MAXIMUM ENTROPY RL (PROVABLY) SOLVES SOME ROBUST RL PROBLEMS

Benjamin Eysenbach
Carnegie Mellon University, Google Brain
beysenba@cs.cmu.edu

Sergey Levine
UC Berkeley, Google Brain

ABSTRACT

Many potential applications of reinforcement learning (RL) require guarantees that the agent will perform well in the face of disturbances to the dynamics or reward function. In this paper, we prove theoretically that standard maximum entropy RL is robust to some disturbances in the dynamics and the reward function. While this capability of MaxEnt RL has been observed empirically in prior work, to the best of our knowledge our work provides the first rigorous proof and theoretical characterization of the MaxEnt RL robust set. While a number of prior robust RL algorithms have been designed to handle similar disturbances to the reward function or dynamics, these methods typically require adding additional moving parts and hyperparameters on top of a base RL algorithm. In contrast, our theoretical results suggest that MaxEnt RL by itself is robust to certain disturbances, without requiring any additional modifications. While this does not imply that MaxEnt RL is the best available robust RL method, MaxEnt RL does possess a striking simplicity and appealing formal guarantees.

1 INTRODUCTION

Many real-world applications of reinforcement learning (RL) require control policies that not only maximize reward but continue to do so when faced with environmental disturbances, modeling errors, or errors in reward specification. These disturbances can arise from human biases and modeling errors, from non-stationary aspects of the environment, or actual adversaries acting in the environment. The main contribution of this paper is a formal proof that the policies produced by maximum entropy (MaxEnt) RL are robust to some natural types of disturbances.

Prior work has observed that MaxEnt RL algorithms produce policies that are robust to disturbances in the environment (Haarnoja et al., 2018b; Huang et al., 2019). Intuitively, this makes sense: stochastic policies implicitly inject noise into the actions during training, thereby preparing themselves for deployment in environments with stochastic disturbances. The MaxEnt RL objective can be viewed as injecting as much noise as possible while still maximizing the task reward. Seen from another perspective, MaxEnt RL produces policies that choose actions randomly, with probability proportional to the likelihood that they solve the task (Ziebart, 2010; Attias, 2003). For example, in the robot pushing task shown in Fig. 1, the policy learned by MaxEnt RL pushes the white puck to the goal using many possible routes. In contrast, (standard) RL learns a deterministic policy, always using the same route to get to the goal. Now, imagine that this environment is perturbed, for example by adding a red barrier in Fig. 1. While the policy learned by (standard) RL always collides with this obstacle, the policy learned by MaxEnt RL uses many routes to solve the task, and some fraction of these

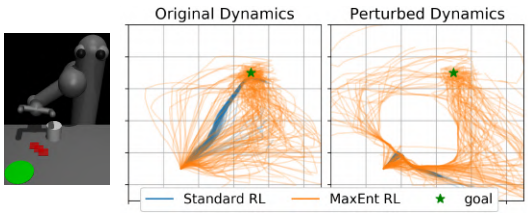


Figure 1: **MaxEnt RL is robust to disturbances.** *(Left)* We applied both standard RL and MaxEnt RL to a manipulation task without obstacles, but added obstacles (red squares) during evaluation. We then plot the position of the object when evaluating the learned policies *(Center)* on the original environment and *(Right)* on the new environment with an obstacle. The stochastic policy learned by MaxEnt RL often navigates around the obstacle, whereas the deterministic policy from standard RL almost always collides with the obstacle.

routes continue to solve the task even when the obstacle is present. While similar intuition has been offered in prior work (Levine, 2018; Abdolmaleki et al., 2018; Lee et al., 2019), to the best of our knowledge there has been no formal proof of this statement. The main contribution of this paper is a proof that MaxEnt RL maximizes a lower bound on reward in the face of disturbances.

This problem of learning robust policies that achieve high reward in the face of *adversarial* environmental disturbances has been well studied in the literature, often going under the name *robust RL* (Bagnell et al., 2001; Nilim & Ghaoui, 2003; Morimoto & Doya, 2005; Pinto et al., 2017). We note that this is different from maximizing the *average* reward across many environments, as done by methods such as domain randomization (Sadeghi & Levine, 2016; Rajeswaran et al., 2016; Peng et al., 2018). We include a review of related work in Appendix A.

The general recipe followed by most robust RL algorithms is to take an existing RL algorithm and then modify the update rule or reward function. For example, Nilim & Ghaoui (2003) modify the Bellman backup in a Q-learning algorithm to take into account uncertainty in what the next state might be. In contrast, we argue that existing MaxEnt RL algorithms, without any modification, already act as robust RL algorithms. While we do not claim that MaxEnt RL will outperform purpose-designed robust RL algorithms, the striking simplicity of MaxEnt RL (compared with alternative robust RL algorithms) may make it an appealing choice in practical settings. Our main contribution is a set of proofs showing that standard MaxEnt RL optimizes lower bounds on several possible robust objectives, reflecting a degree of robustness to changes in the dynamics and to certain changes in the reward. We formally character the robust set in both cases, discuss how this set depends on the choice of reward function, and provide simple numerical experiments to validate these results. Our aim is not to show that MaxEnt RL is a better robust RL method than prior works, but rather to provide analysis of how and when MaxEnt RL policies are robust.

A number of recent prior works have viewed MaxEnt RL through the lens of regularization, formally showing that entropy-regularization can decreases variance and accelerates convergence (Fox et al., 2015; Geist et al., 2019). Indeed, the benefits of entropy regularization for online learning are well established (Shalev-Shwartz et al., 2011; Hoeven et al., 2018). Whereas these prior works highlight how MaxEnt RL is robust to *internal* or algorithmic disturbances, our work complements these prior methods by showing that MaxEnt RL is also robust to *external* or environmental disturbances.

2 PRELIMINARIES

For notation, we assume that an agent observes states $\mathbf{s_t}$, takes actions $\mathbf{a_t} \sim \pi(\mathbf{a_t} \mid \mathbf{s_t})$, and obtains rewards $r(\mathbf{s_t}, \mathbf{a_t})$. The initial state is sampled $\mathbf{s_1} \sim p_1(\mathbf{s_1})$, and subsequent states are sampled $\mathbf{s_{t+1}} \sim p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$. Episodes have T steps, which we summarize as a trajectory $\tau \triangleq (\mathbf{s_1}, \mathbf{a_1}, \cdots, \mathbf{s_T}, \mathbf{a_T})$. Without loss of generality, we can assume that rewards are undiscounted, as any discount can be addressed by modifying the dynamics to transition to an absorbing state with probability $1-\gamma$. The standard RL objective is:

$$\arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=1}^{T} r(\mathbf{s_t}, \mathbf{a_t})\right], \qquad \text{where} \quad \pi(\tau) = p_1(\mathbf{s_1}) \prod_{t=1}^{T} p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) \pi(\mathbf{a_t} \mid \mathbf{s_t}) d\tau$$

is the distribution over trajectories. The MaxEnt RL objective is to maximize the sum of expected reward and conditional action entropy. Formally, we define this objective in terms of a policy π, the dynamics p and the reward function r:

$$J_{\text{MaxEnt}}(\pi; p, r) \triangleq \mathbb{E}_{\substack{\mathbf{a_t}\sim\pi(\mathbf{a_t}|\mathbf{s_t}) \\ \mathbf{s_{t+1}}\sim p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t})}}\left[\sum_{t=1}^{T} r(\mathbf{s_t}, \mathbf{a_t}) + \mathcal{H}_\pi[\mathbf{a_t} \mid \mathbf{s_t}]\right], \tag{1}$$

where $\mathcal{H}_\pi[\mathbf{a_t} \mid \mathbf{s_t}] = \int \pi(\mathbf{a_t} \mid \mathbf{s_t}) \log \frac{1}{\pi(\mathbf{a_t}|\mathbf{s_t})} d\mathbf{a_t}$ denotes the Shannon entropy of the policy's conditional distribution over actions. We do not include a discount factor to simplify notation. We will quantify robustness by measuring the entropy-regularized reward of a policy when evaluated on a *new* reward function $\tilde{r}$ or dynamics function $\tilde{p}$. More precisely, we will use the following *robust RL* objective, where the new reward function or dynamics function is chosen adversarially:

$$\max_\pi \min_{\tilde{p}\in\tilde{\mathcal{P}}, \tilde{r}\in\tilde{\mathcal{R}}} J_{\text{MaxEnt}}(\pi; p, r).$$

We refer to the sets of dynamics $\tilde{\mathcal{P}}$ and reward functions $\tilde{\mathcal{R}}$ as *robust sets*, and our goal is to characterize these sets.

3 MAXENT RL AND ROBUST CONTROL

In this section we will prove some aspects of the folklore claim that MaxEnt RL is robust to disturbances in the environment. Formally, we will show that MaxEnt RL maximizes a lower bound on reward when deployed in an environment with an adversarially-chosen dynamics or reward function. We will also characterize which (new) reward functions and dynamics functions we can expect MaxEnt RL to be robust against. Because of space limitations, we defer the discussion of reward function robustness to Appendix B.

Our analysis of MaxEnt RL's robustness to dynamics aims to answer the following questions: **(1)** What is the relationship between the MaxEnt RL objective and the performance under an adversarially-chosen dynamics function? **(2)** When we run MaxEnt RL with one dynamics function, what other dynamics functions will the learned policy also be robust against? Our main result answers both of these questions:

Theorem 3.1. *Let an MDP with dynamics $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$ and reward function $r(\mathbf{s_t}, \mathbf{a_t}) > 0$ be given. Define the transformed reward function $\bar{r}(\mathbf{s_t}, \mathbf{a_t}) \triangleq \frac{1}{T} \log r(\mathbf{s_t}, \mathbf{a_t}) + \mathcal{H}[\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}]$. Then there exists a constant $\epsilon > 0$ such that the MaxEnt RL objective J_{MaxEnt} (Eq. 1) with dynamics p and reward function $\bar{r}$ maximizes a lower bound on the robust RL objective defined by robust set $\tilde{\mathcal{P}}$:*

$$\min_{\tilde{p} \in \tilde{\mathcal{P}}(\pi)} J_{MaxEnt}(\pi; \tilde{p}, r) \geq \exp(J_{MaxEnt}(\pi; p, \bar{r}) + \log T),$$

where the adversary chooses from the set $\tilde{\mathcal{P}}$ of dynamics that satisfy the following constraint:

$$\mathbb{E}_{\substack{\pi(\mathbf{a_t}|\mathbf{s_t}), \\ p(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t})}} \left[\sum_t \log \iint e^{\log p(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t}) - \log \tilde{p}(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t})} d\mathbf{a_t}' d\mathbf{s_{t+1}}' \right] \leq \epsilon.$$

The proof can be found in Appendix D. The set of dynamics we are robust against, $\tilde{\mathcal{P}}$, has an intuitive interpretation as those that are sufficiently close to the original dynamics $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$. For example, in the case of linear-Gaussian dynamics (described at the end of this subsection), this set contains all dynamics functions for which the next state is (on average) within some Euclidean distance of the next state from the training dynamics. The constraint value ϵ is the adversary's "budget"; the adversary can choose to make large changes to the dynamics in just a few states, or to make smaller changes to the dynamics in many states.

This theorem provides a recipe for converting between MaxEnt RL and robust control problems. On the one hand, if we want to acquire a policy that optimize a reward under many possible dynamics, we can simply run MaxEnt RL with the reward function $\bar{r}(\mathbf{s_t}, \mathbf{a_t}) = \log r(\mathbf{s_t}, \mathbf{a_t}) + \mathcal{H}[\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}]$. In many settings (e.g., dynamics with constant additive noise), we might assume that $\mathcal{H}[\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}]$ is approximately constant, in which case we simply set the MaxEnt RL reward to $\bar{r}(\mathbf{s_t}, \mathbf{a_t}) = \log r(\mathbf{s_t}, \mathbf{a_t})$. On the other hand, this result that when we apply MaxEnt RL with one reward function r, the policy we learn is guaranteed to get high (exponentiated) reward on all dynamics functions in the robust set $\tilde{\mathcal{P}}$, even when the dynamics function is adversarially chosen.

Example. Consider an MDP with 1D, bounded, states and actions $\mathbf{s_t}, \mathbf{a_t} \in [-10, 10]$. The dynamics are $p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) = \mathcal{N}(\mathbf{s_{t+1}}; \mu = A\mathbf{s_t} + B\mathbf{a_t}, \sigma = 1)$ the reward function is $r(\mathbf{s_t}, \mathbf{a_t}) = \|\mathbf{s_t}\|_2^2$, and episodes have finite length T. Note that the dynamics entropy $\mathcal{H}[\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}]$ is constant. We will consider an adversary that modifies the dynamics by increasing the standard deviation to $\sqrt{2}$ and shifts the bias by an amount β, resulting in the following dynamics: $\tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) = \mathcal{N}(\mathbf{s_{t+1}}; \mu = A\mathbf{s_t} + B\mathbf{a_t} + \beta, \sigma = \sqrt{2})$. The robust set defined in Theorem 3.1 specifies that the adversary choose any dynamics, as long as β satisfies

$$\frac{1}{2}\beta^2 + \log(8\sqrt{\pi}) + \log(20) \leq \epsilon.$$

Then, regardless of which dynamics the adversary chooses, we can learn a policy that is guaranteed to get high reward by running MaxEnt RL with the reward function specified by Theorem 3.1: $\bar{r}(\mathbf{s_t}, \mathbf{a_t}) = \frac{2}{T} \log \|\mathbf{s_t}\|_2$. See Appendix D.4 for the full derivation.

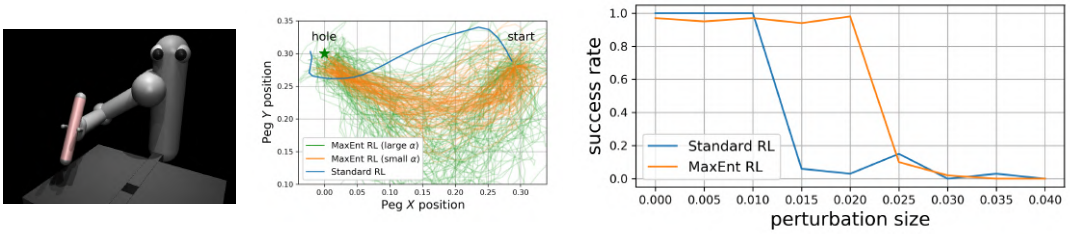


Figure 2: **Robustness to Adversarial Disturbances**: On the peg insertion environment, we perturbed the location of the hole to minimize the policy's performance. MaxEnt RL is more robust against these adversarial disturbances to the environment.

3.1 Numerical Simulations

We conducted a number of numerical simulations to verify our theoretical result that MaxEnt RL is robust to disturbances in the reward function and dynamics function. Because of space limitations, we highlight just two experiments here and include the rest in Appendix E.

Our analysis proves that MaxEnt RL is robust against a strong type of adversary that can perturb the environment based on the behavior of the current policy. We verify this capability in the following experiment. We trained both algorithms on a peg insertion task from Eysenbach et al. (2017), shown in Fig. 2. To visualize the learned policies, we plot the position of the peg for many rollouts, observing that the standard RL policy always takes the same route to the goal, wheres the MaxEnt RL policy uses many different routes to get to the goal. We therefore expect that if the location of the hole is adversarially perturbed, the MaxEnt RL policy will continue to perform better. To test this robustness, for each policy we found the worst-case perturbation to the hole location using CMA-ES (Hansen, 2016). For small perturbations, both standard RL and MaxEnt RL achieve a success rate near 100%. However, for perturbations that are 1.5cm or 2cm in size, only MaxEnt RL continues to solve the task. For larger perturbations neither method can solve the task. In summary, this experiment highlights that MaxEnt RL is robust to *adversarial* disturbances, as predicted by the theory.

Our theory also predicts that MaxEnt RL should be robust to perturbations in the reward function (see Appendix B. To validate this theoretical result, we applied MaxEnt RL on four continuous control tasks from the standard OpenAI Gym (Brockman et al., 2016) benchmark, and use SAC (Haarnoja et al., 2018a) as our MaxEnt RL algorithm. We use SVG (Heess et al., 2015) as our RL baseline because it also learns a stochastic policy, but differs from SAC by excluding the entropy term. We also compared with fictitious play (Brown, 1951), a method from the game theory literature. In the RL setting, this method corresponds to modifying standard RL to use an adversarially-chosen reward function for each Bellman update. To evaluate each algorithm, we computed both the average cumulative reward, as well as the average worst-case reward. The worst-case reward was defined as in Equation 2. Both MaxEnt RL and standard RL can maximize the cumulative reward (see Fig. 9 in Appendix E), but only MaxEnt RL succeeds as maximizing the worst-case reward, as shown in Fig. 3. In summary, this experiment supports our proof that MaxEnt RL is solves the robust RL problem for the set of rewards specified in Theorem B.1.

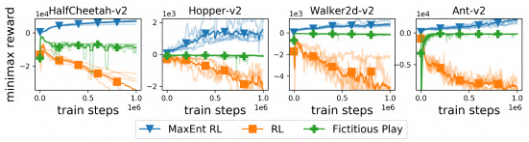


Figure 3: **MaxEnt RL policies are robust to disturbances in the reward function.** We trained policies with MaxEnt RL, standard RL and Fictitious play, evaluating each on adversarial disturbances to the reward function. While all methods get high reward on the training reward function (see Fig. 9 in the Appendix), only those policies learned by MaxEnt RL policies are robust to disturbances in the reward function

4 Discussion

In this paper we showed that the stochastic policies produced by MaxEnt RL effectively maximize a lower bound on the robust RL objective and are therefore robust against a set of adversarial perturbations to the dynamics and the reward function. This result suggests that MaxEnt RL may be particularly well equipped to solve problems with uncertainty in the dynamics and rewards, including not just robust RL, but also problems with *implicit* uncertainty, such as POMDPs, offline RL, and meta-learning tasks. Robustness lies at the core of generalization, so our results also suggest that MaxEnt RL policies might generalize to new tasks more adeptly than those learned by standard RL. We encourage researchers working in these problem areas to use MaxEnt RL (with a tuned temperature) as a simple baseline.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy in policy learning. *arXiv preprint arXiv:1811.11214*, 2018.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Hagai Attias. Planning by probabilistic inference. In *AISTATS*. Citeseer, 2003.

J Andrew Bagnell, Andrew Y Ng, and Jeff G Schneider. Solving uncertain markov decision processes. 2001.

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

George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Jack Clark and Dario Amodei. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 1282–1289. PMLR, 2019.

John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1906.05253*, 2019.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. *arXiv preprint arXiv:1901.11275*, 2019.

Peter D Grünwald, A Philip Dawid, et al. Game theory, maximum entropy, minimum discrepancy and robust bayesian decision theory. *the Annals of Statistics*, 32(4):1367–1433, 2004.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pp. 2829–2838. PMLR, 2016.

Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Chris Harris, Vincent Vanhoucke, et al. Tf-agents: A library for reinforcement learning in tensorflow, 2018. URL `https://github.com/tensorflow/agents`.

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*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018b.

Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

Dirk Hoeven, Tim Erven, and Wojciech Kotłowski. The many faces of exponential weights in online learning. In *Conference On Learning Theory*, pp. 2067–2092. PMLR, 2018.

Shiyu Huang, Hang Su, Jun Zhu, and Ting Chen. Svqn: Sequential variational soft q-learning networks. In *International Conference on Learning Representations*, 2019.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *arXiv preprint arXiv:1905.02175*, 2019.

Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pp. 143–173. Springer, 2012.

Kyungjae Lee, Sungyub Kim, Sungbin Lim, Sungjoon Choi, and Songhwai Oh. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *arXiv preprint arXiv:1902.00137*, 2019.

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

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

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

Arnab Nilim and Laurent Ghaoui. Robustness in markov decision problems with uncertain transition matrices. *Advances in neural information processing systems*, 16:839–846, 2003.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3803–3810. IEEE, 2018.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2817–2826. JMLR. org, 2017.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pp. 1369–1376, 2007.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056. ACM, 2009.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of regularization in rl. *arXiv preprint arXiv:2003.14089*, 2020.

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

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100. PMLR, 2020.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

Kemin Zhou, John Comstock Doyle, Keith Glover, et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.

Brian D. Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.

Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Maximum causal entropy correlated equilibria for markov games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pp. 207–214. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

A Related Work

Prior work on MaxEnt RL offers a number of explanations for why one might prefer MaxEnt RL to standard RL. We summarize four common explanations:

1. Exploration. MaxEnt RL is often motivated as performing good exploration. Unlike many other RL algorithms, such as DQN (Mnih et al., 2015) and DDPG (Lillicrap et al., 2015), MaxEnt RL performs exploration and policy improvement with the same (stochastic) policy.

2. Regularization. Recent work has studied how the entropy term in MaxEnt RL regularizes the policy and the value function (Fox et al., 2015; Neu et al., 2017; Geist et al., 2019; Vieillard et al., 2020). These works note that MaxEnt RL can improve the convergence rates and stability *during training*. In contrast, our work will show that MaxEnt RL is robust to external environmental disturbances, rather than internal algorithmic disturbances.

3. Probabilistic inference. A number of prior works motivate MaxEnt RL as an inference problem (Attias, 2003; Todorov, 2007; Toussaint, 2009; Abdolmaleki et al., 2018; Haarnoja et al., 2018a; Levine, 2018). These connections are intriguing because they hint that the same language (inference) can be used for both perception, localization, and control.

4. Easier optimization. Finally, some prior work (Williams & Peng, 1991; Ahmed et al., 2018) argues that the entropy bonus added by MaxEnt RL makes the optimization landscape smoother.

This paper will argue that MaxEnt RL should be preferred for another, fifth reason:
5. Robustness (this paper). MaxEnt RL implicitly solves a robust RL problem. This suggests that MaxEnt RL might be preferred in settings where the dynamics or reward function are unknown, adversarially chosen, or changing over time. We emphasize that such robustness is useful not only in the presence of adversaries, but more broadly in cases where the dynamics or reward function are misspecified or vary across time.

We will study robustness using the problem statement of robust RL, a problem studied in a number of prior works (Bagnell et al., 2001; Nilim & Ghaoui, 2003; Morimoto & Doya, 2005; Pinto et al., 2017). The motivation for this problem setting is that policies learned by standard RL are often not robust to disturbances (Rajeswaran et al., 2016; Peng et al., 2018). While some prior robust RL methods are quite general, they often require many additional hyperparameters. The problem of robust RL is slightly different from robust control and concepts such as H-∞ control and Lyapunov functions, which are primarily focused on the stability of a system around a fixed point, independent of any reward function (Zhou et al., 1996; Doyle et al., 2013). The problem of robust RL is also different than the problem of learning transferable or generalizable RL agents (Lazaric, 2012; Zhang et al., 2018; Justesen et al., 2018; Cobbe et al., 2019), which focuses on the average-case performance on new environments, rather than the worst-case performance. In this paper will prove that MaxEnt RL solves a subset of robust RL problems. Thus, while MaxEnt RL is not as general as some prior methods, it is simpler, requiring just one additional hyperparameter for the entropy.

B Robustness to Adversarial Reward Functions

While most prior work on robust RL has focused on robustness to dynamics, in many applications we also want the learned policy to be robust against misspecified reward functions. This issue is particularly pertinent when the reward function is designed by hand. Indeed, it is well known that RL agents are adept at exploiting misspecified reward functions (Amodei et al., 2016; Clark & Amodei, 2016). The challenges of reward specification, and the risks of deploying a policy that is not robust to misspecified rewards, motivates us to extend the robust RL framework to include robustness against perturbations to the reward function.

The main result of this section is that MaxEnt RL is robust to some degree of misspecification in the reward function. We study this capability through two concrete questions: **(1)** What is the relationship between the MaxEnt RL objective and the performance under an adversarially-chosen reward function? **(2)** When we run MaxEnt RL with one reward function, what other reward functions $\tilde{r}$ will the learned policy be robust against. The following result answers both of these questions.

Theorem B.1. *For any dynamics p and reward function r, there exists a positive constant $\epsilon > 0$ such that the MaxEnt RL objective J_{MaxEnt} is equivalent to the robust control objective defined by robust*

set $\tilde{\mathcal{R}}(\pi)$:

$$\min_{\tilde{r}\in\tilde{\mathcal{R}}(\pi)} \mathbb{E}\Big[\sum_t \tilde{r}(\mathbf{s_t}, \mathbf{a_t})\Big] = J_{\textit{MaxEnt}}(\pi; p, r) \quad \forall\pi,$$

where the adversary chooses a reward function from the set $\tilde{\mathcal{R}}$ of reward functions that satisfy the following constraint:

$$\mathbb{E}\Big[\sum_t \log \int \exp(r(\mathbf{s_t}, \mathbf{a_t}) - \tilde{r}(\mathbf{s_t}, \mathbf{a_t}))d\mathbf{a_t}'\Big] \le \epsilon. \tag{2}$$

Thus, when we use MaxEnt RL with some reward function r, the policy obtained is guaranteed to also get high reward all similar reward functions $\tilde{r}$ that satisfy Eq. 2. The proof can be found in Appendix D.1, and closely mirrors a result from Grünwald et al. (2004); Ziebart et al. (2011) that relates the principle of maximum entropy to robustness in supervised learning problems. While this result is somewhat elementary, it will be useful as an intermediate step when proving robustness to dynamics in the next subsection.

The robust set $\tilde{\mathcal{R}}$ depends on the policy, so the adversary has the capability of looking at which states the policy visits before choosing the adversarial reward function $\tilde{r}$. MaxEnt RL is also robust against a *weaker* adversary that only considers reward functions that are close to the true reward function at every state:

Corollary B.1.1. *Assume episodes have length T. Then there exists a positive constant ϵ such that MaxEnt RL objective $J_{\textit{MaxEnt}}$ is a lower bound on the robust RL objective defined by a task-agnostic robust set $\tilde{\mathcal{R}}_2$:*

$$\min_{\tilde{r}\in\tilde{\mathcal{R}}_2} \mathbb{E}\Big[\sum_t \tilde{r}(\mathbf{s_t}, \mathbf{a_t})\Big] \ge J_{\textit{MaxEnt}}(\pi; p, r) \quad \forall\pi,$$

where the the robust set $\tilde{\mathcal{R}}_2$ contains all reward functions satisfying the following constraint:

$$\log \int \exp(r(\mathbf{s_t}, \mathbf{a_t}) - \tilde{r}(\mathbf{s_t}, \mathbf{a_t}))d\mathbf{a_t}' \le \frac{\epsilon}{T}.$$

The proof follows immediately by observing $\tilde{\mathcal{R}}_2 \subseteq \tilde{\mathcal{R}}(\pi)$ for all policies π. This result is useful for two reasons. First, it describes what other reward functions a MaxEnt RL policy will also get high reward on. Second, this result can also be read in reverse. If we have a set of reward functions and want a policy that will get high reward on all of them, we can acquire such a policy through MaxEnt RL. Appendix D.7 describes a precise approach for doing this when our set of reward functions do not confirm to Eq. 2.

While this corollary is weaker than the result in Theorem B.1, it will nonetheless be useful for building intuition in the following worked examples.

Example 1. Define a 2-armed bandit with the following reward function:

$$r(\mathbf{a}) = \begin{cases} 2 & \mathbf{a} = 1 \\ 1 & \mathbf{a} = 2 \end{cases}, \; R = \left\{ \tilde{r} \;\middle|\; \int \exp(r(\mathbf{a}) - \tilde{r}(\mathbf{a}))d\mathbf{a} \le \epsilon \right\}.$$

The corresponding robust set is

$$\tilde{\mathcal{R}} = \left\{ \tilde{r} \middle| \int \exp(r(\mathbf{a}) - \tilde{r}(\mathbf{a}))d\mathbf{a} \le \epsilon \right\}$$

Fig. 4 (left) plots the original reward function, r as a red dot, and the collection of reward functions, $\tilde{\mathcal{R}}$, as a blue line. In the center plot we plot the expected reward for each reward in $\tilde{\mathcal{R}}$ as a function of the action $\mathbf{a}$. The robust RL problem in this setting is to choose the policy whose worst-case reward (dark blue line) is largest. The right plot shows the MaxEnt RL objective. We observe that the robust RL objective and the MaxEnt RL objectives are equivalent.

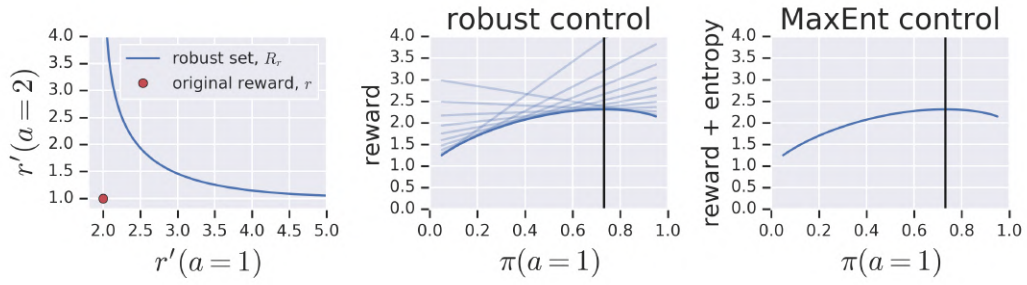


Figure 4: **MaxEnt RL and Robustness to Adversarial Reward Functions**: *(Left)* Applying MaxEnt RL to one reward function (red dot) yields a policy that is guaranteed to get high reward on many other reward functions (blue curve). *(Center)* For each reward function $(r(a=1), r(a=2))$ on that blue curve, we evaluate the expected return of a stochastic policy. The robust RL problem (for rewards) is to choose the policy whose worst-case reward (dark blue line) is largest. *(Right)* Plotting the MaxEnt RL objective (Eq. 1) for those same policies, we observe that the MaxEnt RL objective is identical to the robust RL objective .

Example 2. We use a task with a 1D, bounded action space $\mathcal{A} = [-10, 10]$ and a reward function composed of a task-specific reward r_{task} and a penalty from deviating from some desired action a^*:

$$r(\mathbf{s}, \mathbf{a}) \triangleq r_{\text{task}}(\mathbf{s}, \mathbf{a}) - (\mathbf{a} - \mathbf{a}^*)^2.$$

The adversary will perturb this desired action by an amount Δa and decrease the weight on the control penalty by 50%, resulting in the following reward function:

$$\tilde{r}(\mathbf{s}, \mathbf{a}) \triangleq r_{\text{task}}(\mathbf{s}, \mathbf{a}) - \frac{1}{2}(\mathbf{a} - (\mathbf{a}^* + \Delta\mathbf{a}))^2.$$

In this example, robust set from Corollary D.6 is defined as reward functions corresponds to perturbations $\Delta\mathbf{a}$ that satisfy

$$\Delta\mathbf{a}^2 + \frac{1}{2}\log(2\pi) + \log(20) \leq \frac{\epsilon}{T}.$$

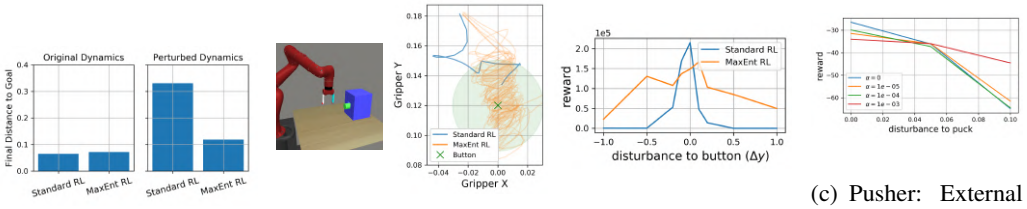
See Appendix D.2 for the full derivation. Thus, MaxEnt RL with reward function r yields a policy that is robust against adversaries that perturb $\mathbf{a}^*$ by at most $\Delta\mathbf{a} = \mathcal{O}(\sqrt{\epsilon})$.

C Additional Experiments

This section will present numerical simulations verifying our theoretical result that MaxEnt RL is robust to disturbances in the reward function and dynamics function. We emphasize that we do not attempt to show that MaxEnt RL outperforms prior robust RL methods, only that it is more robust than standard RL.

Dynamics robustness. We demonstrate the robustness of MaxEnt RL to disturbances in the dynamics of a few simulated control tasks. Our experiments here study (1) whether MaxEnt RL is more robust than standard RL, (2) whether MaxEnt RL is robust to disturbances introduced in the middle of an episode, (3) how MaxEnt RL differs from just adding noise to the policy learned by stadard RL, and (4) whether MaxEnt RL continues to be robust when these disturbances are optimized in an adversarially manner. We study these questions through five experiments.

First, we applied both MaxEnt RL and standard RL to the pushing task shown in Fig. 1, where a new obstacle was added during evaluation. We describe our precise implementation of MaxEnt RL, standard RL, and all environments in Appendix E. As shown in Fig. 1, the policy learned by MaxEnt RL has learned many ways to push the puck to the goal, and thus continues to succeed on some fraction of trials once the obstacle is added. In contrast, the policy learned by standard RL always pushes the puck to the goal along the same path. Thus, when this path is blocked by the obstacle, the standard RL policy always fails to reach the goal. Quantitatively, Fig. 5a measures the final



(a) Pusher: New Obstacle

(b) Button: Moved Goal

(c) Pusher: External Force

Figure 5: **MaxEnt RL policies are robust to disturbances in dynamics.** *(Left)* We take the `Pusher-v2` environment, shown in Fig. 1, and add a new obstacle for evaluation. While both standard RL and MaxEnt RL successfully push the puck to the goal in the original environment (↓ distance is better), standard RL fails at the task when the obstacle is added, while MaxEnt RL still succeeds on many trials. *(Center)* We apply both MaxEnt RL and standard RL to the button pushing environment shown above, observing that MaxEnt RL takes stochastic paths to the button whereas standard RL takes deterministic paths. We then perturb the position of the button and plot the average reward of both methods(↑ reward is better). MaxEnt RL achieves higher reward than standard RL on the perturbed environments. *(Right)* We construct a variant of the pushing task where an external disturbance is added to the puck in the middle of the episode. Evaluating the policies learned by MaxEnt RL with varying temperatures, we observe that higher temperatures yield more robust policies. Note that standard RL corresponds to using a temperature of 0.

distance between the puck and the goal on the original environment and on the perturbed environment containing the obstacle. We observe that both standard RL and MaxEnt RL achieve comparably small distances on the original environment. However, in the perturbed environment, standard RL fails for nearly every trial, resulting in a significantly larger average distance. In contrast, MaxEnt RL success for a high fraction of trials, resulting in a smaller average distance.

Second, we applied both MaxEnt RL and standard RL to the button pushing task shown in Fig. 5b (left). Plotting the gripper position for many trajectories (Fig. 5b center), we observe that standard RL always takes the same path to the goal, whereas MaxEnt RL takes many different paths to the goal, and continues taking random actions after reaching the goal. We therefore expect that MaxEnt RL will be more robust to disturbances to the position of the button. To test the hypothesis, we varied the position of the button along the y axis, plotting the performance of both standard RL and MaxEnt RL for different perturbations. As shown in Fig. 5b (right), standard RL achieves slightly higher reward when no perturbations are present, but its reward is much smaller for even small disturbances. In contrast, MaxEnt RL achieves much higher reward on these perturbed environments. Note that the robust RL objective corresponds to an adversary choosing the disturbance Δy that minimizes the policy's reward; MaxEnt RL performs much better according to this worst-case objective.

Whereas the first two experiments used perturbations that were fixed throughout the episode, our third experiment aimed to study whether MaxEnt RL is robust to dynamic disturbances introduced in the middle of the episode. For this experiment, we took the pushing task show in Fig. 1 and, instead of adding an obstacle, perturbed the XY position of the puck after 20 time steps. By evaluating the reward of a policy while varying the size of this perturbation, we can study the range of disturbances to which MaxEnt RL is robust. As shown in Fig. 5c, we measured the performance of MaxEnt RL policies trained with different entropy coefficients α; note that $\alpha = 0$ corresponds to standard RL. As expected, all methods perform well on the environment without any disturbances, with lower temperatures achieving slightly higher reward. For larger disturbances, only the MaxEnt RL method trained with the largest temperature ($\alpha = 1e-3$) continues to solve the task. This experiment demonstrates that MaxEnt RL can be robust to dynamics disturbances. It also highlights that the temperature controls the degree of robustness, with small temperatures resulting in less robust policies.

Our fourth experiment compares MaxEnt RL to stochastic policies learned by standard RL, akin to Heess et al. (2015); Gu et al. (2016). In many situations, simply adding noise to the deterministic policy found by standard RL can make that policy robust to some disturbances. MaxEnt RL does something more complex, dynamically adjusting the entropy of the policy depending on the current state. This capability allows MaxEnt RL policies to have lower entropy in some states as needed to

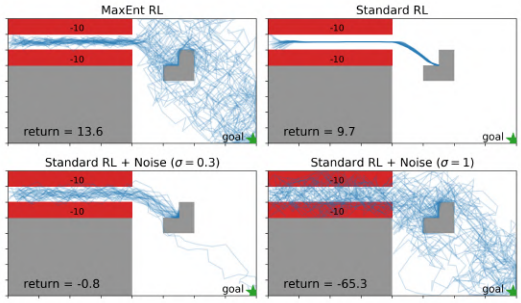


Figure 6: **MaxEnt RL is not standard RL + noise.** *(Top Left)* In this 2D navigation task, the policy learned by MaxEnt RL varies its entropy throughout the episode, using low entropy initially to avoid hitting the low-reward obstacles, and then using high entropy in the second half of the episode, once clear of these obstacles. When a new "L"-shaped obstacle is added for evaluation, the policy learned by MaxEnt RL often navigates around the obstacle, getting an average reward of 13.6 (the maximum reward is 16.8). *(Top Right)* The policy learned by standard RL always gets stuck at the obstacle. *(Bottom Row)* Simply adding independent Gaussian noise to the the policy found by standard RL enables it to navigate around the obstacle but causes it to collide with the low-reward obstacles, causing it to achieve lower reward than the MaxEnt RL policy.

ensure high reward. We studied this capability in the 2D navigation task shown in Fig. 6. The agent starts near the top left corner and gets reward for navigating to the bottom right hand corner, but incurs a large cost for entering the red regions. As expected, the policy learned by MaxEnt RL has low entropy initially to avoid colliding with the red obstacles, and then increases its entropy in the second half of the episode. To study robustness, we introduce a new "L"-shaped obstacle for evaluation. As expected, the policy learned by MaxEnt RL often navigates around this obstacle, whereas the policy from standard RL always collides with this obstacle. By adding independent Gaussian noise to the actions from standard RL policy, we can enable that policy to navigate around the obstacle if we add enough noise. However, because this noise is added independently at each time step, this policy (bottom right) collides with the red obstacles many times and achieves a very low reward. In summary, this experiment highlights that MaxEnt RL is not the same as adding independent noise to policy learned by standard RL; the ability to dynamically adjust this noise enables the MaxEnt RL policy to continue to achieve high reward, even in the presence of external disturbances to the environment.

D PROOFS

D.1 PROOF OF THEOREM B.1

We now provide a proof of Theorem B.1.

Proof. We start by restating the objective for robust control with rewards:

$$\min_{\tilde{r} \in \tilde{\mathcal{R}}(\pi)} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t \tilde{r}(\mathbf{s_t}, \mathbf{a_t}) \right].$$

The KKT conditions guarantee that there exists a threshold ϵ, used in defining the robust set, such that the constrained objective above is equivalent to the following relaxed objective:

$$\min_{\tilde{r}} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t \tilde{r}(\mathbf{s_t}, \mathbf{a_t}) + \log \int \exp(r(\mathbf{s_t}, \mathbf{a_t}) - \tilde{r}(\mathbf{s_t}, \mathbf{a_t})) d\mathbf{a_t}' \right].$$

To clarify exposition, we parameterize $\tilde{r}$ by its deviation from the original reward, $\Delta r(\mathbf{s_t}, \mathbf{a_t}) \triangleq r(\mathbf{s_t}, \mathbf{a_t}) - \tilde{r}(\mathbf{s_t}, \mathbf{a_t})$, resulting in the following objective:

$$\min_{\Delta r} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t r(\mathbf{s_t}, \mathbf{a_t}) - \Delta r(\mathbf{s_t}, \mathbf{a_t}) + \log \int \exp(\Delta r(\mathbf{s_t}, \mathbf{a_t})) d\mathbf{a_t}' \right].$$

Recognizing the terms with Δr as a Fenchel dual, we obtain the desired result:

$$\mathbb{E}\left[\sum_t r(\mathbf{s_t}, \mathbf{a_t}) - \log \pi(\mathbf{a_t} \mid \mathbf{s_t})\right] = \mathbb{E}\left[\sum_t r(\mathbf{s_t}, \mathbf{a_t}) + \mathcal{H}_\pi[\mathbf{a_t} \mid \mathbf{s_t}]\right].$$

□

D.2 Worked Example of Reward Robustness

We compute the penalty for Example 1 in Section B using a Gaussian integral:

$$\begin{aligned}
\textsc{Penalty}(\tilde{r}, \mathbf{s}) &= \log \int_{-10}^{10} \exp(r(\mathbf{s}, \mathbf{a}) - \tilde{r}(\mathbf{s}, \mathbf{a}))d\mathbf{a} \\
&= \log \int_{-10}^{10} \exp(-(\mathbf{a} - \mathbf{a}^*)^2 + \frac{1}{2}(\mathbf{a} - (\mathbf{a}^* + \Delta\mathbf{a}))^2)d\mathbf{a} \\
&= \log \int_{-10}^{10} \exp(-\frac{1}{2}(\mathbf{a} - (\mathbf{a}^* + \Delta\mathbf{a}))^2 + \Delta\mathbf{a}^2)d\mathbf{a} \\
&= \Delta\mathbf{a}^2 + \frac{1}{2}\log(2\pi) + \log(20).
\end{aligned}$$

D.3 Proof of Theorem 3.1

Proof. We now provide the proof of Theorem 3.1. The first step will be to convert the variation in dynamics into a sort of variation in the reward function. The second step will convert the constrained robust control objective into an unconstrained (penalized) objective. The third step will show that the penalty is equivalent to action entropy.

Step 1: Dynamics variation → reward variation. Our aim is to obtain a lower bound on the following objective:

$$\min_{\tilde{p} \in \tilde{\mathcal{P}}(\pi)} \mathbb{E}_{\substack{\mathbf{a_t} \sim \pi(\mathbf{a_t}|\mathbf{s_t}) \\ \mathbf{s_{t+1}} \sim p(\mathbf{s_t}|\mathbf{s_t}, \mathbf{a_t})}} \left[\sum_t r(\mathbf{s_t}, \mathbf{a_t})\right].$$

We start by taking a log-transform of this objective, noting that this transformation does not change the optimal policy as the log function is strictly monotone increasing.

$$\log \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t r(\mathbf{s_t}, \mathbf{a_t})\right].$$

Note that we assumed that the reward was positive, so the logarithm is well defined. We can write this objective in terms of the adversarial dynamics using importance weights.

$$\begin{aligned}
&\log \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim \tilde{p}(s'|s,a)}} \left[\left(\prod_t \frac{p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})}{\tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})}\right) \sum_t r(\mathbf{s_t}, \mathbf{a_t})\right] \\
&= \log \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim \tilde{p}(s'|s,a)}} \left[\exp\left(\log\left(\sum_t r(\mathbf{s_t}, \mathbf{a_t})\right) + \sum_t \log p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) - \log \tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})\right)\right] \\
&\overset{(a)}{\geq} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim \tilde{p}(s'|s,a)}} \left[\log\left(\sum_t r(\mathbf{s_t}, \mathbf{a_t})\right) + \sum_t \log p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) - \log \tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})\right] \\
&= \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim \tilde{p}(s'|s,a)}} \left[\log\left(\frac{1}{T}\sum_t r(\mathbf{s_t}, \mathbf{a_t})\right) + \log T + \sum_t \log p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) - \log \tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})\right] \\
&\overset{(b)}{\geq} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim \tilde{p}(s'|s,a)}} \left[\sum_t \frac{1}{T} \log r(\mathbf{s_t}, \mathbf{a_t}) + \log p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) - \log \tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})\right] + \log T.
\end{aligned} \tag{3}$$

Both inequalities are applications of Jensen's inequality. As before, our assumption that the rewards are positive ensures that the logarithms remain well defined. While the adversary is choosing the dynamics under which we will evaluate the policy, we are optimizing a lower bound which depends on a *different* dynamics function. *This trick allows us to analyze adversarially-chosen dynamics as perturbations to the reward.*

Step 2: Relaxing the constrained objective To clarify exposition, we will parameterize the adversarial dynamics as a deviation from the true dynamics:

$$\Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t}) = \log p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) - \log \tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}).$$

Using this notation, we can write the lower bound on the robust control problem (Eq. 3) as follows:

$$\min_{\Delta r} \mathbb{E}_{\substack{\mathbf{a_t}\sim\pi(\mathbf{a_t}|\mathbf{s_t}),\\ \mathbf{s_{t+1}}\sim p(\mathbf{s_t}|\mathbf{s_t},\mathbf{a_t})}} \left[\sum_t \frac{1}{T} \log r(\mathbf{s_t}, \mathbf{a_t}) - \Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t})\right] + \log T \tag{4}$$

$$\text{s.t.} \quad \mathbb{E}_{\substack{\mathbf{a_t}\sim\pi(\mathbf{a_t}|\mathbf{s_t}),\\ \mathbf{s_{t+1}}\sim p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t})}} \left[\sum_t \log \iint e^{\Delta r(\mathbf{s_{t+1}}', \mathbf{s_t}, \mathbf{a_t}')} d\mathbf{a_t}' d\mathbf{s_{t+1}}'\right] \leq \epsilon \tag{5}$$

$$\text{and} \quad \int \underbrace{p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) e^{-\Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t})}}_{\tilde{p}(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t})} d\mathbf{s_{t+1}} = 1 \quad \forall \mathbf{s_t}, \mathbf{a_t}. \tag{6}$$

The constraint in Eq. 5 is the definition of the set of adversarial dynamics $\tilde{\mathcal{P}}$, and the constraint in Eq. 6 ensures that $\tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})$ represents a valid probability density function.

Note that $\Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t}) = 0$ is always a *feasible* solution to this constrained optimization problem. Thus, the KKT conditions guarantee that there exists a threshold $\epsilon > 0$ such that this constrained objective is equivalent to the following relaxed objective:

$$\min_{\Delta r} \mathbb{E}_{\substack{\mathbf{a_t}\sim\pi(\mathbf{a_t}|\mathbf{s_t}),\\ \mathbf{s_{t+1}}\sim p(\mathbf{s_{t+1}}|\mathbf{s_t},\mathbf{a_t})}} \left[\sum_t \frac{1}{T} \log r(\mathbf{s_t}, \mathbf{a_t}) - \Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t}) + \log \iint e^{\Delta r(\mathbf{s_{t+1}}', \mathbf{s_t}, \mathbf{a_t}')} d\mathbf{a_t}' d\mathbf{s_{t+1}}'\right] + \log T$$

$$\text{s.t.} \quad \int p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t}) e^{-\Delta r(\mathbf{s_{t+1}}', \mathbf{s_t}, \mathbf{a_t}')} d\mathbf{s_{t+1}} = 1 \quad \forall \mathbf{s_t}, \mathbf{a_t}.$$

We now observe that adding a constant c to Δr does not change the objective but scales constraint by e^{-c}. Thus, constraining Δr to represent a probability distribution does not affect the solution (value) to the optimization problem, so we can ignore this constraint without loss of generality:

$$\min_{\Delta r} \mathbb{E}_{\substack{a\sim\pi(a|s),\\ s'\sim p(s'|s,a)}} \left[\sum_t \frac{1}{T} \log r(\mathbf{s_t}, \mathbf{a_t}) - \Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t}) + \log \iint e^{\Delta r(\mathbf{s_{t+1}}, \mathbf{s_t}, \mathbf{a_t})} d\mathbf{a_t}' d\mathbf{s_{t+1}}'\right] + \log T.$$

Step 3: The penalty is the Fenchel dual of action entropy. We now recognize this objective as the Fenchel dual of the log-sum-exp function, which is known to equal the negative entropy. Thus, the optimization problem is

$$\mathbb{E}_{\substack{a\sim\pi(a|s),\\ s'\sim p(s'|s,a)}} \left[\sum_t \frac{1}{T} \log r(\mathbf{s_t}, \mathbf{a_t}) - \log \pi(\mathbf{a_t} \mid \mathbf{s_t}) - \log p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})\right] + \log T,$$

where the third term inside the expectation is the entropy of the dynamics.

Summary. We have thus shown the follow:

$$\min_{\tilde{p}\in\tilde{\mathcal{P}}} \log J_{\text{MaxEnt}}(\pi; \tilde{p}, r) \geq J_{\text{MaxEnt}}(\pi; p, \bar{r}) + \log T.$$

Taking the exponential transform of both sides, we obtain the desired result.

□

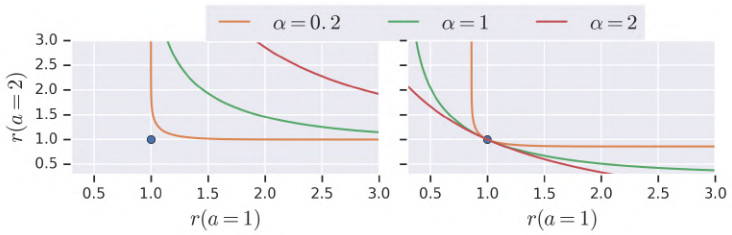


Figure 7: **Effect of Temperature**: *(Left)* For a given reward function (blue dot), we plot the robust sets for various values of the temperature. Somewhat surprisingly, it appears that increasing the temperature decreases the set of reward functions that MaxEnt is robust against. *(Right)* We examine the opposite: for a given reward function, which other robust sets might contain this reward function. We observe that robust sets corresponding to larger temperatures (i.e., the red curve) can be simultaneously robust against more reward functions than robust sets at lower temperatures.

D.4 Worked Example of Dynamics Robustness

We calculate the penalty using a Gaussian integral:

$$\begin{aligned}
\log \iint \frac{p(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})}{\tilde{p}(\mathbf{s_{t+1}} \mid \mathbf{s_t}, \mathbf{a_t})} d\mathbf{a_t} d\mathbf{s_{t+1}} &= \log \iint 2\exp(-\frac{1}{2}(\mathbf{s_{t+1}} - (A\mathbf{s_t} + B\mathbf{a_t}))^2 + \frac{1}{4}(\mathbf{s_{t+1}} - (A\mathbf{s_t} + B\mathbf{a_t} - \beta))^2) d\mathbf{a_t} d\mathbf{s_{t+1}} \\
&= \log \iint 2\exp(-\frac{1}{4}(\mathbf{s_{t+1}} - (A\mathbf{s_t} + B\mathbf{a_t} - \beta))^2 + \frac{1}{2}\beta^2) d\mathbf{a_t} d\mathbf{s_{t+1}} \\
&= \log \left(2\sqrt{4\pi} \int_{-10}^{10} \exp(\frac{1}{2}\beta^2) d\mathbf{a_t} \right) \\
&= \frac{1}{2}\beta^2 + \log(8\sqrt{\pi}) + \log(20).
\end{aligned}$$

We make two observations about this example. First, the penalty would be infinite if the adversary dynamics $\tilde{p}$ had the same variance as the true dynamics, p: the adversary must choose dynamics with higher variance. Second, the penalty depends on the size of the state space. More precisely, the penalty depends on the region over which the adversary applies their perturbation. The adversary can incur a smaller penalty by applying the perturbation over a smaller range of states.

D.5 Temperatures

Many algorithms for MaxEnt RL (Haarnoja et al., 2018a; Fox et al., 2015; Nachum et al., 2017) include a temperature $\alpha > 0$ to balance the reward and entropy terms:

$$J_{\text{MaxEnt}}(\pi, r) = \mathbb{E}_\pi \left[\sum_{t=1}^{T} r(\mathbf{s_t}, \mathbf{a_t}) \right] + \alpha \mathcal{H}_\pi[\mathbf{a_t} \mid \mathbf{s_t}].$$

We can gain some intuition into the effect of this temperature on the set of reward functions to which we are robust. In particular, including a temperature α results in the following robust set:

$$\begin{aligned}
R_r^\alpha &= \left\{ r(\mathbf{s_t}, \mathbf{a_t}) + \alpha u_{\mathbf{s_t}}(\mathbf{a_t}) \mid u_{\mathbf{s_t}}(\mathbf{a_t}) \geq 0 \; \forall \mathbf{s_t}, \mathbf{a_t} \text{ and } \int_{\mathcal{A}} e^{-u_{\mathbf{s_t}}(\mathbf{a_t})} d\mathbf{a_t} \leq 1 \; \forall \mathbf{s_t} \right\} \\
&= \left\{ r(\mathbf{s_t}, \mathbf{a_t}) + u_{\mathbf{s_t}}(\mathbf{a_t}) \mid u_{\mathbf{s_t}}(\mathbf{a_t}) \geq 0 \; \forall \mathbf{s_t}, \mathbf{a_t} \text{ and } \int_{\mathcal{A}} e^{-u_{\mathbf{s_t}}(\mathbf{a_t})/\alpha} d\mathbf{a_t} \leq 1 \; \forall \mathbf{s_t} \right\}. \qquad (7)
\end{aligned}$$

In the second line, we simply moved the temperature from the objective to the constraint by redefining $u_{\mathbf{s_t}}(\mathbf{a_t}) \to \frac{1}{\alpha} u_{\mathbf{s_t}}(\mathbf{a_t})$.

We visualize the effect of the temperature in Fig. 7. First, we fix a reward function r, and plot the robust set R_r^α for varying values of α. Fig. 7 (left) shows the somewhat surprising result that increasing the temperature (i.e., putting more weight on the entropy term) makes the policy *less*

robust. In fact, the robust set for higher temperatures is a strict subset of the robust set for lower temperatures:

$$\alpha_1 < \alpha_2 \implies R_r^{\alpha_2} \subseteq R_r^{\alpha_2}.$$

This statement can be proven by simply noting that the function $e^{-\frac{x}{\alpha}}$ is an increasing function of α in Equation 7. It is important to recognize that being robust against more reward functions is not always desirable. In many cases, to be robust to everything, an optimal policy must do nothing.

We now analyze the temperature in terms of the converse question: if a reward function r' is included in a robust set, what other reward functions are included in that robust set? To do this, we take a reward function r', and find robust sets R_r^α that include r', for varying values of α. As shown in Fig. 7 (right), if we must be robust to r' and use a high temperature, the only other reward functions to which we are robust are those that are similar, or pointwise weakly better, than r'. In contrast, when using a small temperature, we are robust against a wide range of reward functions, including those that are highly dissimilar from our original reward function (i.e., have higher reward for some actions, lower reward for other actions). Intuitively, increasing the temperature allows us to simultaneously be robust to a larger set of reward functions.

D.6 MaxEnt Solves Robust Control for Rewards

In Sec. B, we showed that MaxEnt RL is equivalent to some robust-reward problem. The aim of this section is to go backwards: given a set of reward functions, can we formulate a MaxEnt RL problem such that the robust-reward problem and the MaxEnt RL problem have the same solution?

Lemma D.1. *For any collection of reward functions R, there exists another reward function r such that the MaxEnt RL policy w.r.t. r is an optimal robust-reward policy for R:*

$$\arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^T r(\mathbf{s_t}, \mathbf{a_t}) \right] + \mathcal{H}_\pi[a \mid s] \subseteq \arg\max_\pi \min_{r' \in R} \mathbb{E}_\pi \left[\sum_{t=1}^T r'(\mathbf{s_t}, \mathbf{a_t}) \right].$$

We use set containment, rather than equality, because there may be multiple solutions to the robust-reward control problem.

Proof. Let π^* be a solution to the robust-reward control problem:

$$\pi^* \in \arg\max_\pi \min_{r_i \in R} \mathbb{E}_\pi \left[\sum_{t=1}^T r_i(\mathbf{s_t}, \mathbf{a_t}) \right].$$

Define the MaxEnt RL reward function as follows:

$$r(\mathbf{s_t}, \mathbf{a_t}) = \log \pi^*(\mathbf{a_t} \mid \mathbf{s_t}).$$

Substituting this reward function in Equation 1, we see that the unique solution is $\pi = \pi^*$. □

Intuitively, this theorem states that we can use MaxEnt RL to solve *any* robust-reward control problem that requires robustness with respect to any arbitrary set of rewards, if we can find the right corresponding reward function r for MaxEnt RL. One way of viewing this theorem is as providing an avenue to sidestep the challenges of robust-reward optimization. Unfortunately, we will still have to perform robust optimization to learn this magical reward function, but at least the cost of robust optimization might be amortized. In some sense, this result is similar to Ilyas et al. (2019).

D.7 Finding the Robust Reward Function

In the previous section, we showed that a policy robust against any set of reward functions R can be obtained by solving a MaxEnt RL problem. However, this requires calculating a reward function r^* for MaxEnt RL, which is not in general an element in R. In this section, we aim to find the MaxEnt reward function that results in the optimal policy for the robust-reward control problem. Our main idea is to find a reward function r^* such that its robust set, R_{r^*}, contains the set of reward functions we want to be robust against, R. That is, for each $r_i \in R$, we want

$$r_i(\mathbf{s_t}, \mathbf{a_t}) = r^*(\mathbf{s_t}, \mathbf{a_t}) + u_{\mathbf{s_t}}(\mathbf{a_t}) \quad \text{for some } u_{\mathbf{s_t}}(\mathbf{a_t}) \text{ satisfying} \quad \int_{\mathcal{A}} e^{-u_{\mathbf{s_t}}(\mathbf{a_t})} d\mathbf{a_t} \leq 1 \; \forall \mathbf{s_t}.$$

Replacing u with $r' - r^*$, we see that the MaxEnt reward function r must satisfy the following constraints:

$$\int_{\mathcal{A}} e^{r^*(\mathbf{s_t},\mathbf{a_t}) - r'(\mathbf{s_t},\mathbf{a_t})} d\mathbf{a_t} \leq 1 \; \forall \mathbf{s_t} \in \mathcal{S}, r' \in R.$$

We define $R^*(R)$ as the set of reward functions satisfying this constraint w.r.t. reward functions in R:

$$R^*(R) \triangleq \left\{ r^* \;\middle|\; \int_{\mathcal{A}} e^{r^*(\mathbf{s_t},\mathbf{a_t}) - r'(\mathbf{s_t},\mathbf{a_t})} d\mathbf{a_t} \leq 1 \; \forall \mathbf{s_t} \in \mathcal{S}, r' \in R \right\}$$

Note that we can satisfy the constraint by making r^* arbitrarily negative, so the set $R^*(R)$ is non-empty. We now argue that all any applying MaxEnt RL to any reward function in $r^* \in R^*(R)$ lower bounds the robust-reward control objective.

Lemma D.2. *Let a set of reward functions R be given, and let $r^* \in R^*(R)$ be an arbitrary reward function belonging to the feasible set of MaxEnt reward functions. Then*

$$J_{\textit{MaxEnt}}(\pi, r^*) \leq \min_{r' \in R} \mathbb{E}_\pi \left[\sum_{t=1}^{T} r'(\mathbf{s_t}, \mathbf{a_t}) \right] \qquad \forall \pi \in \Pi.$$

Note that this bound holds for all feasible reward functions and all policies, so it also holds for the maximum r^*:

$$\max_{r^* \in R^*(R)} J_{\text{MaxEnt}}(\pi, r^*) \leq \min_{r' \in R} \mathbb{E}_\pi \left[\sum_{t=1}^{T} r'(\mathbf{s_t}, \mathbf{a_t}) \right] \qquad \forall \pi \in \Pi.$$

Defining $\pi^* = \arg\max_\pi J_{\text{MaxEnt}}(\pi, r^*)$, we get the following inequality:

$$\max_{r^* \in R^*(R), \pi \in \Pi} J_{\text{MaxEnt}}(\pi, r^*) \leq \min_{r' \in R} \mathbb{E}_{\pi^*} \left[\sum_{t=1}^{T} r'(\mathbf{s_t}, \mathbf{a_t}) \right] \leq \max_{\pi \in \Pi} \min_{r' \in R} \mathbb{E}_\pi \left[\sum_{t=1}^{T} r'(\mathbf{s_t}, \mathbf{a_t}) \right]. \quad (8)$$

Thus, we can find the tightest lower bound by finding the policy π and feasibly reward r^* that maximize Equation 8:

$$\begin{aligned} \max_{r,\pi} \quad & \mathbb{E}_\pi \left[\sum_{t=1}^{T} r(\mathbf{s_t}, \mathbf{a_t}) \right] + \mathcal{H}_\pi[\mathbf{a_t} \mid \mathbf{s_t}] \\ \text{s.t.} \quad & \int_{\mathcal{A}} e^{r(\mathbf{s_t},\mathbf{a_t}) - r'(\mathbf{s_t},\mathbf{a_t})} da \leq 1 \; \forall \; \mathbf{s_t} \in \mathcal{S}, r' \in R. \end{aligned} \quad (9)$$

It is useful to note that the constraints are simply LOGSUMEXP functions, which are convex. For continuous action spaces, we might approximate the constraint via sampling. Given a particular policy, the optimization problem w.r.t. r has a linear objective and convex constraint, so it can be solved extremely quickly using a convex optimization toolbox. Moreover, note that the problem can be solved independently for every state. The optimization problem is not necessarily convex in π.

D.8 ANOTHER COMPUTATIONAL EXPERIMENT

This section presents an experiment to study the approach outlined above. Of particular interest is whether the lower bound (Eq 8) comes close to the optimal minimax policy.

We will solve robust-reward control problems on 5-armed bandits, where the robust set is a collection of 5 reward functions, each is drawn from a zero-mean, unit-variance Gaussian. For each reward function, we add a constant to all of the rewards to make them all positive. Doing so guarantees that the optimal minimax reward is positive. Since different bandit problems have different optimal minimax rewards, we will normalize the minimax reward so the maximum possible value is 1.

Our approach, which we refer to as "LowerBound + MaxEnt", solves the optimization problem in Equation 9 by alternating between (1) solving a convex optimization problem to find the optimal reward function, and (2) computing the optimal MaxEnt RL policy for this reward function. Step 1 is done using CVXPY, while step 2 is done by exponentiating the reward function, and normalizing it to sum to one. Note that this approach is actually solving a harder problem: it is solving the

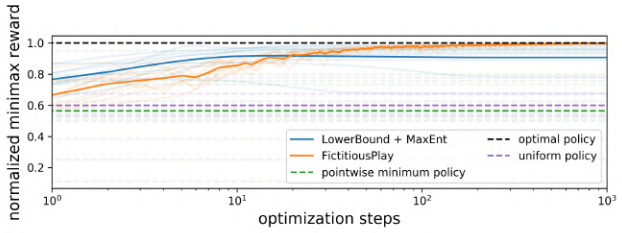


Figure 8: **Approximately solving an arbitrary robust-reward control problem.** In this experiment, we aim to solve the robust-reward control problem for an *arbitrary* set of reward functions. While we know that MaxEnt RL can be used to solve arbitrary robust-reward control problems exactly, doing so requires that we already know the optimal policy (§ D.6). Instead, we use the approach outlined in Sec. D.7, which allows us to *approximately* solve an arbitrary robust-reward control problem without knowing the solution apriori. This approach ("LowerBound + MaxEnt") achieves near-optimal minimax reward.

robust-reward control problem for a much larger set of reward functions that contains the original set of reward functions. Because this approach is solving a more challenging problem, we do not expect that it will achieve the optimal minimax reward. However, we emphasize that this approach may be easier to implement than fictitious play, which we compare against. Different from experiments in Section B, the "LowerBound + MaxEnt" approach assumes access to the full reward function, not just the rewards for the actions taken. For fair comparison, fictitious play will also use a policy player that has access to the reward function. Fictitious play is guaranteed to converge to the optimal minimax policy, so we assume that the minimax reward it converges to is optimal. We compare against two baselines. The "pointwise minimum policy" finds the optimal policy for a new reward function formed by taking the pointwise minimum of all reward functions: $\tilde{r}(\mathbf{a_t}) = \min_{r \in R} r(\mathbf{a_t})$. This strategy is quite simple and intuitive. The other baseline is a "uniform policy" that chooses actions uniformly at random.

We ran each method on the same set of 10 robust-reward control bandit problems. In Fig. 8, we plot the (normalized) minimax reward obtained by each method on each problem, as well as the average performance across all 10 problems. The "LowerBound + MaxEnt" approach converges to a normalized minimax reward of 0.91, close to the optimal value of 1. In contrast, the "pointwise minimum policy" and the "uniform policy" perform poorly, obtaining normalized minimax rewards of 0.56 and 0.60, respectively. In summary, while the method proposed for converting robust-reward control problems to MaxEnt RL problems does not converge to the optimal minimax policy, empirically it performs well.

E Experimental Details

E.1 Dynamics Robustness Experiments (Fig. 5)

We trained the MaxEnt RL policy using the SAC implementation from TF Agents (Guadarrama et al., 2018) with most of the default parameters (unless noted below).

Fig. 5a We used the standard `Pusher-v2` task from OpenAI Gym (Brockman et al., 2016). We used a fixed temperature of $1e-2$ for the MaxEnt RL results. For the standard RL results, we used the exact same codebase to avoid introducing any confounding factors, simply setting the temperature coefficient to a very small value $1e-5$. The obstacle is a series of three axis-aligned blocks with width 3cm, centered at (0.32, -0.2), (0.35, -0.23), and (0.38, -0.26). We chose these positions to be roughly along the perpendicular bisector of the line between the puck's initial position and the goal. We used 100 episodes of length 100 for evaluating each method. To decrease the variance in the results, we fixed the initial state of each episode:

$$\text{qpos} = [0., 0., 0., 0., 0., 0., 0., -0.3, -0.2, 0., 0.],$$
$$\text{qvel} = [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.].$$

Fig. 5b We used the `SawyerButtonPressEnv` environment from Metaworld (Yu et al., 2020), using a maximum episode length of 151. For this experiment, we perturbed the environment by modifying the observations such that the button appeared to be offset along the Y axis. We recorded the average performance over 10 episodes. For this environment we used a temperature of $1e1$ for MaxEnt RL and $1e-100$ for standard RL.[1]

Fig. 5c We used the standard `Pusher-v2` task from OpenAI Gym (Brockman et al., 2016). We modified the environment to perturb the XY position of the puck at time $t = 20$. We randomly sampled an angle $\theta \sim \text{Unif}[0, 2\pi]$ and displaced the puck in that direction by an amount given by the disturbance size. For evaluating the average reward of each policy on each disturbance size, we used 100 rollouts of length 151.

Fig. 6 We used a modified version of the 2D navigation task from Eysenbach et al. (2019) with the following reward function:

$$r(\mathbf{s_t}, \mathbf{a_t}) = \|\mathbf{s_t} - (8, 4)^T\|_2 - 10 \cdot \mathbb{1}(\mathbf{s_t} \in \mathcal{S}_{\text{obstacle}}).$$

Episodes were 48 steps long.

Fig. 2 We used the peg insertion environment from Eysenbach et al. (2017). Episodes were at most 200 steps long, but terminated as soon as the peg was in the hole. The agent receieved as reward of +100 once the peg was in the hole, in addition to the reward shaping terms described in Eysenbach et al. (2017).

E.2 Reward Robustness Experiments (Fig. 3)

Following Haarnoja et al. (2018a), we use a temperature of $\alpha = 1/5$. In Fig. 3, the thick line is the average over five random seeds (thin lines). Fig. 9 shows the evaluation of all methods on both the expected reward and the minimax reward objectives. Note that the minimax reward can be computed analytically as

$$\tilde{r}(\mathbf{s_t}, \mathbf{a_t}) = r(\mathbf{s_t}, \mathbf{a_t}) - \log \pi(\mathbf{a_t} \mid \mathbf{s_t}).$$

E.3 Bandits (Fig. 8)

The mean for arm i, μ_i, is drawn from a zero-mean, unit-variance Gaussian distribution, $\mu_i \sim \mathcal{N}(0, 1)$. When the agent pulls arm i, it observes a noisy reward $r_i \sim \mathcal{N}(\mu_i, 1)$. The thick line is the average over 10 random seeds (thin lines).

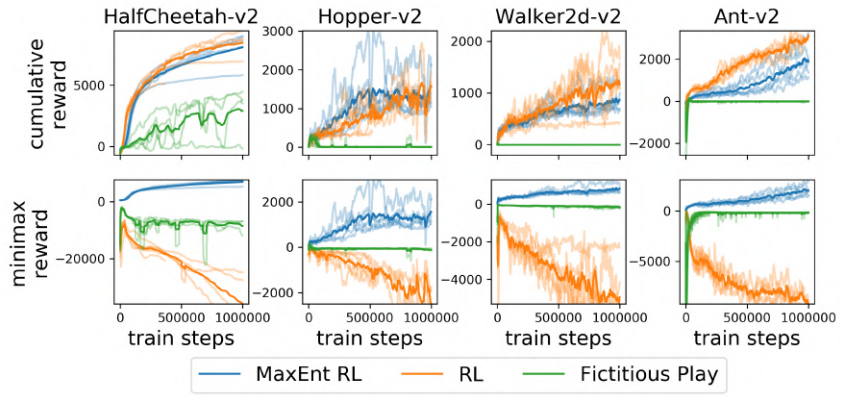


Figure 9: *(Top)* Both RL (SVG) and MaxEnt RL (SAC) effectively maximize expected reward. *(Bottom)* Only MaxEnt RL succeeds in maximizing the minimax reward.

[1] We used a larger value for the temperature for standard RL in the previous experiment to avoid numerical stability problems.