Bayesian Inference Approach for Entropy Regularized Reinforcement Learning with Stochastic Dynamics (Supplementary material)

Argenis Arriojas[1] **Jacob Adamczyk**[1] **Stas Tiomkin**[2] **Rahul V Kulkarni**[1]

[1]Department of Physics, University of Massachusetts Boston, Boston, Massachusetts, USA
[2]Department of Computer Engineering, San Jose State University, San Jose, California, USA

A EXPERIMENTAL DETAILS

A.1 RESULTS ON STOCHASTIC GRIDWORLD ENVIRONMENT WINDYFROZENLAKE

First, a modified version of OpenAI Gym's FrozenLake was used [Brockman et al., 2016], and named WindyFrozenLake. This is a grid world environment with discrete states and actions. There is uncertainty whenever a step is taken, with the wind affecting the transition probabilities between states after executing actions.

Table (S1) summarizes results for several randomly generated mazes of various sizes and complexities. The proposed method consistently finds the constrained optimization solutions for these mazes. Figure (S1) presents a more detailed view of the convergence properties of the algorithm, when several parameters are changed. Here we note that as the temperature decreases, more biasing iterations are required to attain the original dynamics. This trend becomes more evident for the more complex mazes with higher wind intensity and number of traps.

Table S1: Number of iterations required to find the optimal mapping between original (unconstrained) and policy-equivalent (constrained) dynamics for a large set of arbitrary mazes. For each combination of maze size and number of traps, 1620 mazes were randomly generated qualitatively resembling the one in Fig. (2), with varying wind intensity and values for the temperature $\beta \in \{2, 5, 20, 50\}$. For every maze a solution is found, taking between 2 and 126 iterations to find the corresponding biases. Figure (S1) offers a detailed faceted visualization where influence of beta and wind strength is also evaluated.

maze size	trap density	number of replicas	mean iterations	3 largest number of iterations
10	0.10	1620	3.68	[8, 8, 9]
	0.15	1620	3.82	[8, 8, 9]
	0.20	1620	3.84	[8, 9, 9]
	0.25	1620	3.83	[8, 8, 10]
20	0.10	1620	3.59	[8, 8, 8]
	0.15	1620	3.69	[8, 9, 14]
	0.20	1620	3.73	[8, 9, 13]
	0.25	1620	3.80	[12, 13, 23]
30	0.10	1620	3.68	[12, 13, 13]
	0.15	1620	3.80	[22, 24, 39]
	0.20	1620	3.85	[21, 26, 30]
	0.25	1620	4.04	[21, 71, 126]

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

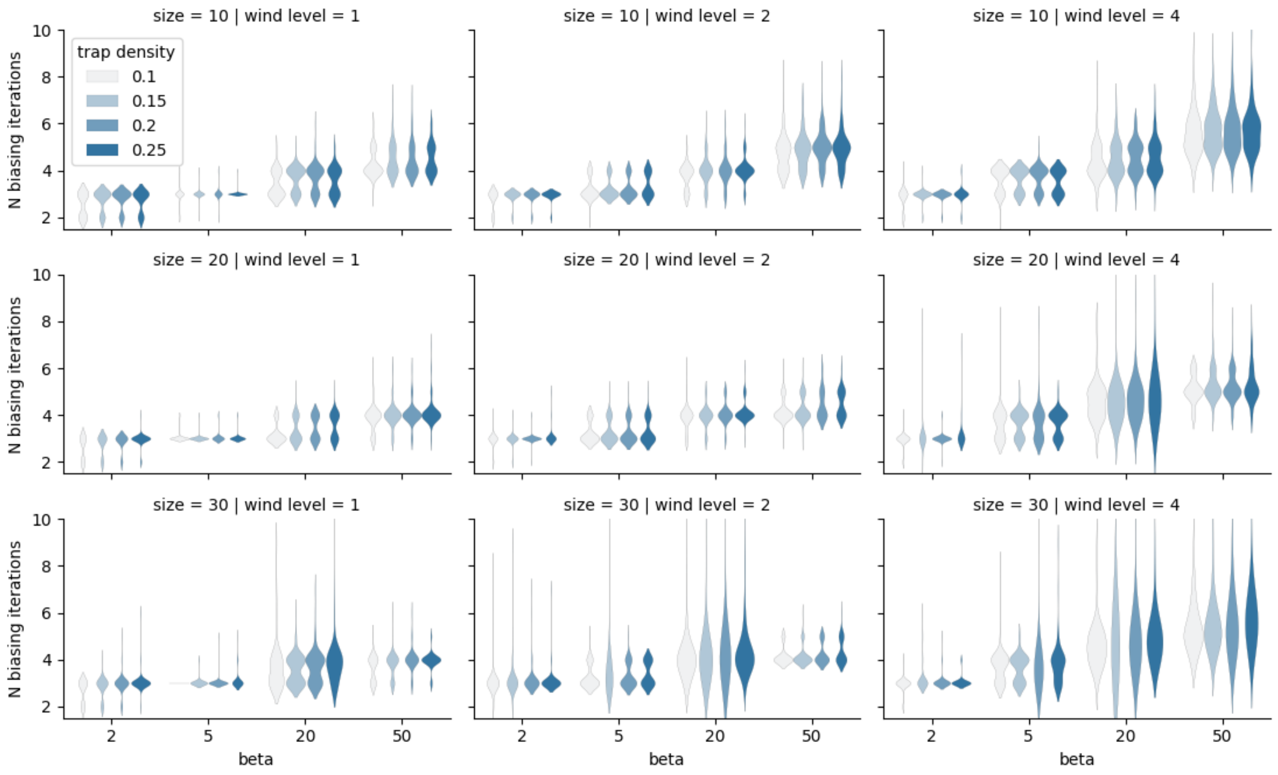

Figure S1: Number of biasing iterations required for solving random windy frozen lake environments, in terms of 4 different variables: size of the maze, wind strength level, trap density and beta used for the solution. We observe that most of the mazes required between 2 and 6 biasing iterations to converge. In general, we can appreciate that for lower temperatures (higher beta), more iterations are required. For higher temperatures, maze size and trap density have no major impact. For the cases were variables combine to produce difficult environments (strong wind, many traps), a higher number of biasing iterations are required. In this visualization we have removed outlier values above 50 iterations.

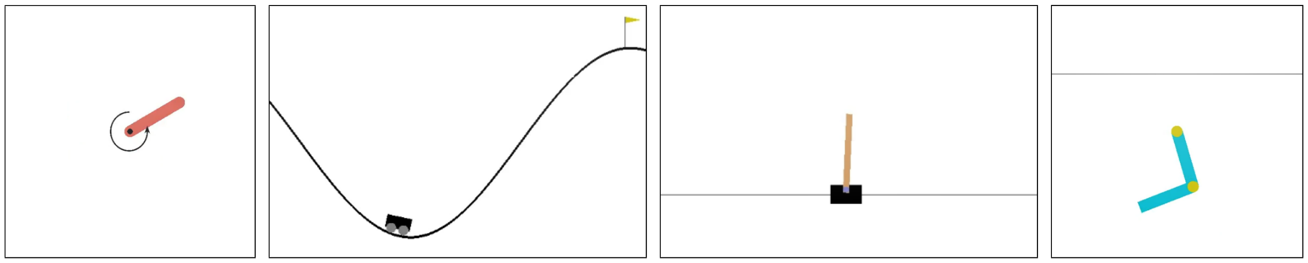
(a) Pendulum-v0 (b) MountainCar-v0 [Moore, 1990] (c) CartPole-v1 [Barto et al., 1983] (d) Acrobot-v1 [Sutton, 1996, Geramifard et al., 2015]

Figure S2: Reinforcement learning environments with continuous observations, made available by OpenAI under the MIT License [Brockman et al., 2016]. For these environments, the observation space was discretized. See appendix (A) and Table (S2)

Table S2: Experiments on environments with continuous observation spaces. $|\mathcal{S}|$ denotes the size of the observation state-space after discretization, and $\mathcal{A}$ the corresponding action-space. $|f_m|$ and $|\tau_m|$ are the maximal force and torque available to the agent, respectively. For Pendulum, the continuous action space was discretized to 3 actions. Each environment was solved with a given β hyper-parameter as specified. Each of these problems was solved in k iterations for the biasing process.

Environment	$\|\mathcal{S}\|$	$\mathcal{A}$	β	k
Pendulum	16^2	$\{-\|\tau_m\|, 0, +\|\tau_m\|\}$	1	28
MountainCar	12^2	$\{-\|f_m\|, 0, +\|f_m\|\}$	2	38
CartPole	8^4	$\{-\|f_m\|, +\|f_m\|\}$	10	29
Acrobot	12^4	$\{-\|\tau_m\|, 0, +\|\tau_m\|\}$	25	58

A.2 EXPERIMENTS ON DISCRETIZED VERSIONS OF ENVIRONMENTS WITH CONTINUOUS STATE SPACE

To demonstrate the usefulness of the method on more complex examples, we proceeded to solve multiple reinforcement learning environments which fall under the classic control category. In this section we present a description of the four continuous Gym environments used. Table (S2) presents a description of the state-space discretization parameters used for these environments, as well as the β values used to solve them and the number of biasing iterations required for convergence to the original system transition dynamics.

For each environment, the following steps were followed

- Make rewards instantaneous. This is necessary for CartPole and MountainCar.
- Digitize/Discretize state space. In the case of Pendulum, actions also need to be discretized.
- Sample transition dynamics and reward function. This is done exhaustively by manually setting the environment state, taking all available actions and observing the resulting state and reward. We sampled 500 repetitions of each combination of state and action.
- Choose a β parameter and compute the constrained optimization solution by using the estimated dynamics and rewards, and Algorithm (1). At each iteration, the intermediate policies are stored for later evaluation.
- Perform evaluation of intermediate policies and optimal policy.

A detailed description of each of these four environments is presented in the following subsections.

A.2.1 Pendulum

Shown in Figure (S2a). This is a direct control environment. The goal is to swing up a pendulum by using a torque as input control. The 2-dimensional observation space is continuous, consisting of the angular position and the angular velocity of the pendulum. A continuous-valued torque τ can be applied with $\tau \in [-2, 2]$. The reward function is proportional to the angle of the pendulum, with the lowest reward given in the lower position.

A.2.2 MountainCar

Shown in Figure (S2b). In this environment the goal is to reach the flag at the top of a hill. The car power is not enough to climb the hill in one shot, and instead needs to swing back and forth to gain enough energy. The continuous state space is 2-dimensional and consists of the linear position and speed of the car. There are 3 discrete actions available: accelerate left, no acceleration, and accelerate right. Each step has reward -1 until the flag is reached, where a reward of 0 is given and the episode terminates.

A.2.3 CartPole

Shown in Figure (S2c). A cart can move horizontally on a track, while balancing a pole in a vertical position. The goal is to continue balancing the pole for as long as possible, without moving beyond the edges of the environment. The 4-dimensional observation is comprised of the linear position and velocity of the cart, and the angular position and velocity of the pole. A reward of $+1$ is given at each step while the bar is in the upright position. The episode is terminated when the pole reaches a threshold angle, and 0 reward is given.

A.2.4 Acrobot

Shown in Figure (S2d). This is a double pendulum system, which is a chaotic system. The goal is to apply torque to the joint between the two pendulums, and make the second pendulum go above a vertical threshold. The observation space has 4 dimensions, with the angular positions and velocities of each joint. All rewards are -1, until the goal is attained and 0 reward is given.

B LEARNING BIASES IN A MODEL-FREE SETTING

In the main paper we have shown how to obtain the biasing functions for the transition dynamics and rewards to solve the constrained optimization problem. The results presented rely in knowing both the dynamics and rewards beforehand. In this section, we present an algorithm for learning the biasing functions from experience in a model-free setting.

We first note the connections between the optimal soft-value functions $Q(s,a)$ and $V(s)$ discussed in the main text with the dominant eigenvalue ($\rho = e^{-\theta}$) and the corresponding left eigenvector ($u(s,a)$) of the tilted matrix $\widetilde{P}$. as previously established in [Arriojas et al., 2023], given by the following equations:

$$Q(s,a) = -\theta N + \frac{1}{\beta}\log u(s,a) \tag{1}$$

$$V(s) = -\theta N + \frac{1}{\beta}\log \sum_{a} \pi^0(a|s)u(s,a), \tag{2}$$

where θ is the so-called bulk free energy and N is the number of steps in the trajectory. In the following, we will work with the dominant eigenvalue ($\rho = e^{-\theta}$) and the corresponding left eigenvector ($u(s,a)$) and derived quantities. The results obtained can be expressed in terms of the optimal value functions using the preceding equations.

We start by considering the eigenvalue equation for the tilted matrix $\widetilde{P}$ and its left eigenvector u

$$\sum_{(s',a')} e^{\beta r(s,a)}\pi^0(a'|s')p(s'|s,a)u(s',a') = \rho u(s,a) \tag{3}$$

and after applying biases to r and p we obtain

$$\sum_{(s',a')} e^{\beta(r(s,a)+\delta(s,a))}\pi^0(a'|s')p(s'|s,a)b(s'|s,a)u_b(s',a') = \rho_b u_b(s,a) \tag{4}$$

then for some fixed biasing functions $b(s'|s,a)$ and $\delta(s,a)$, we adapt the update equations from [Arriojas et al., 2023] resulting in the following update equations that can be used to estimate the corresponding u_b and ρ_b

$$u_b(s,a) \leftarrow (1-\alpha)u_b(s,a) + \alpha \frac{e^{\beta r_b(s,a)}}{\rho_b} u_b(s',a') b(s'|s,a) \tag{5}$$

$$\rho_b \leftarrow (1-\alpha_\rho)\rho_b + \alpha_\rho e^{\beta r(s,a)} \frac{u_b(s',a')}{u_b(s,a)} b(s'|s,a). \tag{6}$$

Now let

$$\psi(s) = e^{-\beta V(s)} = \frac{1}{\sum_a \pi^0(a|s) u(s,a)} \tag{7}$$

From Eqn. (17) and the normalization condition we can write the dynamics bias function

$$b(s'|s,a) = \frac{\psi(s')}{\sum_{s'} \psi(s') p(s'|s,a)}. \tag{8}$$

Using this form to rewrite Eqn. (18), we obtain the reward bias function

$$\beta\delta(s,a) = \log\left[\sum_{s'} \psi(s') p(s'|s,a)\right] - \sum_{s'} \log\left[\psi(s')\right] p(s'|s,a). \tag{9}$$

After rewriting the biasing functions b and δ in terms of ψ, we see that it suffices to find estimations for the expectations $\mathbb{E}_{s'|s,a}\left[\psi(s')\right]$ and $\mathbb{E}_{s'|s,a}\left[\log\left(\psi(s')\right)\right]$, which can be achieved through the following update equations

$$\langle\psi\rangle_{(s,a)} \leftarrow (1-\alpha_\psi)\langle\psi\rangle_{(s,a)} + \alpha_\psi \psi(s') \tag{10}$$

$$\langle\log\psi\rangle_{(s,a)} \leftarrow (1-\alpha_\psi)\langle\log\psi\rangle_{(s,a)} + \alpha_\psi \log\psi(s'). \tag{11}$$

With all four update equations, we can construct an algorithm that is capable of learning the biases $b(s'|s,a)$ and $\delta(s,a)$, which enables the learning of the left eigenvector $u_b(s,a)$ for the biased problem. The corresponding pseudocode for this method is presented in Algorithm (S1). We have tested this algorithm on the windy cliff environment presented in Figure (1). Results for this experiment are presented in Figure (5) of the main text.

In conclusion, we have developed an off-policy, model-free algorithm that successfully finds the maximum entropy optimal policy for a stochastic RL environment, while avoiding the optimistic agent issue. The ideas here presented may serve of inspiration to find an equivalent solution for the case of continuous state-action spaces.

References

Argenis Arriojas, Jacob Adamczyk, Stas Tiomkin, and Rahul V. Kulkarni. Entropy regularized reinforcement learning using large deviation theory. *Phys. Rev. Res.*, 5:023085, May 2023. doi: 10.1103/PhysRevResearch.5.023085. URL `https://link.aps.org/doi/10.1103/PhysRevResearch.5.023085`.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

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

Alborz Geramifard, Christoph Dann, Robert H. Klein, William Dabney, and Jonathan P. How. Rlpy: A value-function-based reinforcement learning framework for education and research. *Journal of Machine Learning Research*, 16(46):1573–1578, 2015. URL `http://jmlr.org/papers/v16/geramifard15a.html`.

Andrew W Moore. *Efficient Memory-Based Learning for Robot Control.* PhD thesis, University of Cambridge, 1990.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. Touretzky, M. C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996. URL `https://proceedings.neurips.cc/paper/1995/file/8f1d43620bc6bb580df6e80b0dc05c48-Paper.pdf`.

Algorithm S1 Find optimal policy from samples

```
Parameters: inverse temperature $\beta$; update rates $\alpha$, $\alpha_\rho$, $\alpha_\psi$; prior policy $\pi^0$; $N_{\text{epochs}}$; $N_{\text{bias}}$
Input: experience samples of tuples $(s, a, r, s', a')$
Output: optimal policy $\pi^*$

1. Initialize $\psi(s) \leftarrow 1$; $\langle\psi\rangle_{(s,a)} \leftarrow 1$ and $\langle\log\psi\rangle_{(s,a)} \leftarrow 0$
repeat
  repeat
    for $s, a, r, s', a'$ in experience do
      2. $\langle\psi\rangle_{(s,a)} \leftarrow (1-\alpha_\psi)\langle\psi\rangle_{(s,a)} + \alpha_\psi\psi(s')$
      3. $\langle\log\psi\rangle_{(s,a)} \leftarrow (1-\alpha_\psi)\langle\log\psi\rangle_{(s,a)} + \alpha_\psi\log\psi(s')$
    end for
  until relative error between epochs $< 0.01\%$

  4. $b(s'|s,a) \leftarrow \psi(s')/\langle\psi\rangle_{(s,a)}$
  5. $\delta(s,a) \leftarrow \frac{1}{\beta}(\log\langle\psi\rangle_{(s,a)} - \langle\log\psi\rangle_{(s,a)})$
  6. $\Delta \leftarrow \max(\delta(s,a))$
  7. $u(s,a) \leftarrow 1$
  8. $\rho \leftarrow e^{-\beta}$
  repeat
    for $s, a, r, s', a'$ in experience do
      9. $r_b \leftarrow r + \delta(s,a) - \Delta$
      10. $u(s,a) \leftarrow (1-\alpha)u(s,a) + \alpha\frac{e^{\beta r_b}}{\rho}u(s',a')b(s'|s,a)$
      11. $\rho \leftarrow (1-\alpha_\rho)\rho + \alpha_\rho e^{\beta r_b}\frac{u(s',a')}{u(s,a)}b(s'|s,a)$
    end for
  until completed $N_{\text{epochs}}$

  12. $\psi(s) \leftarrow \left(\sum_a \pi^0(a|s)u(s,a)\right)^{-1}$
until completed $N_{\text{bias}}$ iterations

13. $\pi^*(a|s) \leftarrow \pi^0(a|s)u(s,a)$
14. normalize $\pi^*$
return: $\pi^*$
```