the same as in Section B.3.2.

Table 6. TRPO Parameters for Goal-Based Reinforcement Learning

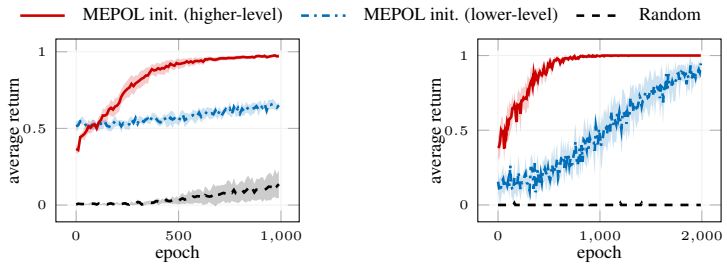
	GridWorld	AntEscape	AntJump	AntNavigate	HumanoidUp
Num. Iter.	100	500	1000	1000	2000
Horizon	1200	500	500	500	2000
Sim. steps per Iter.	12000	5000	50000	50000	20000
δ_{KL}	10^{-4}	10^{-2}	10^{-2}	10^{-2}	10^{-2}
Discount (γ)	0.99	0.99	0.99	0.99	0.99
Number of seeds	8	8	8	8	8

Table 7. SAC Parameters for Goal-Based Reinforcement Learning

	GridWorld	AntEscape	AntJump	AntNavigate	HumanoidUp
Epoch	100	500	1000	1000	2000
Num. Updates	12000	5000	5000	5000	6000
Learning Rate	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$
Discount (γ)	0.99	0.99	0.99	0.99	0.99
Replay buffer size (γ)	10^6	10^6	10^6	10^6	10^6
Number of samples per mini batch	256	256	256	256	256
Number of seeds	8	8	8	8	8

B.4.3. Higher-Level and Lower-Level Policies

In this section, we discuss the performance achieved by MEPOL policies when facing higher-level tasks, such as 2D navigation (Figure 10a) and standing-up (Figure 10b). Especially, we can see that higher-level MEPOL policies, which are trained to maximize the entropy over spatial coordinates (x-y in Ant, x-y-z in Humanoid), outperform lower-level MEPOL policies, which are trained to maximize entropy over spatial coordinates, orientation, and joint angles (as reported in Section 6.1). This is not surprising, since higher-level policies better match the level of abstraction of the considered tasks. However, it is worth noting that also lower-level policies achieve a remarkable initial performance, and a positive learning trend, albeit experiencing lower convergence.



(a) Ant Navigate (b) Humanoid Up

Figure 10. Comparison of the average return as a function of learning epochs achieved by TRPO with MEPOL initialization (higher-level, lower-level) and a Random initialization (95% c.i. over 8 runs).

B.5. Additional Experiments

In this section, we present two additional experiments. First, we present another experiment in the HandReach domain in which we maximize the entropy over the 3D position of each of the fingertip, for a total of 15 dimensions, whose performance is reported in Figure 11. Then, we present MEPOL in the AntMaze domain, in which the objective is to uniformly cover the 2D position of the agent, moving in a maze. The performance is reported, together with a visual representation of the environment, and a state-visitation evolution, in Figure 12.

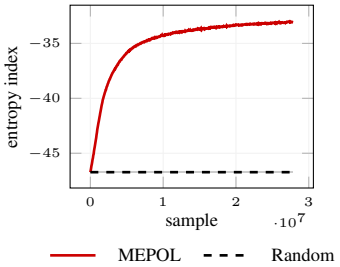


Figure 11. Performance of the entropy index as a function of training samples achieved by MEPOL and a random policy in the additional HandReach experiment (95% c.i. over 8 runs, $k = 4$, $T = 50$, $N_{traj} = 100$, $\delta = 0.05$).

C. Reproducing Results

A public repository will be released upon publication.

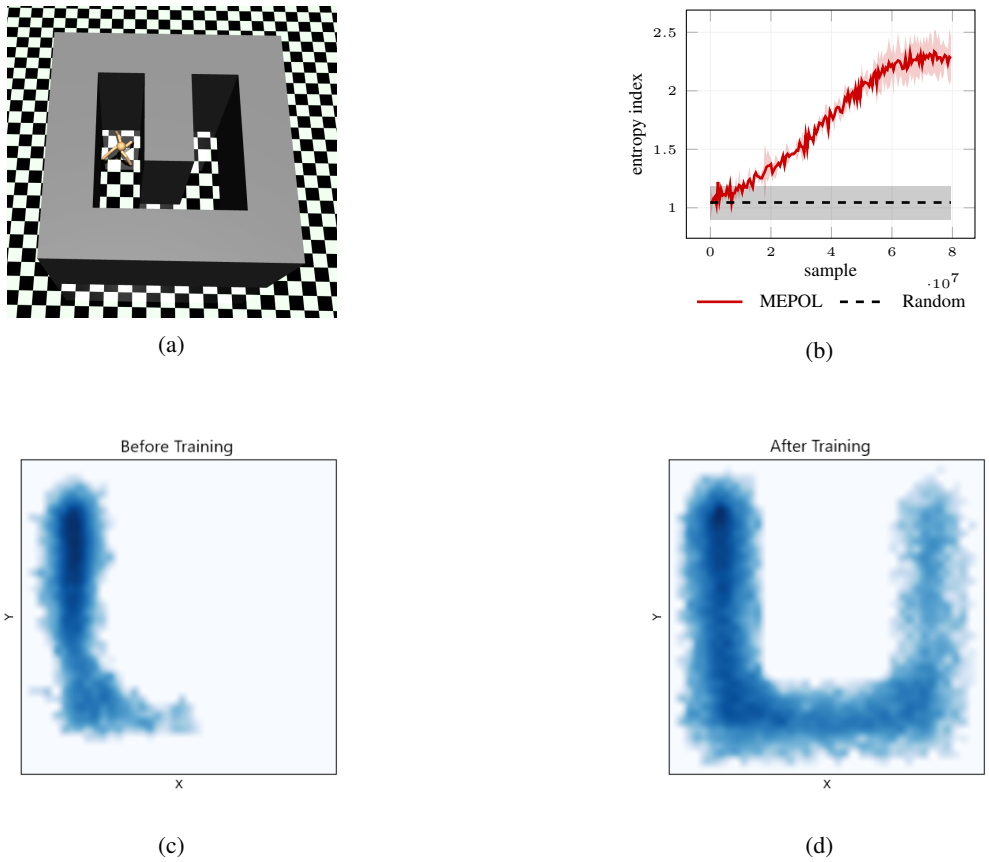


Figure 12. Performance of the entropy index (b) as a function of training samples achieved by MEPOL and a random policy in the AntMaze environment (a) (95% c.i. over 2 runs, $k = 50$, $T = 500$, $N_{traj} = 100$, $\delta = 0.05$) together with the log-probability state evolution before (c) and after (d) training created by running the policy for $N_{traj} = 100$ trajectories in the optimized time horizon.