Published in the proceedings of the Workshop on "Structure & Priors in Reinforcement Learning" at ICLR 2019

STATE MARGINAL MATCHING WITH MIXTURES OF POLICIES

Lisa Lee* Carnegie Mellon University lslee@cs.cmu.edu

Benjamin Eysenbach* Carnegie Mellon University beysenba@cs.cmu.edu

Emilio Parisotto Carnegie Mellon University eparisot@cs.cmu.edu

Ruslan Salakhutdinov Carnegie Mellon University

Sergey Levine UC Berkeley

ABSTRACT

Reinforcement learning (RL) algorithms today are hindered by their slothful rate of learning. When faced with a new task, RL algorithms fail to efficiently explore and learn which states have high reward. While most previous work on exploration in RL has focused on the single-task setting, we consider the multi-task setting, where many different reward functions can be provided for the same set of states and dynamics. To avoid re-inventing the wheel for each task, our method learns a single exploration policy that can quickly solve many downstream tasks. In this work, we define our exploration objective as the distance between the distribution over states visited by the policy and some prior distribution over states. In the absence of any prior information, this objective reduces to simply maximizing the marginal state entropy. We will illustrate how both standard RL and maximum action-entropy RL fail to solve *state distribution* matching problems. We propose a straightforward algorithm that iteratively fits a state density model and then updates the policy to visit states with low density under this model. Extending our approach to use *mixtures of policies*, we discover connections between distribution matching and the intrinsic motivation objectives based on mutual information in prior work. We demonstrate empirically that our method can effectively solve state distribution matching and reward-maximization tasks on complex navigation tasks, outperforming strong baselines.

1 INTRODUCTION

In principle, reinforcement learning (RL) can acquire policies that take actions to maximize long-term, expected utility on a wide range of tasks. In practice, RL algorithms today are hindered by their slothful rate of learning. When faced with a new task, RL algorithms fail to efficiently explore and learn which states have high reward.

While most previous work on exploration in RL has focused on the single-task setting, we consider the multi-task setting, where many different reward functions can be provided for the same set of states and dynamics. We consider two phases of training: (1) During the task-agnostic training phase, the agent simply learns to explore. (2) When given a task reward at test-time, the agent must adapt to maximize the task reward as quickly as possible. Since the exploration policy acquired during training is task-agnostic, it can be used to solve a wide range of downstream tasks. One way of viewing this few-shot learning setting is that, rather than re-inventing the wheel and learning to explore anew for each task, we amortize the problem of exploration by learning a single exploration policy that can be adapted to any reward function. This exploration policy can be viewed as a prior on the policy for solving downstream tasks.

While learning a single exploration policy is potentially quite useful, it is considerably more difficult than doing exploration throughout the course of learning a single task. For example, a reasonable plan of action for exploring a house is to spend one episode exploring the kitchen, the next episode exploring the living room, and the final episode exploring the bedroom. If, along the way, we find states with high reward, we can stop exploration and continue going to those high-reward states.

While this plan certainly results in good exploration, the policy for any particular episode is not a good exploration policy. For example, if we naively take the final policy, we will only ever explore the bedroom. The straightforward solution is to simply take the historical average over policies: At the start of each episode, sample one of the three policies and use the corresponding policy to sample actions in that episode. Our algorithm will implicitly do this.

We use the marginal entropy over states, $\mathcal{H}[s]$, as a metric for exploration, and seek to maximize this objective during training. Intuitively, this objective rewards the agent for visiting states it rarely visits. Policies that explore a wider range of states are preferred under this objective. This objective can also be viewed as minimizing the Kullback-Leibler Divergence between the policy's marginal state distribution and the uniform distribution, which is the least-informative prior. In the case where we have prior knowledge about the task, we can instead minimize the KL with some target distribution over states. We will show that this exploration objective is optimal for solving a general class of downstream tasks.[1]

While simple to state, our exploration objective cannot be maximized with standard RL because it is not a proper reward function. The reward function depends on the policy, because the reward given at a particular state is a function of how often the policy visits that state. We break this cyclic dependency by learning a separate density model over states. Our algorithm alternates between (A) updating the policy to visit states with low density under the density model, and (B) updating the density model to reflect the states visited by the policy. Crucially, we update the policy with respect to the historical average of density models and update the density model with respect to the historical average of policies.

In short, the contributions of this paper are (1) a principled, provably optimal exploration objective, (2) a convergent algorithm for maximizing this objective, and (3) analysis of our algorithm that relates exploration with competitive games. Empirical experiments show that our algorithm solves hard exploration tasks faster than state-of-the-art baselines in both navigation and manipulation domains. We hope that this work will spur further work on quantifying exploration and proving when certain exploration strategies are optimal.

2 Related Work

Most prior work has looked at exploration bonuses and intrinsic motivation. Typically, these algorithms (Pathak et al., 2017; Oudeyer et al., 2007; Schmidhuber, 1991; Houthooft et al., 2016) formulate some auxiliary task, and use prediction error on that task as an exploration bonus to encourage the agent to visit states where the predictive model is poor. Another class of methods (Tang et al., 2017; Bellemare et al., 2016; Schmidhuber, 2010) directly encourage the agent to visit novel states. While all methods effectively explore during the course of solving a single task, the policy they obtain at convergence is not a good exploration policy.

Some prior work has explicitly looked at the problem of *learning to explore* (Gupta et al., 2018; Xu et al., 2018). However, these methods rely on meta-learning algorithms which are often complicated and brittle. Our method is surprisingly simple, yet retains the benefit of fast test-time adaptation. Surprisingly, little work has actually looked at maximizing *state* entropy, though recent work by Hazan et al. (2018) presents a good survey of the field.

Closely related to our approach is standard maximum *action* entropy algorithms (Haarnoja et al., 2018; Kappen et al., 2012; Rawlik et al., 2013; Ziebart et al., 2008; Theodorou & Todorov, 2012). This class of algorithms are often motivated as matching a target distribution over *trajectories*, implicitly defined by the reward function (Levine, 2018). While distributions over trajectories define distributions over states, the relationship is complicated. Given a target distribution over states, it is quite challenging to design a reward function such that the optimal maximum action entropy policy matches the target state distribution. In Section C.1, we show that these maximum *action* entropy algorithms cannot solve state marginal matching problems.

[1] It is worth noting that matching state marginals is fully general: an arbitrary trajectory matching task can always be converted into state distribution matching tasks using a modified state space that appends all previous states to the current state.

3 STATE MARGINAL MATCHING

Formally, we define the state marginal matching problem as follows. We consider a parametric policy $\pi_\theta \in \Pi \triangleq \{\pi_\theta \mid \theta \in \Theta\}$ and a Markov Decision Process (MDP) $\mathcal{M}$ with fixed episode lengths T, dynamics function f, and initial state distribution p_0. An MDP $\mathcal{M}$ together with a policy π form an implicit density model over states. We overload notation to use $\pi(s)$ to denote this density:

$$\pi(s) \triangleq \mathbb{E}_{\substack{s_t \sim f(s_{t+1}|s_t,a_t) \\ s_0 \sim p_0(s), \pi(a_t|s_t)}} \left[\sum_{t=1}^{T} s_t = s \right]$$

To simplify notation, we will use $s \sim \pi(s)$ to denote states generated from this implicit generative process. Given a target distribution $p^*(s)$ over states $s \in \mathcal{S}$, our goal is to find a parametric policy that is "closest" to this target distribution, where distance is measured using using the Kullback-Leibler divergence between the state marginal distribution and the target state distribution:

$$\max_{\pi \in \Pi} -D_{\text{KL}}(\pi(s) \parallel p^*(s)) \tag{1}$$

Common Misinterpretations: Before proceeding, we debunk two potential misconceptions. First, state marginal matching is not a reinforcement learning problem. While our algorithm may solve RL problems in an inner loop, our objective is not a RL problem because the reward $\log \frac{p^*(s)}{\pi(s)}$ depends on the policy π. Second, we use $\pi(s)$ to denote the state distribution over finite-length episodes, not the stationary distribution converged to in the infinite limit.

3.1 SINGLE POLICY

To begin, we aim to learn a single policy $\pi(a \mid s)$ whose corresponding state distribution $\pi(s)$ is close to the target distribution $p^*(s)$ in terms of KL divergence:

$$\max_{\pi \in \Pi} -D_{\text{KL}}(\pi(s) \parallel p^*(s)) = \mathbb{E}_{s \sim \pi(s)} \left[\log \frac{p^*(s)}{\pi(s)} \right] = \mathbb{E}_{s \sim \pi(s)}[\log p^*(s)] + \mathcal{H}_\pi[s] \tag{2}$$

We immediately note that minimizing the KL divergence with the target distribution is equivalent to maximizing the reward function $r(s) \triangleq \log p^*(s)$ with a state-entropy-regularized policy.

3.2 MIXTURE OF POLICIES

Inspired by the success of mixture models in classical density modeling (Li & Barron, 2000; Bishop, 1994), we can likewise take a mixture over a collection of latent-conditioned policies, each with their own state distribution. We aim to learn a policy for each agent such that the marginal state distribution (the mixture of each agent's state distribution) closely matches the target distribution. Using $\pi(s \mid z)$ to denote the state distribution of the policy conditioned on the latent variable z, the marginal state distribution is

$$\pi(s) = \int_{\mathcal{Z}} \pi(s \mid z) \pi(z) dz = \mathbb{E}_{z \sim \pi(z)} \left[\pi(s \mid z) \right] \tag{3}$$

As before, we will minimize the KL divergence between this mixture distribution and the target distribution. Using Bayes rule to re-write $\pi(s)$ in terms of conditional probabilities, we obtain the following optimization problem:

$$\max_{\pi(\cdot|z)} \mathbb{E}_{\substack{z \sim \pi(z) \\ s \sim \pi(s|z)}} \left[\log \frac{p^*(s)}{\frac{\pi(s|z)\pi(z)}{\pi(z|s)}} \right] = \mathbb{E}_{\substack{z \sim \pi(z) \\ s \sim \pi(s|z)}} \left[\log p^*(s) + \log \pi(z \mid s) - \log \pi(s \mid z) - \log \pi(z) \right] \tag{4}$$

This decomposition separates into a pseudo-reward function for each agent z:

$$r_z(s) = \underbrace{\log p^*(s)}_{\text{(a)}} + \underbrace{\log \pi(z \mid s)}_{\text{(b)}} - \underbrace{\log \pi(s \mid z)}_{\text{(c)}} - \underbrace{\log \pi(z)}_{\text{(d)}} \tag{5}$$

Intuitively, this says that the agent should (a) go to states that have high density under the target state distribution, (b) go to states where this agent is clearly distinguishable from the other agents, (c) go to states where this agent has not been before. The last term (d) says that, when choosing how often to sample each agent, prefer the agents which are sampled less frequently.

Equation (4) bears a resemblance to the mutual-information objectives in recent work (Achiam et al., 2018; Eysenbach et al., 2018; Co-Reyes et al., 2018). Thus, one interpretation of our work is as explaining that mutual information objectives almost perform distribution matching. The caveat is that prior work omits the state entropy term, $-\log \pi(s \mid z)$. This term incentivizes exploration, possibly explaining why these previous works have failed to scale to complex tasks.

In Appendix C.1, we give a didactic example that illustrates why stochastic policies are necessary for state marginal distribution matching. Furthermore, we formally show in Appendix A that learning a policy and a density model jointly is an adversarial, zero-sum game; and that there always exists a solution to the problem if we use a mixture model (by the Nash existence theorem).

3.3 A Practical Algorithm

We now extend the state-entropy maximizing procedure to maximize the distribution matching objective. We learn a separate density model over states, $q(s) \approx \pi(s)$, to break the cyclic dependency between the reward function and the policy. For the single policy case, the only change is to maximize the policy w.r.t. the log difference of the target density and the learned density:

$$q_{t+1}(s) \leftarrow \arg\max_{q \in \mathcal{P}} \frac{1}{t} \sum_{i=1}^{t} \mathbb{E}_{s \sim \pi_t(s)} [\log q(s)]$$
$$\pi_{t+1} \leftarrow \arg\max_{\pi_t \in \Pi} \mathbb{E}_{s \sim \pi_t(s)} [\log p^*(s) - \log q_{t+1}(s)]$$

In the mixture-modeling setting, we additionally learn a discriminator $d(z \mid s)$ that predicts the mixture component from the state. Jensen's inequality tells us that maximizing the log-density of the learned discriminator will maximize a lower bound on the true density (see Agakov (2004)):

$$\mathbb{E}_{s \sim \pi(s|z), z \sim \pi(z)} [\log d(z \mid s)] \leq \mathbb{E}_{s \sim \pi(s|z), z \sim \pi(z)} [\log p(z \mid s)]$$

In our experiments, we leave the latent prior $\pi(z)$ as fixed and uniform. The resulting algorithm iterates between the following steps, while sampling $z \sim \pi(z)$ at the start of each epoch:

$$q^{(z)}_{t+1} \leftarrow \arg\max_{q \in \mathcal{P}} \frac{1}{t} \sum_{i=1}^{t} \mathbb{E}_{s \sim \pi_t(s|z)} [\log q(s)]$$
$$d_{t+1} \leftarrow \arg\max_{d} \mathbb{E}_{s \sim \pi_t(s|z), z \sim \pi(z)} [\log d(z \mid s)]$$
$$\pi_{t+1} \leftarrow \arg\max_{\pi_t \in \Pi} \mathbb{E}_{s \sim \pi_t(s|z), z \sim \pi(z)} \left[\log p^*(s) - \log q^{(z)}_{t+1}(s) + \log d_{t+1}(z \mid s) - \log \pi(z) \right]$$

4 Experiments

Question 1: *Does state marginal matching accelerate exploration in hard exploration tasks?*

To answer this question, we designed a suite of navigation tasks, one of which is shown in Figure 1. The agent starts at the center and must reach a goal at the end of one of the three hallways. We can vary the length of the hallway and the number of halls to finely control the task difficulty to effectively measure exploration. We consider both 2D navigation where the agent is a point mass moving in (x, y)-coordinates, and 3D navigation where the agent is a MuJoCo Ant robot (Todorov et al., 2012) learning to control its joints. See the appendix for full experimental details, additional experiments and ablation analysis.

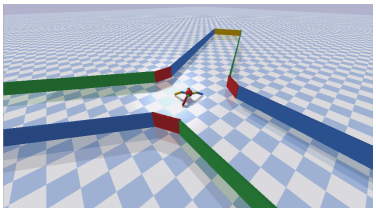
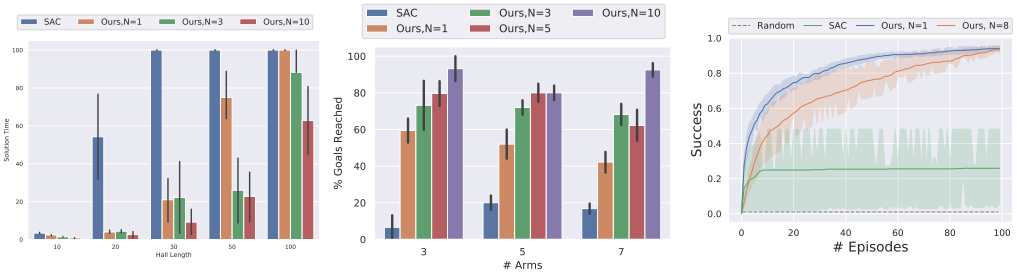


Figure 1: Ant navigation.

To begin, we define our target state distribution to place mass at the end of one hallway. Figure 2a shows how the time to solve the task varies as we increase the difficulty of the task by increasing



(a) Nav.: Epochs to solve task. (b) Nav.: Ratio of goals reached. (c) Manip.: Test-time adaptation.

Figure 2: On a 2D navigation task, we observe that our method (a) solves sparse reward navigation tasks significantly faster than SAC (1 epoch = 1k steps) and (b) can capture multi-model target distributions, especially when using mixtures of policies. (c) On the manipulation task, our method visits more goals. All error bars show 50% confidence interval across 5 random seeds.

the length of the halls. We observe that only our method can solve the task with long hallways, and using mixtures of policies aids exploration, enabling the agent to solve the task in less time.

Question 2: *Do mixtures of policies capture complex target distributions?*

We increased the number of hallways and defined the target state distribution to put probability mass at the end of each hallway. For a given agent (with either a single policy or a mixture of policies), we counted the number of times it reached the end of each hallway during training. Figure 2b records this fraction of hallways explored as we vary the number of hallways. While SAC rarely visits more than 20% of the goals, our method consistently visits 60% of goals. More importantly, we find that mixture policies visit more goals, suggesting that they better capture the multi-modal nature of the target density. Note that in all experiments, we run each method for the same number of environment transitions; a mixture of 3 policies *does not* get to take 3 times more transitions.

Question 3: *Does state marginal matching allow us to quickly learn tasks provided at test time?*

We applied our method to the Fetch manipulation environment (Plappert et al., 2018), shown to the right. We defined the target distribution to place uniform probability on states where the block was on the table and very low probability to states off the table. The target distribution also incorporated the prior that actions should be small and the arm should be close to the object (see Appendix D.2 for more details). We trained both our method and SAC on the same reward function (i.e., target distribution). At test time, we sampled goal states uniformly on the table and finetuned each agent for 100 steps. The goal state is not given to the agent, requiring the agent to explore the table.

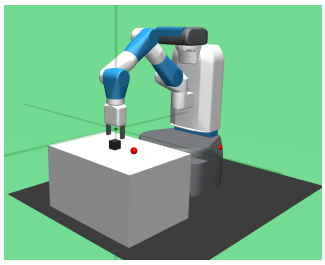
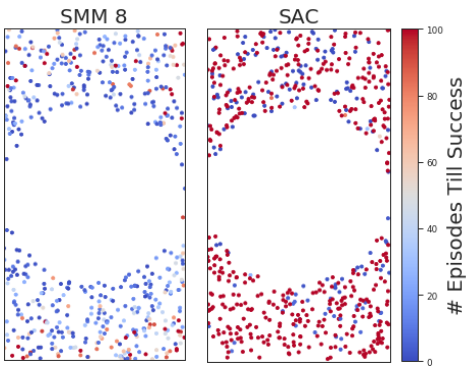


Figure 2c shows that our method effectively explores and quickly finds the goal state, while the baseline (SAC) struggles to find over 70% of goals. The figure on the right shows the xy-coordinates of the sampled goal locations (on the table surface) at test time, colored according to the number of episodes it took each algorithm to find the goal (blue = 0 episodes, red = 100 episodes). The block starts at the center of the table and goals (shown as red/blue dots) are some distance away. We observe that our method ("SMM 8") quickly finds most goals while SAC fails to reach a large portion of the goals.



5 Discussion

In this paper, we introduced a formal objective for exploration along with an algorithm for optimizing it. Experiments on navigation and manipulation tasks demonstrated how our method accelerates standard RL and allows for fast adaptation to new tasks provided at test time. Broadly, we believe that state marginal matching is a flexibly tool for incorporating priors into RL algorithms to accelerate learning.

REFERENCES

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

David Barber Felix Agakov. The im algorithm: a variational approach to information maximization. *Advances in Neural Information Processing Systems*, 16:201, 2004.

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*, pp. 1471–1479, 2016.

Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994.

G. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 1951.

John D Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *arXiv preprint arXiv:1806.02813*, 2018.

Constantinos Daskalakis and Qinxuan Pan. A counter-example to karlin's strong conjecture for fictitious play. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pp. 11–20. IEEE, 2014.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5302–5311, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Elad Hazan, Sham M Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. *arXiv preprint arXiv:1812.02690*, 2018.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Jonathan Q Li and Andrew R Barron. Mixture density estimation. In *Advances in neural information processing systems*, pp. 279–285, 2000.

John Nash. Non-cooperative games. *Annals of mathematics*, pp. 286–295, 1951.

Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

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*, pp. 16–17, 2017.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Julia Robinson. An iterative method of solving a game. *Annals of mathematics*, pp. 296–301, 1951.

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

Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

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*, pp. 2753–2762, 2017.

Evangelos A Theodorou and Emanuel Todorov. Relative entropy and free energy dualities: Connections to path integral and kl control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 1466–1473. IEEE, 2012.

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

Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore with meta-policy gradient. *arXiv preprint arXiv:1803.05044*, 2018.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A MAXIMIZING STATE ENTROPY

An integral part of optimizing the state marginal matching objective (Eq. 2) is maximizing the marginal state entropy of the policy, $\max_\pi \mathbb{E}_{s\sim\pi}[-\log \pi(s)]$. Optimizing this is difficult because the objective depends on π in two places. To break this cyclic dependency, we can decouple this dependency by introducing a proxy. We introduce a parametric state density model $q_\phi \in Q \triangleq \{q_\phi \mid \phi \in \Phi\}$ proxy distribution q to model the state density. If the class of density models is sufficiently expressive so the marginal state distribution of every policy $\pi \in \Pi$ can be represented by a density model $q \in Q$, then we can optimize the policy w.r.t. the proxy distribution:

Lemma A.1. *Let a set of policies Π and density models Q be given. If for any $\pi \in \Pi$ there exists $q \in Q$ such that $D_{KL}(\pi \parallel q) = 0$, then the following optimization problems are equivalent:*

$$\max_\pi \mathbb{E}_\pi[-\log \pi(s)] \quad \textit{and} \quad \max_\pi \min_q \mathbb{E}_\pi[-\log q(s)]$$

Now, we aim to solve the second optimization. We can formalize optimization problem as an equilibrium of a two-player, zero-sum game between a *policy player*, whose strategies are to selecting policy parameters, and a *density player*, whose strategies are to select density models of states. Note that we now have two sets of actions to distinguish between: (i) those due to traditional RL problem (*actions*) and (ii) those due to the decision problem (*strategies*).

$$\max_\phi \mathbb{E}_{s\sim\pi}[\log \pi(s)]$$

The Nash existence theorem proves that such a stationary point always exists:

Theorem A.2 (Nash (1951))**.** *Every two-player, zero-sum game with finite actions has a mixed strategy equilibrium point:*

While the number of strategies for both players is large, the finite precision of computers dictates that the number of pure strategies (all unique combinations of neural network parameters) be finite: $|\Theta|, |\Phi| < \infty$. While Theorem A.2 proves that a stationary point exists, it fails to provide a method for finding it. Fictitious play (Brown, 1951) provides an algorithm that does converge to a stationary point by having each player iteratively choosing the best strategy in response to the *historical average* of the opponent's strategies.

Theorem A.3 (Robinson (1951))**.** *If two players use fictitious play in a zero-sum game, the historical average of their strategies will converge to a Nash equilibrium in the limit.*

Moreover, Robinson (1951) suggests and Daskalakis & Pan (2014) further explain how fictitious play will converge to achieve a value within ϵ of a Nash Equilibrium within finite time.

In our setting, fictitious play alternates between (1) fitting the density model to the historical average of policies, and (2) updating the policy with RL to minimize the log-density of the state, using a historical average of the density models:

$$q_{t+1} \leftarrow \arg\max_q \frac{1}{t}\sum_{i=1}^{t} \mathbb{E}_{\pi_t}[\log q(s)] \tag{6}$$

$$\pi_{t+1} \leftarrow \arg\max_q \mathbb{E}_\pi\left[-\log\left(\frac{1}{t}\sum_{i=1}^{t} q_i(s)\right)\right] \tag{7}$$

In practice, we can efficiently implement Equation 6 and avoid storing the policy parameters from every iteration by instead storing sampled states from each iteration. This is equivalent to maintaining an infinite-sized replay buffer, and fitting the density to the replay buffer at each iteration. We cannot perform the same trick for Equation 7, and instead resort to approximating the historical average of density models with the most recent iterate:

$$\pi_{t+1} \leftarrow \arg\max_q \mathbb{E}_\pi[-\log q_t(s)]$$

B ALTERNATIVE DERIVATION OF MIXTURE POLICY REWARD

The language of information theory gives an alternate view on the objective in Equation 4. First, we recall that mutual information can be decomposed in two ways

$$I(s; z) = \mathcal{H}[s] - \mathcal{H}[s \mid z] = \mathcal{H}[z] - \mathcal{H}[z \mid s]. \tag{8}$$

Thus, we have the following identity:

$$\mathcal{H}[s] = \mathcal{H}[s \mid z] + \mathcal{H}[z] - \mathcal{H}[z \mid s] \tag{9}$$

Plugging this identity into Equation 4, we see that our mixture policy approach is identical to single agent approach (Equation 2), albeit using a mixture of policies:

$$\mathcal{F}(\pi) = \mathbb{E}_{z\sim\pi(z)}[\mathbb{E}_{s\sim\pi(s|z)}[\log p^*(s)]] + \mathcal{H}[s] \tag{10}$$

C Additional Experiments

C.1 2-armed bandit

We give a didactic example that illustrates why of stochastic policies are necessary for distribution matching. We will examine a 2-arm bandit, and will assume that the target distribution over the two arms is $r(s) = (r_1, r_2)$. First, we consider trying to do distribution matching with a single, deterministic policy. It is clear that, for any reward function, the policy can only choose a single action. Thus, unless $r(s)$ is a Dirac Delta function, *a deterministic policy cannot match arbitrary state distributions.*

Next, we consider a mixture policy on the same, 2-armed bandit. We consider two agents, one which always chooses arm 1 and the second always chooses arm 2. Our only free parameters are the distribution over agents, $\pi(z)$. Setting this equal to the target distribution $p(z)$, the resulting mixture of policies matches the target distribution. Note that since actions are equivalent to states in the MDP, a maximum action policy would likewise succeed in distribution matching on this task. In general, action-entropy policies will fail to perform distribution matching, as illustrated in the next example.

C.2 2-state MDP

We consider another simple task to illustrate why action entropy is insufficient for distribution matching. We consider an MDP with two states and two actions, shown in Figure 5, and no reward function. A standard maximum action-entropy policy (e.g., SAC) will choose actions uniformly at random. However, because the self-loop in state A has a smaller probability that the self-loop in state B, the agent will spend 60% of its time in state B and only 40% of its time in state A. Thus, *maximum action entropy policies will not yield uniform state distributions*. We apply our method to learn a policy that maximizes *state* entropy. As shown in Figure 5, our method achieves the highest possible state entropy.

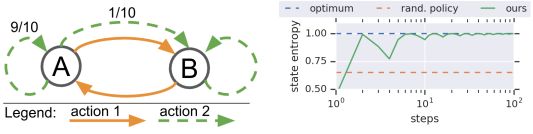
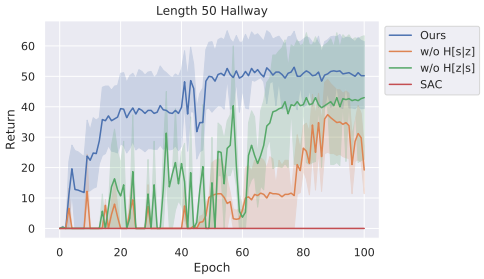


Figure 5: One a 2-state MDP, our algorithm converges to have a uniform marginal distribution over states while a maximum action-entropy policy does not.

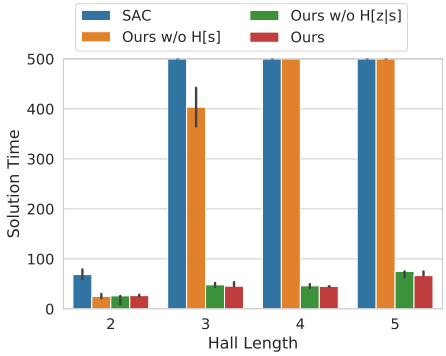
While we consider the case without a reward function, we can further show that *there does not exist reward function for which the optimal policy achieves a uniform state distribution.* It is enough to consider the relative reward on state A and B. If $r(A) > r(B)$, then the agent will take actions to remain at state A as often as possible; the optimal policies achieves remains at state A 91% of the time. If $r(A) < r(B)$, the optimal policy can remain at state B 100% of the time. if $r(A) = r(B)$, all policies are optimal and we have no guarantee that an arbitrary policy will have a uniform state distribution.

C.3 Ablation Analysis on 2D Navigation

We perform an ablation of our method on the 2D navigation task (Section 4) to understand the relative contribution of each component. Figure 6a compares our method to baselines that lack conditional state entropy, conditional action entropy, or both (i.e, SAC). We observe that both these ingredients are crucial to our method's success, and combining both terms further accelerates learning.



(a) Ablation analysis on 2D navigation task.



(b) Epochs to solve 3D navigation task (1 epoch = 5k steps).

Figure 6: (a) Our method relies heavily on both key differences from SAC. (b) Our method solves sparse reward navigation tasks significantly faster than SAC, and state entropy is especially important for 3D (Ant) navigation. All error bars show 50% confidence interval across 5 random seeds.

C.4 3D Navigation

To test whether our results scale to even more complex tasks, we also compare the exploration ability of SAC vs. our method on a complex 3D navigation task. The environment topology (shown in Figure 1) is the same as the 2D navigation tasks in Section 4, with three hallways of various lengths. However, the point mass agent is replaced with a MuJoCo Ant robot that has to learn to move its joints in order to move around the environment.

In order to measure exploration capability, we place a goal location at the end of every hallway, and measure the number of training epochs for the agent to reach any of the goals. In Figure 6b, we compare the performance of SAC vs. our method on the 3D navigation task. We see that our method solves the task significantly faster than SAC. We note that, if the maximum time horizon per episode is increased from 500 to 750 steps, SAC is able to solve 3D Navigation with hall length 3 after about 700 epochs (but still fails to solve for hall lengths > 3). On the other hand, our method is able to quickly solve 3D Navigation for all hall lengths with either 500 or 750 maximum episode steps, and we observe that state entropy is especially helpful in the 3D navigation task.

D Environments

D.1 StarEnv

StarEnv consists of an agent (point mass or Ant robot) that is spawned at a center location connecting n long corridors. Its task is to navigate to the end of a goal corridor $i \sim \text{Categorical}(p_1, \ldots, p_n)$, where $p_1, \ldots, p_n$ are known. Let g_i be the goal location for each corridor $i \in [n]$.

The state vector $s \in \mathcal{S}$ includes the xy-coordinates of the agent, $s_{\text{robot}} \in \mathbb{R}^3$. The state marginal distribution is

$$p^*(s) \propto \exp(r_{\text{goal}}(s))$$

where

$$r_{\text{goal}}(s) = \begin{cases} p_i & \text{if } \|s_{\text{robot}} - g_i\|_2^2 < \epsilon \text{ for any } i \in [n] \\ 0 & \text{otherwise} \end{cases}$$

For the StarEnv, we experimented with a variety of scaling factors for each component in the loss (state entropy, latent conditional entropy, action entropy, and reward scale). We found in preliminary experiments that for most cases, setting all entropy scales to 1 worked well. For the rest, we used the default SAC hyperparameters: learning rate 3e-4, discount factor 0.99, and initial target weight $\tau = 0.005$. For a density model, we simply discretized the agent's (x, y) coordinate and counted the visitation frequency of each bin.

Point Mass: We found a reward scale of 10 or 25 worked decently. For experiments involving Fig. 2b we found that an increased latent conditional entropy of 10 encouraged more significant exploration and used that for all results in that figure with a latent vector dimension larger than 1. Episodes had a maximum time horizon of 100 steps, after which they were terminated.

Ant: We found that a reward scale of 25 worked decently. The maximum time horizon per episode was set to 500 steps.

D.2 FETCHENV

FetchEnv consists of a robot with a single gripper arm, and a block object resting on top of a table surface. The robot's task is to move the object to a goal location $g \in \mathbb{R}^3$ which is not observed by the agent, thus requiring the agent to explore the table.

The state vector $s \in \mathcal{S}$ includes the action a taken by the robot and the xyz-coordinates $s_{\text{obj}}, s_{\text{robot}} \in \mathbb{R}^3$ of the object and the robot gripper, respectively. The target state marginal distribution is given by

$$p^*(s) \propto \exp\left(r_{\text{goal}}(s) + r_{\text{robot}}(s) + r_{\text{action}}(s)\right)$$

where the rewards

$$\begin{aligned} r_{\text{goal}}(s) &:= \mathbf{1}(s_{\text{obj}} \text{ is above the table surface}) \\ r_{\text{robot}}(s) &:= -\|s_{\text{obj}} - s_{\text{robot}}\|_2^2 \\ r_{\text{action}}(s) &:= -\|a\|_2^2 \end{aligned}$$

correspond to (1) a uniform distribution above the table surface, (2) distance between object and robot gripper, and (3) action penalty, respectively.

We used a VAE as our density model, demonstrating that SMM can be readily applied to high-dimensional tasks.

During test time, we sample a goal object location $g \in \mathbb{R}^3$ uniformly across the table, and finetune the baselines using the sparse reward $y_g(s) = \mathbf{1}(\|s_{\text{obj}} - g\| < \epsilon)$.