Rewriting History with Inverse RL: Hindsight Inference for Policy Improvement

Anonymous Authors[1]

Abstract

Multi-task reinforcement learning (RL) aims to simultaneously learn policies for solving many tasks. Several prior works have found that relabeling past experience with different reward functions can improve sample efficiency. Relabeling methods typically pose the question: if, in hindsight, we assume that our experience was optimal for some task, for what task was it optimal? Inverse RL answers this question. In this paper we show that inverse RL is a principled mechanism for reusing experience across tasks. We use this idea to generalize goal-relabeling techniques from prior work to arbitrary types of reward functions. Our experiments confirm that relabeling data using inverse RL outperforms prior relabeling methods on goal-reaching tasks, and accelerates learning on more general multi-task settings where prior methods are not applicable, such as domains with discrete sets of rewards and those with linear reward functions.

1. Introduction

Reinforcement learning (RL) aims to acquire control policies that take actions to maximize their reward, though existing RL algorithms remain data inefficient (Dubey et al., 2018; Kapturowski et al., 2018). Multi-task RL, where many RL problems are solved in parallel, has the potential to be more sample efficient than single-task RL, as data can be shared across tasks. Nonetheless, the problem of effectively sharing data across tasks remains largely unsolved.

The idea of sharing data across tasks has been studied at least since the 1990s (Caruana, 1997). More recently, a number of works have observed that retroactive relabeling of experience with different tasks can improve data efficiency (Kaelbling, 1993; Andrychowicz et al., 2017). A common theme in prior relabeling methods is to relabel past trials with whatever goal or task was performed successfully in that trial. For example, in a goal-reaching task, we might use the state actually reached at the end of the trajectory as the relabeled goal, since the trajectory corresponds to a successful trial *for the goal that was actually reached* (Andrychowicz et al., 2017; Pong et al., 2018). However, prior work on goal relabeling is inapplicable to more general reward functions, such as discrete sets of reward functions or tasks defined by various linear combinations of reward terms.

In this paper, we formalize prior relabeling techniques under the umbrella of *inverse* RL: by inferring the most likely task for a given trial via inverse RL, we provide a principled formula for relabeling in arbitrary multi-task problems. Inverse RL is *not* the same as evaluating a trajectory under all tasks and choosing whichever task yielded the highest reward. In fact, this strategy would often result in assigning most trajectories to the easiest task. Rather, inverse RL automatically takes into account the difficulty of each task by normalizing each reward function by the partition function. RL and inverse RL can be seen as complementary tools for maximizing reward: RL takes tasks and produces high-reward trajectories, and inverse RL takes trajectories and produces task labels such that the trajectories receive high reward. Formally, we prove that maximum entropy (MaxEnt) RL and MaxEnt inverse RL optimize the same multi-task objective: MaxEnt RL optimizes with respect to trajectories, while MaxEnt inverse RL optimizes with respect to tasks. Unlike prior goal-relabeling techniques, we can use inverse RL to relabel experience for arbitrary task distributions, including linear or discrete reward sets. This observation suggests that RL and inverse RL might be combined to efficiently solve many tasks simultaneously. The combination we develop, Hindsight Inference for Policy Improvement (HIPI), first relabels experience with inverse RL and then uses the relabeled experience to learn a *task-conditioned* policy (see Fig. 1). One variant of this framework follows the same design as prior value-based goal-relabeling methods (Kaelbling, 1993; Andrychowicz et al., 2017; Pong et al., 2018) but uses inverse RL to relabel experience, a difference that allows our method to handle arbitrary task families. The second variant has a similar design to self-imitation behavior cloning methods (Oh et al.,

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

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

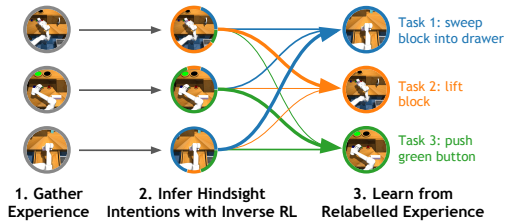

Figure 1. **Hindsight Inference for Policy Improvement (HIPI)**: Given a dataset of prior experience, we use inverse RL to infer the intentions of the agent's own past actions. We use the relabeled experience with any policy learning algorithm, such as off-policy RL or supervised learning.

2018; Ghosh et al., 2019; Savinov et al., 2018): we relabel past experience using inverse RL and then learn a policy via task-conditioned behavioral cloning. Both algorithms are probabilistic reinterpretations and generalizations of prior work.

The main contribution of our paper is the observation that inverse RL is a principled mechanism for reusing experience across tasks. This observation not only provides insight into success of prior relabeling methods, but it also provides guidance on applying relabeling to arbitrary multi-task RL problems. Our second contribution is two simple algorithms that use inverse RL-based relabeling to accelerate multi-task RL. These algorithms *do not* require expert demonstrations, but rather perform inverse RL on the agent's own (possibly-random) past experience. Our experiments on complex simulated locomotion and manipulation tasks show that our approach outperforms prior multi-task RL methods.

2. Related Work

We focus on multi-task RL problems, for which a number of algorithms have been proposed over the past decades (Thrun & Pratt, 2012; Hessel et al., 2019; Teh et al., 2017; Espeholt et al., 2018; Riedmiller et al., 2018). When applying off-policy RL in the multi-task setting, a common technique is to take data collected when performing task A and pretend that it was collected for task B by recomputing the rewards at each step, effectively inflating the amount of data available for learning (Kaelbling, 1993; Pong et al., 2018; Andrychowicz et al., 2017; Schaul et al., 2015). In this paper we will show that this relabeling can be understood as inverse RL.

While RL asks how to go from a reward function to a policy, inverse RL asks the opposite question: after observing an agent acting in an environment, can we infer which reward function the agent was trying to optimize? (Ziebart et al., 2008; Abbeel & Ng, 2004; Ratliff et al., 2006) Many MaxEnt inverse RL algorithms involve solving a MaxEnt RL problem in the inner loop. We propose the opposite, using MaxEnt *inverse* RL in the inner loop of MaxEnt RL.

Our work builds on the idea that MaxEnt (forward) RL can be viewed as probabilistic inference. This idea has been proposed in a number of prior works (Kappen et al., 2012; Toussaint, 2009; Todorov, 2008; 2007; Rawlik et al., 2013; Theodorou & Todorov, 2012; Levine, 2018) and used to construct practical RL algorithms (Haarnoja et al., 2017; 2018a; Abdolmaleki et al., 2018).

3. Hindsight Relabeling is Inverse RL

Notation. We will analyze an MDP with states s_t, actions a_t, and a reward function $r(s_t, a_t)$. We sample actions from a policy $q(a_t \mid s_t)$. The initial state is sampled $s_1 \sim p_1(s_1)$ and subsequent transitions are governed by a dynamics distribution $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$. We define a trajectory as a sequence of states and actions, $\tau = (s_1, a_1, \cdots)$, and write the likelihood of τ under policy q as

$$q(\tau) = p_1(s_1) \prod_t p(s_{t+1} \mid s_t, a_t) q(a_t \mid s_t).$$

In the multi-task setting, we use $\psi \in \Psi$ to identify each task and assume that the prior $p(\psi)$ over tasks is known. The set of tasks Ψ can be continuous or discrete, finite or infinite; each particular task $\psi \in \Psi$ can be continuous or discrete valued. We define $r_\psi(s_t, a_t)$ as the reward function for task ψ. Our experiments use both goal-reaching tasks, where ψ is a goal state and r_ψ is a distance metric, as well as more general task distributions, where ψ specifies the hyperparameters of the reward function. See Appendix A for a review of MaxEnt RL and MaxEnt IRL.

Off-policy RL allows us to use experience collected from one task to train the policy to solve another task. As one might expect, some trajectories are more useful for some tasks than others. Can we automatically determine for which tasks a given trajectory will be useful? Our analysis in the section says that the answer is to apply MaxEnt inverse RL.

We start by defining a multi-task version of the MaxEnt RL objective. A given prior over tasks, $p(\psi)$, combined with the target trajectory distribution, $p(\tau \mid \psi)$ (Eq. 6), yields the target joint distribution:

$$p(\tau, \psi) = p(\psi) \frac{1}{Z(\psi)} p_1(s_1) \prod_t p(s_{t+1} \mid s_t, a_t) e^{r_\psi(s_t, a_t)}. \tag{1}$$

We can express the multi-task (MaxEnt) RL objective as the reverse KL divergence between the joint trajectory-task distributions:

$$\max_{q(\tau,\psi)} -D_{\mathrm{KL}}(q(\tau, \psi) \parallel p(\tau, \psi)). \tag{2}$$

If we factor the joint distribution as $q(\tau, \psi) = q(\tau \mid \psi) p(\psi)$, Eq. 2 is equivalent to maximizing the expected (entropy-regularized) reward of a task-conditioned policy $q(\tau \mid \psi)$:

$$\mathbb{E}_{\substack{\psi \sim p(\psi) \\ \tau \sim q(\tau \mid \psi)}} \left[\left(\sum_t r_\psi(s_t, a_t) - \log q(a_t \mid s_t, \psi) \right) - \cancel{\log Z(\psi)} \right].$$

Algorithm 1 HIPI-RL:
Inverse RL for Off-Policy RL

while not converged **do**
$\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, \psi^{(i)})\} \sim$ ReplayBuffer
$\{\widetilde{\psi}^{(i)}\} \leftarrow$ InverseRL$(\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, \psi^{(i)})\})$
$\widetilde{Q} \leftarrow$ MaxEnt RL$(\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, \widetilde{\psi}^{(i)})\})$

Algorithm 2 HIPI-BC:
Inverse RL for Behavior Cloning

while not converged **do**
$\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, \psi^{(i)})\} \sim$ ReplayBuffer
$\{\widetilde{\psi}^{(i)}\} \leftarrow$ InverseRL$(\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, \psi^{(i)})\})$
$\theta \leftarrow \theta + \eta \nabla_\theta \sum_i \log \pi_\theta \left(a_t^{(i)} \mid s_t^{(i)}, \widetilde{\psi}^{(i)}\right)$
return π_θ

Since we want to figure out for which tasks a given trajectory will be useful, we instead choose to factor $q(\tau, \psi) = q(\psi \mid \tau) q(\tau)$, where $q(\tau)$ is a distribution over previously-observed trajectories. The *relabeling distribution* $q(\psi \mid \tau)$ tells us the tasks for which a given trajectory τ behaved similar to the optimal policy. We therefore expect that τ will be most useful for learning tasks ψ where $q(\psi \mid \tau)$ is large. We can now compute the relabeling distribution $q(\psi \mid \tau)$ which optimizes our objective:

$$q(\psi \mid \tau) \propto p(\psi) \exp\left(\sum_t r_\psi(s_t, a_t) - \log Z(\psi)\right). \quad (3)$$

Intuitively, we should use trajectory τ for solving tasks where the trajectory receives the highest reward, *as compared with the partition function*. The key observation here is that *the optimal relabeling distribution corresponds exactly to the MaxEnt inverse RL posterior over tasks* (Eq. 7). Appendix C shows how to perform relabeling when given a transition rather than an entire trajectory:

3.1. Special Case: Goal Relabeling

A number of prior works have explicitly (Kaelbling, 1993; Andrychowicz et al., 2017; Pong et al., 2018) and implicitly (Savinov et al., 2018; Lynch et al., 2019; Ghosh et al., 2019) found that hindsight relabeling can accelerate learning for *goal-reaching* tasks, where tasks ψ correspond to goal states. We will show that these prior methods are a special case of inverse RL. Define a goal-conditioned reward function that penalizes the agent for failing to reaching the goal at the last step:

$$r_\psi(s_t, a_t) = \begin{cases} -\infty & \text{if } t = T \text{ and } s_t \neq \psi \\ 0 & \text{otherwise} \end{cases}. \quad (4)$$

With this reward function, the optimal relabeling distribution $q(\psi \mid \tau)$ from Eq. 3 is simply $q(\psi \mid \tau) = \mathbb{1}(\psi = s_T)$, where s_T is the final state in trajectory τ. Thus, *relabeling with the state actually reached is equivalent inverse RL when using the reward function in Eq. 4.* While inverse RL is convenient when using this reward function, we rarely care about a policy's expected reward under this particular reward function. Viewing goal relabeling as a special case of inverse RL lets us to extend goal relabeling to arbitrary task distributions. We will show that inverse RL handles task distributions including goal-reaching, discrete sets of tasks, and linear reward functions. We provide additional analysis of the benefit of relabeling in Appendix B.

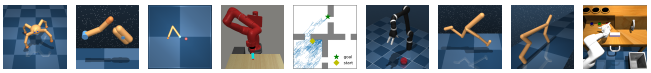

Figure 2. **Environments for experiments**: *(a)* quadruped, *(b)* finger, *(c)* 2D reacher, *(d)* sawyer reach, *(e)* 2D navigation *(f)* jaco reach, *(g)* walker, *(h)* cheetah, and *(i)* desk manipulation.

4. Using Inverse RL to Accelerate RL

Propose two multi-task policy search algorithms, Hindsight Inference for Policy Improvement (HIPI), that use inverse RL to relabel experience. Before introducing the algorithms themselves, we give an overview of the dataflow. Inverse RL (Alg. 3) takes as input transitions or trajectories and returns a distribution over tasks, ψ. Our policy search algorithms (Algorithms 1 and 2) take past experience, use inverse RL to sample a corresponding task, and then update the policy using experience together with the task label. The two algorithms differ in how they update the policy: HIPI-RL will use off-policy MaxEnt RL, whereas HIPI-BC will use behavior cloning.

5. Experiments

Our experiments focus on two methods for using relabeled data: off-policy RL (Alg. 1) and behavior cloning (Alg. 2). We evaluate our method on both goal-reaching tasks as well as more general task distributions, including linear combinations of a reward basis and discrete sets of tasks. The aim of all experiments is to maximize the task reward, not to imitate an expert.

5.1. HIPI-RL: Inverse RL for Off-Policy RL

Our first set of experiments apply Alg. 1 to domains with varying reward structure, demonstrating how relabeling data with inverse RL can accelerate off-policy RL.

Goal-reaching task distributions. We apply our method to goal-reaching tasks, where each task ψ corresponds to reaching a different goal state. We used six domains: a quadruped locomotion task, a robotic finger turning a knob, a 2D reacher, a reaching task on the Sawyer robot, a 2D navigation environment with obstacles, and a reaching task on the Jaco robot. Appendix E provides details of all tasks. We compared our method against four alter-

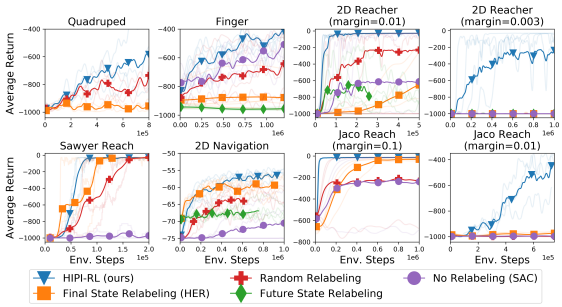

Figure 3. **Relabeling for goals-reaching tasks**: On six goal-reaching domains, relabeling with inverse RL (our method) learns faster than with previous relabeling strategies. On extremely sparse versions of two tasks, shown in the right column, only our method learns the tasks. See text for details.

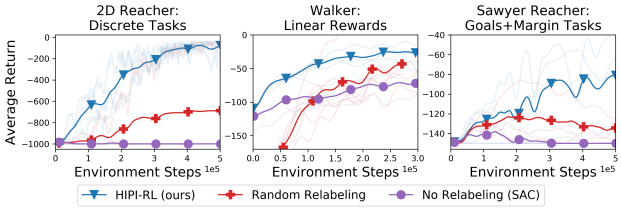

Figure 4. **Relabeling for general tasks distributions**: On all tasks, relabeling with inverse RL accelerates learning and leads to larger asymptotic reward. Existing relabeling strategies are not applicable in this setting.

native relabeling strategies: relabeling with the final state reached (HER (Andrychowicz et al., 2017)), relabeling with a randomly-sampled task, relabeling with a future state in the same trajectory, and doing no relabeling (SAC (Haarnoja et al., 2018a)). For tasks where the goal state only specifies certain dimensions of the state, relabeling with the final state and future state requires privileged information indicating which state dimensions correspond to the goal.

As shown in Fig. 3, relabeling experience with inverse RL (our method) always learns at least as quickly as the other relabeling strategies, and often achieves larger asymptotic reward. While final state relabeling (HER) performs well on some tasks, it is worse than random relabeling on other tasks. We also observe that random relabeling is a competitive baseline, provided that the number of gradient steps is sufficiently tuned. We conjectured that soft relabeling would be most beneficial in settings with extremely sparse rewards. To test this hypothesis, we modified the reward functions in 2D reacher and Jaco reaching environments to be much sparser. As shown in the far right column on Fig. 3, only HIPI-RL solves these tasks.

More general task distributions. Our next experiment demonstrates that, in addition to relabeling goals, inverse RL can also relabel experience for more general tasks distributions. Our first task distribution is a discrete set of goal states $\psi \in \{1, \cdots, 32\}$ for the 2D reacher environment. The second task distribution highlights the capability of inverse RL to relabel experience for classes of reward functions defined as linear combinations $r_\psi(s,a) = \sum_{i=1}^{d} \psi_i \phi_i(s,a)$ of features $\phi(s,a) \in \mathbb{R}^d$. We use the walker environment, with features corresponding to torso height, velocity, relative position of the feet, and a control cost. The third task distribution is again a goal reaching task, but one where the task $\phi = (s_g, m)$ indicates both the goal state as well as the desired margin from that goal state. As prior relabeling approaches are not applicable to these general task distributions, we only compared our approach to random relabeling and no relabeling. As shown in Fig. 4, relabeling with inverse RL provides more sample efficient learning in all tasks, and the asymptotic reward is larger than the baselines by a non-trivial amount in two of the three tasks.

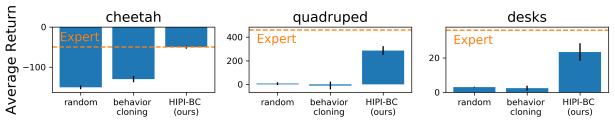

Figure 5. **HIPI-BC**: Behavior cloning on experience relabeled with inverse RL boosts reward on goal-reaching tasks, linear reward functions, and discrete tasks.

5.2. HIPI-BC: Behavioral Cloning on Experience Relabeled with Inverse RL

We now present experiments using HIPI-BC (Alg. 2), performing behavior cloning on the experience relabeled with inverse RL. We use three domains with varying types of rewards: (1) half-cheetah with continuous goal velocities; (2) quadruped with linear reward functions; and (3) manipulation with nine discrete tasks. For the half-cheetah and quadruped domains, we collected 1000 demonstrations from a policy trained with off-policy RL. For the manipulation environment, Lynch et al. (2019) provided a dataset of 100 demonstrations for each of these tasks, which we aggregate into a dataset of 900 demonstrations. In all settings, we discarded the task labels, simulating the common real-world setting where experience does not come prepared with task labels. As shown in Fig. 5, first inferring the tasks with inverse RL and then performing behavioral cloning results in significantly higher final rewards than task-agnostic behavior cloning on the entire dataset.

6. Discussion

In this paper, we proved that inverse RL is as a principled mechanism for sharing experience across tasks. We showed that a number of prior works can be understood as special cases of this framework. The idea that inverse RL might be used to relabel data is powerful because it enables us to extend relabeling techniques to general classes of reward functions. We used this idea to propose two multi-task policy search algorithms, which relabel past experience with inverse RL and use this relabeled experience for off-policy RL and supervised learning.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Caruana, R. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Cover, T. M. and Thomas, J. A. Elements of information theory (wiley series in telecommunications and signal processing), 2006.

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., and Efros, A. A. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217*, 2018.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Eysenbach, B., Salakhutdinov, R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 15220–15231, 2019.

Ghosh, D., Gupta, A., Fu, J., Reddy, A., Devine, C., Eysenbach, B., and Levine, S. Learning to reach goals without reinforcement learning. *arXiv preprint arXiv:1912.06088*, 2019.

Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Harris, C., Vanhoucke, V., et al. Tf-agents: A library for reinforcement learning in tensorflow, 2018.

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1352–1361. JMLR. org, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Hessel, M., Soyer, H., Espeholt, L., Czarnecki, W., Schmitt, S., and van Hasselt, H. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

Kaelbling, L. P. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

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

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. 2018.

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

Liang, C., Berant, J., Le, Q., Forbus, K. D., and Lao, N. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*, 2016.

Lynch, C., Khansari, M., Xiao, T., Kumar, V., Tompson, J., Levine, S., and Sermanet, P. Learning latent plans from play. *arXiv preprint arXiv:1903.01973*, 2019.

Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750. ACM, 2007.

Pong, V., Gu, S., Dalal, M., and Levine, S. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient off-policy meta-reinforcement learning via probabilistic context variables. *arXiv preprint arXiv:1903.08254*, 2019.

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736. ACM, 2006.

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

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Van de Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

Savinov, N., Dosovitskiy, A., and Koltun, V. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind control suite. Technical report, DeepMind, January 2018. URL https://arxiv.org/abs/1801.00690.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multi-task reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Theodorou, E. A. and Todorov, E. 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.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

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

Todorov, E. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pp. 4286–4292. IEEE, 2008.

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

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019.

Ziebart, B. D. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. PhD thesis, figshare, 2010.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. 2008.

A. Preliminaries

MaxEnt RL. MaxEnt RL maximizes the entropy-regularized sum of rewards (Ziebart, 2010). MaxEnt RL can be equivalently expressed as a distribution matching problem. First, construct a reward-based *target distribution* $p(\tau)$ over trajectories,

$$p(\tau) \triangleq \frac{1}{Z} p_1(s_1) \prod_t p(s_{t+1} \mid s_t, a_t) e^{r(s_t, a_t)}. \tag{5}$$

Then, minimizing the reverse KL between the policy's trajectory distribution $q(\tau)$ and the target trajectory distribution $p(\tau)$ is equivalent to maximizing the entropy-regularized sum of rewards:

$$\max_q -D_{\mathrm{KL}}(q(\tau) \parallel p(\tau)) = \mathbb{E}_q \left[\left(\sum_t r(s_t, a_t) - \log q(a_t \mid s_t) \right) - \log Z \right],$$

The partition function Z is introduced to make $p(\tau)$ integrate to one. Although the partition function is independent of the policy, and prior RL algorithms have therefore ignored it, it will play a crucial role in relabeling experience. Section 3 will examine the multi-task version of this objective, where rewards will depend on the task ψ.

MaxEnt inverse RL. Inverse RL observes previously-collected data and attempts to infer which reward function r_ψ that actor was trying to maximize. While many inverse RL algorithms also yield a policy which is optimal for the inferred reward function, we will use "inverse RL" to refer to just the problem of inferring the reward. MaxEnt inverse RL (Ziebart et al., 2008) is a variant of inverse RL that defines the probability of trajectory τ being produced for task ψ as

$$p(\tau \mid \psi) = \frac{p_1(s_1) \prod_t p(s_{t+1} \mid s_t, a_t) e^{r_\psi(s_t, a_t)}}{Z(\psi)}, \quad \text{where} \quad Z(\psi) \triangleq \int p_1(s_1) \prod_t p(s_{t+1} \mid s_t, a_t) e^{r_\psi(s_t, a_t)} d\tau. \tag{6}$$

Applying Bayes' rule, the posterior distribution over tasks ψ is given as follows:

$$p(\psi \mid \tau) = \frac{p(\tau \mid \psi) p(\psi)}{p(\tau)} \propto p(\psi) e^{\sum_t r_\psi(s_t, a_t) - \log Z(\psi)}. \tag{7}$$

B. How Much Does Relabeling Help?

Up to now, we have shown that the optimal way to relabel data is via inverse RL. We now obtain a lower bound on the improvement from relabeling. Both lemmas in this section will assume that a joint distribution $q(\tau, \psi)$ over tasks and trajectories be given (e.g., specified by a policy $q(a_t \mid s_t, \psi)$). We will define $q_\tau(\tau) = \int q(\tau, \psi)$ as the marginal distribution over trajectories and $q_\tau(\psi \mid \tau)$ as the corresponding optimal relabeling distribution (Eq. 3). We then construct the joint distirbution $q_\tau(\tau, \psi) = q_\tau(\psi \mid \tau) q_\tau(\tau)$ using the optimal relabeling distribution $q_\tau(\psi \mid \tau)$. We first show that relabeling data using inverse RL improves the MaxEnt RL objective:

Lemma 1. *The relabeled distribution $q_\tau(\tau, \psi)$ is closer to the target distribution than the original distribution, as measured by the KL divergence:*

$$D_{\mathrm{KL}}(q_\tau(\tau, \psi) \parallel p(\tau, \psi)) \leq D_{\mathrm{KL}}(q(\tau, \psi) \parallel p(\tau, \psi)).$$

Proof. Of the many possible relabeling distributions, one choice is to do no relabeling, assigning to each trajectory τ the task ψ that was commanded when the trajectory was collected. Denote this relabeling distribution $q_0(\psi \mid \tau)$, so $q_0(\psi \mid \tau) q_\tau(\tau) = q(\tau, \psi)$. Because $q_\tau(\psi \mid \tau)$ minimizes the KL among all relabeling distributions (including $q_0(\psi \mid \tau)$), the desired inequality holds:

$$D_{\mathrm{KL}}(q_\tau(\psi \mid \tau) q_\tau(\tau) \parallel p(\tau, \psi)) \leq D_{\mathrm{KL}}(q_0(\psi \mid \tau) q_\tau(\tau) \parallel p(\tau, \psi)). \qquad \square$$

Thus, the relabeled data is an improvement over the original data, achieving a larger entropy-regularized reward (Eq. 2). As our experiments will confirm, relabeling data will accelerate learning. Our next result will give us a lower bound on the size of this improvement:

Lemma 2. *The improvement in the MaxEnt RL objective (Eq. 2) gained by relabeling is lower bounded as follows:*

$$D_{\mathrm{KL}}(q(\tau, \psi) \parallel p(\tau, \psi)) - D_{\mathrm{KL}}(q_\tau(\tau, \psi) \parallel p(\tau, \psi)) \geq \mathbb{E}_{q_\tau} \left[D_{\mathrm{KL}}(q(\psi \mid \tau) \parallel q_\tau(\psi \mid \tau)) \right].$$

Proof. The optimal relabeling distribution can be viewed as an information projection of the joint distribution $q_\tau(\psi \mid \tau) q_\tau(\tau)$ onto the target distribution $p(\tau, \psi)$ (Eq. 1):

$$q_\tau(\psi \mid \tau) q_\tau(\tau) = \min_{q_\tau \in \mathcal{Q}_\tau} D_{\mathrm{KL}}(q_\tau(\psi \mid \tau) q_\tau(\tau, \psi) \parallel p(\tau, \psi)),$$

where $\mathcal{Q} = \{q(\tau, \psi) \text{ s.t. } \int q(\tau, \psi) d\psi = q_\tau(\tau)\}$ is the set of all joint distributions with marginal $q_\tau(\tau)$. Note that this set $\mathcal{Q}$ is closed and convex. We then apply Theorem 11.6.1 from Cover & Thomas (2006):

$$D_{\mathrm{KL}}(q(\tau, \psi) \parallel p(\tau, \psi)) \geq D_{\mathrm{KL}}(q_\tau(\tau, \psi) \parallel p(\tau, \psi)) + D_{\mathrm{KL}}(q(\tau, \psi) \parallel q_\tau(\tau, \psi)). \tag{8}$$

The second KL divergence on the RHS can be simplified:

$$\begin{aligned} D_{\mathrm{KL}}(q(\tau, \psi) \parallel q_\tau(\tau, \psi)) &= D_{\mathrm{KL}}(q(\psi \mid \tau) q_\tau(\tau) \parallel q_\tau(\psi \mid \tau) q_\tau(\tau)) \\ &= \cancel{D_{\mathrm{KL}}(q_\tau(\tau) \parallel q_\tau(\tau))} + \mathbb{E}_{q_\tau}\left[D_{\mathrm{KL}}(q(\psi \mid \tau) \parallel q_\tau(\psi \mid \tau))\right] \end{aligned}$$

Substituting this simplification into Eq. 8 and rearranging terms, we obtain the desired result. □

This result says that the amount that relabeling helps is at least as large as the difference between the original task labels $q(\psi \mid \tau)$ and the task labels inferred by inverse RL, $q_\tau(\psi \mid \tau)$. Experience from the optimal policy is already optimally labeled, so the improvement from relabeling drops to zero when we acquire the optimal policy.

C. Inverse RL on Transitions

For simplicity, our derivation of relabeling in Section 3 assumed that entire trajectories were provided. This section outlines how to do relabeling with inverse RL when we are only provided with (s, a, s') transitions, rather than entire trajectories. In this case, policy distribution q in the MaxEnt RL objective (Eq. 2) is conditioned on the current state and action (s_t, a_t) in addition to the task ψ:

$$\max_{q(\tau, \psi \mid s_t, a_t)} -D_{\mathrm{KL}}(q(\tau, \psi \mid s_t, a_t) \parallel p(\tau, \psi)). \tag{9}$$

Following the derivation in Section 3, we expand this objective, using $q(\psi \mid s_t, a_t)$ as our relabeling distribution:

$$\begin{aligned} \mathbb{E}_{\substack{\psi \sim q(\psi \mid s_t, a_t) \\ \tau \sim q(\tau \mid \psi, s_t, a_t)}} \Bigg[\sum_{t'=t} r_\psi(s_{t'}, a_{t'}) + \cancel{\log p(s_{t'+1} \mid s_{t'}, a_{t'})} - \log q(a_{t'} \mid s_{t'}, \psi) - \cancel{\log p(s_{t'+1} \mid s_{t'}, a_{t'})} \\ + p(\psi) - \log q(\psi \mid s_t, a_t) - \log Z(\psi) \Bigg]. \end{aligned} \tag{10}$$

The expected value of the two summations is the *soft* Q-function for the policy $q(a \mid s, \psi)$:

$$\widetilde{Q}^q(s_t, a_t, \psi) = \mathbb{E}_{\substack{\psi \sim q(\psi \mid s_t, a_t) \\ \tau \sim q(\tau \mid \psi, s_t, a_t)}} \left[\sum_{t'=t} r_\psi(s_{t'}, a_{t'}) - \log q(a_{t'} \mid s_{t'}, \psi) \right]. \tag{11}$$

Substituting Eq. 11 into Eq. 10 and ignoring terms that do not depend on ψ, we can solve the optimal relabeling distribution:

$$q(\psi \mid s_t, a_t) \propto p(\psi) e^{\widetilde{Q}^q(s_t, a_t, \psi) - \log Z(\psi)}. \tag{12}$$

D. Prior Methods are Special Cases of HIPI

Prior work on both goal-conditioned supervised learning, self-imitation learning, and reward-weighted regression can all be understood as special cases of HIPI-BC. Goal-conditioned supervised learning (Savinov et al., 2018; Ghosh et al., 2019; Lynch et al., 2019) learns a goal-conditioned policy using a dataset of past experience. For a given state, the action that was actually taken is treated as the correct action (i.e., label) for states reached in the future, and a policy is learned via supervised learning. As discussed in Section 3.1, relabeling with the goal actually achieved is a special case of our framework. We refer the reader to those papers for additional evidence that combining inverse RL (albeit a trivial special case) with behavior cloning can effectively learn complex control policies. Self-imitation learning (Oh et al., 2018) and

iterative maximum likelihood training (Liang et al., 2016) augment RL with supervised learning on a handful of the best previously-seen trajectories, an approach that can be viewed in the inverse RL followed by supervised learning framework. However, because the connection to inverse RL is not made precise, these methods omit the partition function, which may prove problematic when extending these methods to multi-task settings. Finally, single-task RL methods based on variational policy search (Levine, 2018) and reward-weighted regression (Peters & Schaal, 2007; Peng et al., 2019) can also be viewed in this framework. Noting that the optimal relabeling distribution is given as $q(\psi \mid \tau) \propto \exp(R_\psi(\tau) - \log Z(\psi))$, relabeling by sampling from the inverse RL posterior and then performing behavior cloning can be written concisely as the following objective:

$$\int e^{R_\psi(\tau) - \log Z(\psi)} \sum_t \log \pi(a_t \mid s_t, \psi) d\psi d\tau.$$

The key difference between this objective and prior work is the partition function. The observation that these prior methods are special cases of inverse RL allows us to apply similar ideas to arbitrary classes of reward functions, a capability we showcase in our experiments.

D.1. The Importance of the Partition Function

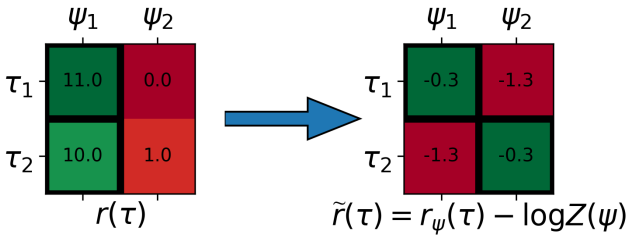

Figure 6. **The partition function normalizes rewards of different scales**: Two trajectories are evaluated on tasks with different reward scales. Black borders indicate the task to which we assign each trajectory. *(Left)* Without normalization, both trajectories are assigned to task ψ_1. *(Right)* After normalizing with the partition function, as is done by inverse RL (our method), trajectory τ_1 is assigned task ψ_1 and τ_2 is assigned to ψ_2.

The partition function used by inverse RL is important for hindsight relabeling as it normalizes the rewards from tasks with varying difficulty and reward scale. Fig. 6 shows a didactic example with two tasks, where the rewards for one task are larger than the rewards for the other task. Relabeling with the reward under which the agent received the largest reward (akin to Andrychowicz et al. (2017)) fails, because all experience will be relabeled with the first (easier) task. Subtracting the partition function from the rewards (as in Eq. 3) results in the desired behavior: trajectory τ_1 is assigned task ψ_1 and τ_2 is assigned to ψ_2.

D.2. HIPI-BC Ablation without Partition Function

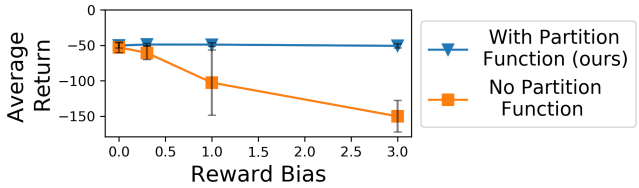

Figure 7. Removing the partition function from inverse RL results in poor performance.

Our final experiment demonstrates the importance of the partition function. On the cheetah domain, we synthetically corrupt the demonstrations by adding a constant bias to the reward for the first task (whichever velocity was sampled first). We then compare the performance of our approach against an ablation that did not normalize by the partition function when relabeling data. As shown in Fig. 7, the performance of this ablation degrades as the reward bias increases, whereas our method, which normalizes the task rewards in the inverse RL step, is not affected.

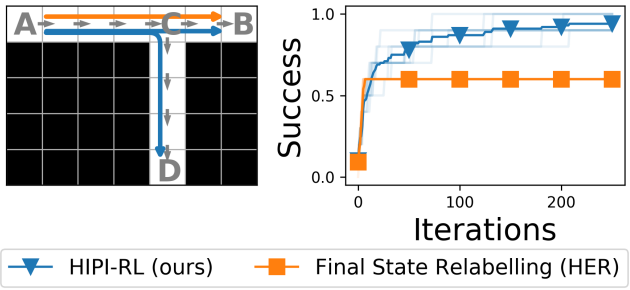

Figure 8. **Relabeling stitches crossing trajectories**: *(Left)* A gridworld with two observed trajectories $A \to B$ and $C \to D$. Inverse RL identifies both B and D as likely intentions from state A and includes both $A \to B$ and $A \to D$ in the relabeled data. Final state relabeling (HER) only relabels with the goal actually achieved, corresponding to trajectory $A \to B$. *(Right)* HIPI-RL learns to reach all goals, whereas HER only learns to reach the 6/10 goals along the top row.

E. Experimental Details

E.1. Approximate Inverse RL

Algorithm 3 Approximate Inverse RL.

When used in HIPI-RL (Alg. 1) we only have transitions, so we compute $R^{(i)}_{\psi^{(j)}}$ using Eq. 12 (blue line). When used in HIPI-BC (Alg. 2) we have full trajectories, so we compute $R^{(i)}_{\psi^{(j)}}$ using Eq. 3 (red line).

function INVERSERL($\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}, \psi^{(i)}\})$
  for $j = 1, \cdots, B$ **do** ▷ task index
    for $i = 1, \cdots, B$ **do** ▷ state-action index
      $R^{(i)}_{\psi^{(j)}} \leftarrow \widetilde{Q}(s^{(i)}, a^{(i)}, \psi^{(j)})$ ▷ Eq. 12
      $R^{(i)}_{\psi^{(j)}} \leftarrow \sum_t r_{\psi^{(j)}}(s_t^{(i)}, a_t^{(i)})$ ▷ Eq. 3
    $\log Z(\psi^{(j)}) \leftarrow \frac{1}{B} \sum_{i=1}^{B} e^{R^{(i)}_{\psi^{(j)}}}$
  for $i = 1, \cdots, B$ **do**
    $\widetilde{\psi}^{(i)} \sim$ SOFTMAX$(R^{(i)}_{\psi^{(1)}} - \log Z(\psi^{(1)}), \cdots)$
  return $\{\widetilde{\psi}^{(i)}\}$

The key design decision is the choice of inverse RL algorithm. We will now outline one approximate inverse RL algorithm that can be efficiently integrated into off-policy RL, providing pseudo-code in Alg. 3. The algorithm takes as input B transitions $\{(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)})\}$ along with the tasks that were commanded when these transitions were collected, $\{\psi^{(i)}\}$. Rather than consider all (possibly-infinite) tasks, we make a non-parametric approximation and only consider the likelihood of these B originally-commanded tasks. Then, for all pairs $1 \le i, j \le B$, we compute the likelihood that transition $(s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)})$ is optimal for task $\psi^{(j)}$ following Eq. 12. Finally, we normalize these likelihoods by taking a softmax, which is equivalent to using the following (biased) approximation of the partition function $\log Z(\psi)$:

$$\log Z(\psi) = \log \int e^{R_\psi(s,a)} ds da \approx \log \frac{1}{B} \sum_{i=1}^{B} e^{R_\psi(s^{(i)}, a^{(i)})}.$$

Alg. 3 is one of many possible methods for inverse RL. Alternative methods include gradient-based optimization over all task labels in the replay buffer and learning a parametric task-sampler to approximate the optimal relabeling distribution (Eq. 3). We leave this as future work.

F. Didactic example.

We start with a didactic example to motivate why relabeling experience with inverse RL should accelerate off-policy RL. In the gridworld shown in Fig. 8, we construct a dataset with two trajectories: $A \to B$ and $C \to D$. From state A, inverse RL identifies many possible intentions, including states B and D, so both $A \to B$ and $A \to D$ get included in the relabeled

data. In contrast, final state relabeling (HER) only uses trajectory $A \to B$. We then apply (soft) Q-learning to both datasets. Whereas Q-learning with final state relabeling only succeeds at reaching those goals in the top row (6/10 goals), our approach, which corresponds to Q-learning with inverse RL, relabeling succeeds at reaching all goals. Note that relabeling with future states would also fail to use states from trajectory $C \to D$ as goals state A. The remainder of this section will show the benefits of relabeling using inverse RL in domains of increasing complexity.

F.1. Hyperparameters for Off-Policy RL

Except for the didactic experiment, we used SAC (Haarnoja et al., 2018a) as our RL algorithm, taking the implementation from Guadarrama et al. (2018). This implementation scales the critic loss by a factor of 0.5. Following prior work (Pong et al., 2018), we only relabeled 50% of the samples drawn from the replay buffer, using the originally-commanded task the remaining 50%. The only hyperparameter that differed across relabeling strategies was the number of gradient updates per environment step. For each experiment, we evaluated each method with values in $\{1, 3, 10, 30\}$ and reported the results of the best hyperparameter in our plots. Perhaps surprisingly, doing random relabeling but simply increasing the number of gradient updates per environment step was a remarkably competitive baseline.

- Learning Rate: 3e-4 (same for actor, critic, and entropy dual parameter)
- Batch Size: 32
- Network architecture: The input was the concatenation of the state observation and the task ψ. Both the actor and critic networks were 2 hidden layer ReLu networks. The actor output was squashed by a tanh activation to lie within the actor space constraints. There was no activation at the final layer of the critic network, except in the desk environment (see comment below). The hidden layer dimensions were (32, 32) for the 2D navigation environments, (256, 256) for the quadruped and desk environments, and (64, 64) for all other environments.
- Discount γ: 0.99
- Initial data collection steps: 1e5
- Target network update period: 1
- Target network τ: 0.005
- Entropy coefficient α: We used the entropy-constrained version of SAC (Haarnoja et al., 2018b), using $-\dim(\mathcal{A})$ as the target value, where $\dim(\mathcal{A})$ is the action space dimension.
- Replay buffer capacity: 1e6
- Optimizer: Adam
- Gradient Clipping: We found that clipping the gradients to have unit norm was important to get RL working on the Sawyer and Jaco tasks.

To implement final state relabeling, we modified transitions as they were being added to the replay buffer, adding both the original transition and the transition augmented to use the final state as the goal. To implement future state relabeling, we modified transitions as they were being added to the replay buffer, adding both the original transition and a transition augmented to use one of the next 4 states in the same trajectory as the goal.

F.2. Hyperparameters for Behavior Cloning Experiments

To account for randomness in the learning process, we collect at least 200 evaluation episodes per domain; we repeat this experiment for at least 5 random seeds on each domain, and plot the mean and standard deviation over the random seeds. We used a 2-layer neural network with ReLu activations for all experiments. The hidden layers had size (256, 256). We optimized the network to minimize MSE using the Adam optimizer with a learning rate of 3e-4. We used early stopping, halting training when the validation loss increased for 3 consecutive epochs. Typically training converged in 30 - 50 epochs. We normalized both the states and actions. For the task-conditioned experiments, we concatenated the task vectors to the state vectors.

F.3. Quadruped Environment

The quadruped was a modified version of the environment from Abdolmaleki et al. (2018). We modified the initial state distribution so the agent always started upright, and modified the observation space to include the termination signal as part of the observation. Tasks $\psi \in \mathbb{R}^2$ were sampled uniformly from the unit circle. Let $s_{\text{XY vel}}$ and $s_{\text{XY pos}}$ indicate the XY velocity and position of the agent. For the HIPI-RL experiments, we used the following sparse reward function:

$$r_\psi(s, a) = \mathbb{1}(\|s_{\text{XY pos}} - \psi\|_2 \leq 0.3) - 1.0,$$

and the episode terminated when $\|s_{\text{XY pos}} - \psi\|_2 \leq 0.3$. We also reset the environment after 300 steps if the agent had failed to reach the goal. For the HIPI-BC experiments, we used the following dense reward function:

$$r_\psi(s, a) = s_{\text{XY vel}}^T \psi + 0.1\|a\|_2^2.$$

Episodes were 300 steps long.

Figure 9. Quadruped

F.4. Finger Environment

The finger environment was taken from Tassa et al. (2018). Tasks ψ were sampled using the environment's default goal sampling function. Let s_{XY} denote the XY position of the knob that the agent can manipulate. The reward function was defined as

$$r_\psi(s, a) = \mathbb{1}(\|s_{\text{XY}} - \psi\|_2 \leq 0.01) - 1.0$$

and the episode terminated when $\|s_{\text{XY}} - \psi\|_2 \leq 0.01$. We also reset the environment after 1000 steps if the agent had failed to reach the goal.

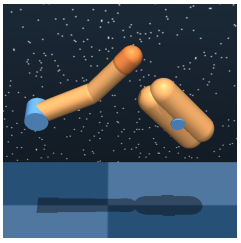

Figure 10. Finger

F.5. 2D Reacher Environment

The 2D reacher environment was taken from Tassa et al. (2018). Let s_{XY} denote the XY position of the robot end effector. The reward function was defined as

$$r_\psi(s, a) = \mathbb{1}(\|s_{\text{XY}} - \psi\|_2 \leq m) - 1.0$$

and the episode terminated when $\|s_{\text{XY}} - \psi\|_2 \leq m$, where $m > 0$ is a margin around the goal. We used $m = 0.01$ and $m = 0.003$ in our experiments. We also reset the environment after 1000 steps if the agent had failed to reach the goal. Tasks were sampled using the environment's default goal sampling function.

Figure 11. 2D Reacher

F.6. Sawyer Reach Environment

The sawyer reach environment was taken from Yu et al. (2019). Let s_{XYZ} denote the XYZ position of the robot end effector. The reward function was defined as

$$r_\psi(s, a) = \mathbb{1}(\|s_{\text{XYZ}} - \psi\|_2 \leq m) - 1.0$$

and the episode terminated when $\|s_{\text{XY}} - \psi\|_2 \leq m$, where $m > 0$ is a margin around the goal. We used $m = 0.01$ and $m = 0.003$ in our experiments. We also reset the environment after 150 steps if the agent had failed to reach the goal. Tasks were sampled using the environment's default goal sampling function. For the experiment where the task indicator ψ also specified the margin m, the margin was sampled uniformly from the interval $[0, 0.1]$.

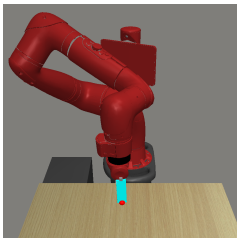

Figure 12. Sawyer Reach

F.7. 2D Navigation Environment

We used the 2D navigation environment from Eysenbach et al. (2019). The action space is continuous and indicates the desired change of position. The dynamics are stochastic, and the initial state and goal are sampled uniformly at random for each episode. To increase the difficulties of credit assignment and exploration, the agent is always initialized in the lower left corner, and we randomly sampled goal states that are at least 15 steps away. The layout of the obstacles is taken from the classic FourRooms domain, but dilated by a factor of three.

Figure 13. 2D Navigation

F.8. Jaco Reach Environment

We implemented a reaching task using a simulated Jaco robot. Goal states ψ were sampled from uniformly from the interval $[-0.1, 0.1] \times [-0.1, 0.1] \times [0.02, 0.4]$. The agent controlled the velocity of 6 arm joints and 3 finger joints, so the action space was 9 dimensional. The action observation space was 43 dimensional. Let s_{XYZ} denote the XYZ position of the robot end effector. The reward function was defined as

$$r_\psi(s, a) = \mathbb{1}(\|s_{\text{XYZ}} - \psi\|_2 \leq m) - 1.0$$

and the episode terminated when $\|s_{\text{XYZ}} - \psi\|_2 \leq m$, where $m > 0$ is a margin around the goal. We used $m = 0.1$ and $m = 0.01$ in our experiments. We also reset the environment after 250 steps if the agent had failed to reach the goal.

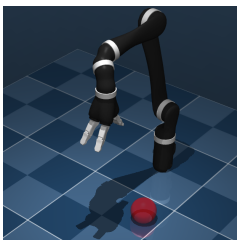

Figure 14. Jaco Reach

F.9. Walker Environment

The walker environment was a modified version of the environment from Tassa et al. (2018). We modified the initial state distribution so the agent always started upright, and modified the observation space to include the termination signal as part of the observation. For the linear reward function, the features are the torso height (normalized by subtracting 0.5m), velocity along the forward/aft axis, the XZ displacement of the two feet relative to the agent's center of mass (the agent cannot move along the Y axis), and the squared L2 norm of the actions. The task coefficients $\psi \in \mathbb{R}^d$ can take on values in the range $[-1, 1]$ for all dimensions, except for the control penalty, which takes on values in $[-1, 0]$. Episodes were 100 steps long.

Figure 15. Walker

F.10. Half-Cheetah Environment

The half-cheetah environment was taken from Tassa et al. (2018). We define tasks to correspond to goal velocities and use the reward function from Rakelly et al. (2019):

$$r_\psi(s, a) = -|s_{\text{vel}} - \psi| - 0.05\|a\|_2^2,$$

where s_{vel} is the horizontal root velocity. Tasks were sampled uniformly $\psi \in [0, 3]$, with units of meters per second. Episodes were 100 steps long.

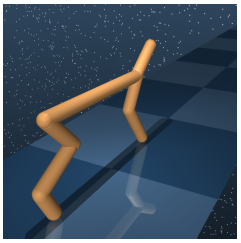

Figure 16. Half-Cheetah

F.11. Desk Environment

The environment provided by Lynch et al. (2019) included 19 tasks. We selected the nine most challenging tasks by looking how often a task was accidentally solved. In the demonstrations for each task, we recorded the average return on the remaining 18 tasks. We chose the nine tasks whose average reward was lowest. The nine tasks were three button pushing tasks and six block manipulation tasks.

For experiments in this environment, we found that normalizing the action space was crucial. We computed the coordinate-wise mean and standard deviation of the actions from the demonstrations, and modified the environment to implicitly normalize actions by subtracting the mean and dividing by the standard deviation. We clipped the action space to $[-1, +1]$, so the agent was only allowed to command actions within one standard deviation (as measured by the expert demos).

Figure 17. Desk Manipulation

Another trick that was crucial for RL in this environment was clipping the critic outputs. Since the reward was in $[0, 1]$ and the episode length was capped at 128 steps, we squashed the Q-value predictions with a scaled sigmoid to be in the range $[0, 128]$.

G. Failed Experiments

1. **100% Relabeling**: When using inverse RL to relabel data for off-policy RL, we initially relabeled 100% of samples from the replay buffer, but found that learning was often worse than doing no relabeling at all. We therefore switched to only 50% relabeling in our experiments. We speculate that retaining some of the originally-commanded goals serves as a sort of hard-negative mining.

2. **Coordinate Ascent on Eq. 2**: We attempted to devise an EM-style algorithm that performed coordinate ascent in Eq. 2, alternating between (1) doing MaxEnt RL and (2) relabeling that data and acquiring the corresponding policy via behavior cloning. While we were unable to get this algorithm to outperform standard MaxEnt RL, we conjecture that this procedure would work with the right choice of inverse RL algorithm.