WHAT WOULD π^* DO?: IMITATION LEARNING VIA OFF-POLICY REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning to imitate expert actions given demonstrations containing image observations is a difficult problem in robotic control. The key challenge is generalizing behavior to out-of-distribution states that differ from those in the demonstrations. State-of-the-art imitation learning algorithms perform well in environments with low-dimensional observations, but typically involve adversarial optimization procedures, which can be difficult to use with high-dimensional image observations. We propose a remarkably simple alternative based on off-policy reinforcement learning, which rewards the agent for matching demonstrated actions in demonstrated states – the key idea is initially filling the agent's experience replay buffer with demonstrations where rewards are set to a positive constant, and setting rewards to zero in all additional experiences. We derive this RL algorithm from first principles as a method for performing approximate inference under the Max-CausalEnt model of expert behavior – the approximate inference objective trades off between a pure behavioral cloning loss and a regularization term that incorporates information about state transitions via the soft Bellman error. Our experiments show that this algorithm matches the state of the art in low-dimensional environments, and significantly outperforms prior work in playing video games from high-dimensional images.

1 INTRODUCTION

Many real-world sequential decision-making problems can be tackled by imitation learning, where an expert demonstrates near-optimal behavior to an agent that then attempts to replicate that behavior in novel situations (Argall et al., 2009; Bojarski et al., 2016; Giusti et al., 2016; Zhang et al., 2017; Rahmatizadeh et al., 2017; 2016). In this paper, we consider the problem of training an agent to imitate an expert policy, given expert action demonstrations and access to the environment. The agent does not get to observe a reward signal or query the expert, and does not know the state transition dynamics ex-ante.

Standard approaches to this problem based on behavioral cloning seek to imitate the expert's actions, but do not reason about the consequences of actions (Pomerleau, 1991). As a result, they suffer from state distribution shift, and fail to generalize to states that are very different from those seen in the demonstrations (Ross & Bagnell, 2010; Ross et al., 2011). Approaches based on inverse reinforcement learning (IRL) deal with this issue by fitting a reward function that represents preferences over trajectories rather than individual actions (Ng et al., 2000; Ziebart et al., 2008), then using the learned reward function to train the imitation agent through reinforcement learning (Finn et al., 2016; Fu et al., 2017). This is the core idea behind generative adversarial imitation learning (GAIL), which implicitly combines IRL and RL using generative adversarial networks (Ho & Ermon, 2016; Goodfellow et al., 2014). GAIL is state-of-the-art in environments with low-dimensional observations, but requires additional reward augmentation and feature engineering when applied to environments with high-dimensional image observations (Li et al., 2017), due to the difficulty of training GANs (Kurach et al., 2018). In short, imitation learning from raw pixel inputs without prior knowledge of the domain remains a challenge.

The key insight in this paper is that instead of using a learned reward function or discriminator to provide a reward signal to the imitation policy, as is done in IRL-based imitation methods and GAIL, we can simply reward the imitation agent for matching demonstrated actions in demonstrated

states using off-policy reinforcement learning – the agent's experience replay buffer is initially filled with demonstrations where rewards are set to some positive constant, and gradually accumulates additional experiences with rewards set to zero. The agent has an incentive to imitate the expert in demonstrated states, and to take actions that lead it back to demonstrated states when it encounters new states that differ from the demonstrations.

We motivate this simple RL algorithm by deriving it from first principles as a method for performing approximate inference under the MaxCausalEnt model of expert behavior (Ziebart et al., 2010). In our analysis, we show that the algorithm maximizes an approximate lower bound on the likelihood of the demonstrations – a bound that trades off between a pure behavioral cloning loss and a regularization term that incorporates state transition information via the soft Bellman error.

The main contribution of this paper is a practical and general imitation learning algorithm that is (1) effective in MDPs with high-dimensional, continuous observations, including raw pixel inputs, and (2) has a natural theoretical interpretation as approximate inference. We run experiments in the image-based Car Racing and low-dimensional Lunar Lander game environments from OpenAI Gym to compare our algorithm to two prior methods: behavioral cloning and GAIL. We find that our method significantly outperforms both prior methods, and achieves the largest gains on the image-based Car Racing task.

2 PRELIMINARIES

In our work, we build on the maximum causal entropy (MaxCausalEnt) model of expert behavior (Ziebart et al., 2010; Levine, 2018). In a Markov Decision Process (MDP) with a discrete action space $\mathcal{A}$,[1] the demonstrator is assumed to follow a policy π that maximizes reward $R(s, a, s')$ under dynamics $T(s'|s, a)$. The policy forms a Boltzmann distribution over actions,

$$\pi(a|s) \triangleq \exp\left(Q(s,a) - V(s)\right), \tag{1}$$

where Q is the soft Q function, which satisfies the soft Bellman equation,

$$Q(s,a) = \mathbb{E}_{s' \sim T(\cdot|s,a)}\left[R(s,a,s') + \gamma V(s')\right], \tag{2}$$

and where V is the soft value function,

$$V(s) \triangleq \log\left(\sum_{a \in \mathcal{A}} \exp\left(Q(s,a)\right)\right). \tag{3}$$

3 IMITATION LEARNING VIA APPROXIMATE MAXCAUSALENT INFERENCE

We aim for an imitation learning algorithm that generalizes to new states, without resorting to adversarial optimization procedures that would be difficult to use with image observations. Our approach is to derive an approximate lower bound on the likelihood of the demonstrations, which can be tractably maximized using a simple RL algorithm.

Preliminaries. To simplify our exposition, we assume the transition dynamics T are deterministic.[2] Let $s' = T(s, a)$ denote the state that follows from taking action a in state s, let $\tau = (s_0, a_0, s_1, ..., s_T)$ denote a rollout, let $\mathcal{D}$ be a set of rollouts, and let B denote the sum of squared soft Bellman errors,

$$B(\mathcal{D}, r) \triangleq \sum_{\tau \in \mathcal{D}} \sum_{t=0}^{T} (Q(s_t, a_t) - (r + \mathbb{1}[t < T]\gamma V(s_{t+1})))^2, \tag{4}$$

where $r \in \mathbb{R}$ is a constant that does not depend on the state or action, and V is the soft value function given by Q and Equation 3. To represent a soft Q function that operates on continuous states, we use

[1] We assume a discrete action space to simplify our exposition and experiments. Our algorithm can be extended to handle MDPs with a continuous action space using existing sampling methods (Haarnoja et al., 2017).

[2] Our method can be extended to MDPs with stochastic dynamics by computing Monte Carlo estimates of $V(s')$.

a function approximator $Q_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$. In our experiments, we use a convolutional neural network or multi-layer perceptron to model $Q_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ are the weights of the neural network (Schmidhuber, 2015).

Deriving the approximate inference objective. Consider the likelihood of a single demonstration rollout $p(\tau)$. In standard MaxCausalEnt IRL, we would search for the reward function R that maximizes the conditional likelihood of the demonstrations $p(\tau|R)$. Our goal, however, is not to infer the expert's reward function R; it is to infer the expert's policy, which is represented by their soft Q function. The difficulty lies in the fact that the expert's soft Q values are a deterministic function of the expert's rewards R, and these rewards are unknown. We approach this problem by formalizing our uncertainty about the rewards as a Bayesian model, in which the rewards are represented as random variables $R(s, a)$. We then fit a soft Q function that maximizes the expectation of the demonstration likelihood under the Bayesian model's prior distribution over the rewards. Formally, we marginalize out the rewards as follows.

$$\begin{aligned} p(\tau) &= \mathbb{E}_R[p(\tau|R)] \\ &= \mathbb{E}_R[\mathbb{E}_Q[p(\tau|Q)|R]]. \end{aligned} \tag{5}$$

Rather than optimize the likelihood in Equation 5 directly, we optimize the log-likelihood, which is more numerically stable, and equivalent to optimizing the likelihood. By Jensen's inequality,

$$\begin{aligned} \log p(\tau) &\geq \mathbb{E}_R[\log \mathbb{E}_Q[p(\tau|Q)|R]] \\ &\geq \mathbb{E}_R[\log\left(p(\tau|Q)p(Q|R)\right)]\ \forall Q \in \mathcal{Q}, \end{aligned} \tag{6}$$

where $\mathcal{Q}$ is the set of all functions $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The conditional likelihood of the demonstrations $p(\tau|Q)$ is given by Equations 1:

$$p(\tau|Q) = \exp\left(\sum_{t=0}^{T} Q(s_t, a_t) - V(s_t)\right). \tag{7}$$

To evaluate $p(Q|R)$, the conditional distribution of the soft Q function given rewards R, recall the soft Bellman equation (Equation 2), which constrains Q to be the soft Q function for rewards R. An equivalent way to express this condition is that Q satisfies $\delta_{Q,R}(s, a) = 0\ \forall s, a$, where $\delta_{Q,R}$ is the soft Bellman error,

$$\delta_{Q,R}(s, a) \triangleq Q(s, a) - (R(s, a) + \gamma V(s')). \tag{8}$$

We can now define $p(Q|R)$ as

$$\begin{aligned} p(Q|R) &= \mathbb{1}[\delta_{Q,R}(s, a) = 0\ \forall s, a] \\ &= \prod_{s,a} \lim_{\sigma \to 0} \mathcal{N}(\delta_{Q,R}(s, a); 0, \sigma^2), \end{aligned} \tag{9}$$

where the last line represents the delta function as the density of a Gaussian distribution with infinitesimally small variance. Searching for a soft Q function that maximizes the lower bound on the log-likelihood given by Equation 6 directly would require running soft Q-learning[3] to convergence on every optimization step, due to the need to evaluate $p(Q|R)$ exactly. This can be computationally prohibitive when using function approximation, since it would involve deep Q-learning in the inner loop of the optimization. To make this optimization problem tractable, we set σ^2 in Equation 9 to a small constant instead of taking it to zero. This is the key approximation made in our analysis, which makes the inference objective tractable to optimize. Next, using Equations 7 and 9 to simplify the lower bound in Equation 6, we have that for all $Q \in \mathcal{Q}$,

$$\begin{aligned} (6) &\approx \mathbb{E}_R[\log p(\tau|Q) + \log \prod_{s,a} \mathcal{N}(\delta_{Q,R}(s, a); 0, \sigma^2)] \\ &= \sum_{t=0}^{T} Q(s_t, a_t) - V(s_t) - \frac{1}{2\sigma^2} \int_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} (Q(s, a) - (\mathbb{E}[R(s, a)] + \gamma V(s')))^2 + \mathrm{Var}[R(s, a)] \mathrm{d}s. \end{aligned} \tag{10}$$

Here, we make an important assumption about the prior distribution over rewards: that $\mathbb{E}[R(s, a)] = 0$. Applying this assumption, we have that

$$(10) = \sum_{t=0}^{T} Q(s_t, a_t) - V(s_t) - \frac{1}{2\sigma^2} \int_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} (Q(s, a) - \gamma V(s'))^2 + \mathrm{Var}[R(s, a)] \mathrm{d}s. \tag{11}$$

[3] the forward RL problem, for which MaxCausalEnt IRL is the inverse problem (Haarnoja et al., 2017; Ziebart et al., 2010; Levine, 2018)

Here, we intentionally loosen the bound, in order to make it more tractable to optimize.[4] From the MaxCausalEnt assumptions, we have that

$$\begin{aligned} V(s_0) &= \mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T} R(s_t, a_t) + \mathcal{H}(\pi(\cdot|s_t)) \right] \\ &\leq T(R_{\max} + \log |\mathcal{A}|) \\ (11) &\geq (11) + V(s_0) - T(R_{\max} + \log |\mathcal{A}|), \end{aligned} \tag{12}$$

where $R_{\max} \in \mathbb{R}$ is a finite upper bound on the reward at each timestep, and $\mathcal{H}$ is entropy. Now, we have that the soft Q function that maximizes our approximate lower bound to the log-likelihood of the demonstrations in Equation 12 is

$$Q^* \triangleq \underset{Q \in \mathcal{Q}}{\arg\max}\, V(s_0) + \sum_{t=0}^{T} Q(s_t, a_t) - V(s_t) - \frac{1}{2\sigma^2} \int_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} (Q(s,a) - \gamma V(s'))^2 \mathrm{d}s. \tag{13}$$

For environments with a discrete state space $\mathcal{S}$, we can simply enumerate the states in the sum in the second term above. For environments with a continuous state space $\mathcal{S}$, we approximate the integral by summing over states and actions from the demonstration rollouts $\mathcal{D}_{\text{demo}}$. To improve this estimate, we also sum over states and actions from additional rollouts $\mathcal{D}_{\text{samp}}$ of the imitation policy, which are periodically sampled during training. We fit the parameterized soft Q function $Q_{\boldsymbol{\theta}}$ to minimize the loss,

$$\ell(\boldsymbol{\theta}) \triangleq \alpha \sum_{\tau \in \mathcal{D}_{\text{demo}}} \left(-V_{\boldsymbol{\theta}}(s_0) + \sum_{t=0}^{T} -(Q_{\boldsymbol{\theta}}(s_t, a_t) - V_{\boldsymbol{\theta}}(s_t)) \right) + \lambda_{\text{demo}} B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{demo}}, 0) + \lambda_{\text{samp}} B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0), \tag{14}$$

where $\alpha, \lambda_{\text{demo}}, \lambda_{\text{samp}} \in \mathbb{R}_{\geq 0}$ are constant hyperparameters, $V_{\boldsymbol{\theta}}$ denotes the soft value function given by $Q_{\boldsymbol{\theta}}$ and Equation 3, and $B_{\boldsymbol{\theta}}$ is defined in Equation 4. The loss in Equation 14 forms an approximate upper bound to the negative log-likelihood of the demonstrations. Searching for a soft Q function that optimizes this objective is equivalent to performing approximate inference under the MaxCausalEnt model of expert behavior. The objective combines a behavioral cloning loss, which does not reason about the consequences of actions, with a regularization term that incorporates information about state transitions. The regularization term performs soft Bellman backups on a reward of zero – a consequence of our prior distribution over the rewards, which has an expected value of zero – and can be seen as imposing a prior distribution on the soft Q values that is independent of the expert demonstrations.

Implementing approximate inference with off-policy RL. Surprisingly, we can show that the gradient of the approximate inference objective in Equation 14 is equivalent to the gradient of the objective of a soft Q-learning algorithm (Haarnoja et al., 2017) that gives the agent a constant positive reward for matching demonstrated actions in demonstrated states, and zero reward otherwise. The key insight here is that soft Q-learning is an off-policy RL algorithm, so we can initially fill the agent's experience replay buffer with demonstration experiences where rewards are positive, and allow the agent to gradually accumulate additional experiences where rewards are zero. Differentiating the objective in Equation 14, we have that

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \alpha &\sum_{\tau \in \mathcal{D}_{\text{demo}}} \left(-\nabla V_{\boldsymbol{\theta}}(s_0) + \sum_{t=0}^{T} -(\nabla Q_{\boldsymbol{\theta}}(s_t, a_t) - \nabla V_{\boldsymbol{\theta}}(s_t)) \right) \\ &+ \lambda_{\text{demo}} \sum_{\tau \in \mathcal{D}_{\text{demo}}} \sum_{t=0}^{T} \nabla (Q_{\boldsymbol{\theta}}(s_t, a_t) - \mathbb{1}[t < T] \gamma V_{\boldsymbol{\theta}}(s_{t+1}))^2 + \lambda_{\text{samp}} \nabla B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0) \\ = &\sum_{\tau \in \mathcal{D}_{\text{demo}}} \left(-\alpha \nabla V_{\boldsymbol{\theta}}(s_0) + \sum_{t=0}^{T} \alpha \nabla V_{\boldsymbol{\theta}}(s_t) - \mathbb{1}[t < T] \alpha \gamma \nabla V_{\boldsymbol{\theta}}(s_{t+1}) \right) \\ &+ \lambda_{\text{demo}} \nabla B_{\boldsymbol{\theta}} \left(\mathcal{D}_{\text{demo}}, \frac{\alpha}{2\lambda_{\text{demo}}} \right) + \lambda_{\text{samp}} \nabla B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0). \end{aligned} \tag{15}$$

[4] This is helpful for deriving a bound that can be optimized by the SQIL algorithm proposed in Algorithm 1. In our experiments, we compare the performance of SQIL, which optimizes the bound in Equation 12, to the performance of SQIL-11, which optimizes the tighter bound in Equation 11.

Algorithm 1 Soft Q Imitation Learning (SQIL, pronounced "skill")

1: Initialize $\mathcal{D}_{\text{samp}} = \emptyset$
2: Initialize $Q_{\boldsymbol{\theta}}$
3: **for** $i = 1, N$ **do**
4: $\quad \boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} (B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{demo}}, r) + \lambda_{\text{samp}} B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0))$ ▷ $\lambda_{\text{samp}} \in \mathbb{R}_{\geq 0}$ is a hyperparameter
5: ▷ See Equation 4
6: $\quad$ **if** $i \bmod k \equiv 0$ **then**
7: $\quad\quad$ Sample rollout $\tau = (s_0, a_0, s_1, a_1, ..., s_T)$ with imitation policy $\pi_{\boldsymbol{\theta}}(a_t|s_t)$ ▷ See Equation 1
8: $\quad\quad \mathcal{D}_{\text{samp}} \leftarrow \mathcal{D}_{\text{samp}} \cup \{\tau\}$
9: $\quad$ **end if**
10: **end for**

If we set $\gamma = 1$, then the inner sum in the first term becomes a telescoping sum, and we have

$$\nabla_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \nabla \left(\lambda_{\text{demo}} B_{\boldsymbol{\theta}} \left(\mathcal{D}_{\text{demo}}, \frac{\alpha}{2\lambda_{\text{demo}}} \right) + \lambda_{\text{samp}} B_{\boldsymbol{\theta}}(\mathcal{D}_{\text{samp}}, 0) \right). \tag{16}$$

Setting $\lambda_{\text{demo}} = 1$ and $\alpha = 2\lambda_{\text{demo}} r = 2r$, we get the gradient of a soft Q-learning algorithm that gives the agent a reward of r for taking the demonstrated action in a demonstrated state, and zero reward otherwise. The agent's replay buffer consists of $\mathcal{D}_{\text{demo}} \cup \mathcal{D}_{\text{samp}}$. Next, we describe this simple RL algorithm in further detail.

Soft Q imitation learning. Our algorithm, which we call soft Q imitation learning (SQIL, pronounced "skill"), is summarized in Algorithm 1.[5] It performs soft Q-learning with two important modifications: (1) it initially fills the agent's experience replay buffer with demonstrations, where the rewards are set to some positive constant, and (2) as the agent interacts with the world and accumulates new experiences, it adds them to the replay buffer, and sets the rewards for these additional experiences to zero. Thus, the agent has an incentive to imitate the expert in states that are similar to those encountered in the demonstrations. During training, the agent may encounter out-of-distribution states that are very different from those in the demonstrations. In such states, the agent has an incentive to take actions that lead it back to demonstration states, where rewards are positive.

4 Simulation Experiments

The purpose of our experiments is two-fold. Our first objective is to compare our method to prior work; in particular, on tasks with high-dimensional image observations, where prior methods are difficult to use. Our second objective is to understand which components of our algorithm contribute most to its performance. To accomplish the first objective, we use the image-based Car Racing environment and the low-dimensional Lunar Lander game from OpenAI Gym (Brockman et al., 2016) to benchmark our SQIL algorithm against GAIL and behavioral cloning (BC). We find that our method consistently outperforms BC and GAIL, and achieves the largest gains on the image-based Car Racing task. To accomplish the second objective, we use the Lunar Lander game to conduct an ablation study in which we test various hypotheses about which components of our method contribute to its good performance.

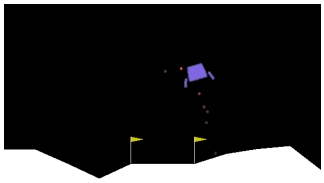
4.1 Comparison to Prior Methods on Low-Dimensional Lunar Lander

In our first set of experiments, we use the low-dimensional Lunar Lander game to verify that SQIL performs well on a relatively easy task. To emphasize the gap between imitation methods that take into account the consequences of actions, like SQIL and GAIL, and methods that do not, like BC, we exacerbate the state distribution shift that usually occurs due to compounding errors, by training the imitation agents in an environment with a different initial state distribution $\mathcal{S}_0^{\text{train}}$ than that of the expert demonstrations $\mathcal{S}_0^{\text{demo}}$. This intervention enables us to explicitly test the generalization

[5] We use Adam to take the gradient step in line 4 of Algorithm 1 (Kingma & Ba, 2014). Sampling experiences uniformly at random from the replay buffer to create each batch for line 4 leads to a vanishing probability of sampling demonstrations, since $\mathcal{D}_{\text{samp}}$ keeps growing while $\mathcal{D}_{\text{demo}}$ stays fixed. We alleviate this problem by balancing the number of demonstration experiences and new experiences sampled in each batch.

capabilities of the imitation methods, by placing the agent in start states $\mathcal{S}_0^{\text{train}}$ that are very different from those seen in the demonstrations.

Experimental setup. The Lunar Lander game is a navigation task in which the objective is to land on the ground, without crashing or flying out of bounds, using two lateral thrusters and a main engine. The action space consists of six discrete actions for steering and firing the main engine. Low-dimensional state observations $s \in \mathbb{R}^9$ encode position, velocity, orientation, and the location of the landing site. To collect demonstrations, we use an expert trained using deep Q-learning (Mnih et al., 2015). Our goal in this experiment is to study not only how well each method can mimic the expert demonstrations, but also how well they can acquire policies that generalize effectively to new states. After all, the main benefit of actually learning a policy, as opposed to simply playing back the expert's actions, is to acquire a strategy that generalizes to new situations. Therefore, we manipulate the initial state distribution of the environment in which the imitation agents are trained and evaluated. The expert demonstrations are collected starting from the initial state distribution $\mathcal{S}_0^{\text{demo}}$. The imitation agents are then trained in the same environment, but starting from a different initial state distribution $\mathcal{S}_0^{\text{train}}$. To create $\mathcal{S}_0^{\text{train}}$ in the Lunar Lander game, we place the agent in a starting position that is rarely visited in the demonstrations. We evaluate the agents by training and testing them with $\mathcal{S}_0^{\text{demo}}$, as well as training and testing them with $\mathcal{S}_0^{\text{train}}$, and report both results in Table 1 and Figure 1. We measure the agents' success rate in landing at the target without crashing, flying out of bounds, or running out of time.



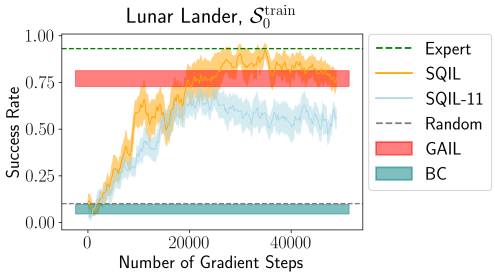


Figure 1: Error regions show standard error across five random seeds.

	$\mathcal{S}_0^{\text{train}}$	$\mathcal{S}_0^{\text{demo}}$
Random	0.10 ± 0.30	0.04 ± 0.02
BC	0.07 ± 0.03	$\mathbf{0.93 \pm 0.03}$
GAIL	0.77 ± 0.04	$\mathbf{0.95 \pm 0.02}$
SQIL-11	0.66 ± 0.02	$\mathbf{0.89 \pm 0.01}$
SQIL	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.88 \pm 0.03}$
Expert	0.93 ± 0.03	0.89 ± 0.31

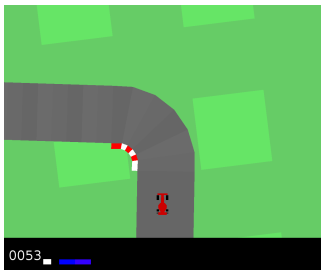
Table 1: Lunar Lander. Best performance on 100 consecutive episodes during training. Standard errors shown for five random seeds.

Analysis. The results show that when there is no variation in the initial state, all methods perform equally well (see the $\mathcal{S}_0^{\text{demo}}$ column in Table 1). The task is easy enough that even behavioral cloning achieves a high success rate. When we change the initial state for the purpose of testing generalization, then SQIL and GAIL both perform much better than BC (see Figure 1 and the $\mathcal{S}_0^{\text{train}}$ column in Table 1). This is unsurprising, since BC suffers from state distribution shift, and GAIL tends to perform well on tasks with low-dimensional observations. SQIL, which optimizes the bound in Equation 12, performs better than its cousin, SQIL-11, which optimizes the tighter bound in Equation 11. This is a somewhat surprising result, since optimizing the tighter bound makes the approximate inference method more accurate, which should lead to better performance. One possible explanation for this not being the case is that in practice, optimizing Equation 11 via gradient descent is harder than optimizing Equation 12.

4.2 Comparison to Prior Methods on Image-Based Car Racing

In Section 4.1, we showed that SQIL performs on par with GAIL and better than BC in environments with low-dimensional observations. Next, we show that SQIL can significantly outperform GAIL and BC in environments with high-dimensional image observations. With images, the policy learning problem becomes substantially harder: now, the policy must learn to interpret raw image observations and determine the best course of action. This is notoriously difficult for methods based on adversarial learning, such as GAIL.

Experimental setup. The Car Racing game is a navigation task in which the objective is to visit as many road markers as possible, while avoiding the grass. The agent receives 64x64 RGB images as observations, and outputs a discrete steering command (left, right, or straight). An episode lasts at most 1000 steps, and terminates early if the car deviates too far from the road. To collect demonstrations, we use an expert trained using model-based reinforcement learning (Ha & Schmidhuber, 2018). We manipulate the initial state distribution, as in Section 4.1. To create $\mathcal{S}_0^{\text{train}}$ in the Car Racing game, we rotate the car 90 degrees so that it begins perpendicular to the track, instead of parallel to the track as in $\mathcal{S}_0^{\text{demo}}$. This simple intervention presents a significant generalization challenge to the imitation learner, since the expert demonstrations do not contain any examples of states where the car is perpendicular to the road, or even significantly off the road axis. The agent must learn to make a tight turn to get back on the road, then stabilize its orientation so that it is parallel to the road, and only then proceed forward to mimic the expert demonstrations. We measure the reward collected by the agent, which counts the number of road markers it visits, and includes a -100 penalty for deviating too far from the road and causing the episode to terminate early.



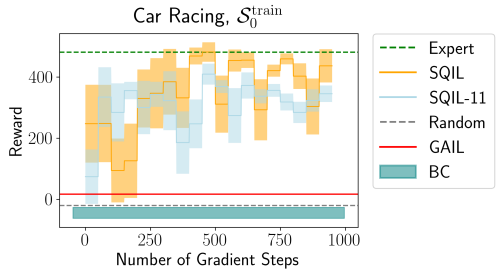


Figure 2: Error regions show standard error across five random seeds.

	$\mathcal{S}_0^{\text{train}}$	$\mathcal{S}_0^{\text{demo}}$
Random	-21 ± 56	-68 ± 4
BC	-45 ± 18	$\mathbf{698 \pm 10}$
GAIL	17 ± 0	7 ± 1
SQIL-11	311 ± 10	$\mathbf{693 \pm 7}$
SQIL	$\mathbf{375 \pm 19}$	$\mathbf{704 \pm 6}$
Expert	480 ± 11	704 ± 79

Table 2: Car Racing. Final performance on 20 episodes after training. Standard errors shown for five random seeds.

Analysis. Table 2 and Figure 2 show that SQIL performs much better than both BC and GAIL, when starting from $\mathcal{S}_0^{\text{train}}$. SQIL learns to make a tight turn that takes the car through the grass and back onto the road, then stabilizes the car's orientation so that it is parallel to the track, and then proceeds forward like the expert does in the demonstrations. GAIL does not improve upon its initialized policy, which keeps turning in a loop, and does not move forward once the agent is parallel to the road. BC tends to drive straight ahead into the grass instead of turning back onto the road. One explanation of these results is that GAIL uses TRPO (Schulman et al., 2015) to fit the imitation policy, which inherits the challenges of reinforcement learning with image observations. SQIL, on the other hand, inherits the ease of fitting an image-based policy from behavioral cloning via the analysis in Section 3, and only relies on soft Bellman backups for regularization and fine-tuning (vs. reinforcement learning from scratch). SQIL outperforms SQIL-11, as in Section 4.1, though the performance gap is smaller here than in Section 4.1.

4.3 Ablation Study

The experiments in Sections 4.1 and 4.2 show that SQIL can outperform prior methods, but they do not illuminate how the various terms in the SQIL gradient in line 4 of Algorithm 1 interact to yield these results. In this section, we use the Lunar Lander environment to conduct an ablation study to better understand the importance of each component of our algorithm. We hypothesize that incorporating the right information about state transitions into the agent's experience replay buffer is essential: the replay buffer needs to contain more than just the demonstrations, and these additional experiences need to be sampled on-policy (see line 7 of Algorithm 1), as opposed to being sampled from a uniform random policy.

Experimental setup. We manipulate the parameters of SQIL to hobble it in various ways that test our hypotheses. In one condition, we set $\lambda_{\text{samp}} = 0$ to prevent SQIL from using additional samples drawn from the training environment. In other conditions, we set $\gamma = 0$ to prevent SQIL from accessing information about state transitions, and use a uniform random policy to sample additional rollouts instead of the imitation policy $\pi_{\boldsymbol{\theta}}$ in line 7 of Algorithm 1. We measure the agents' success rate, as in Section 4.1.

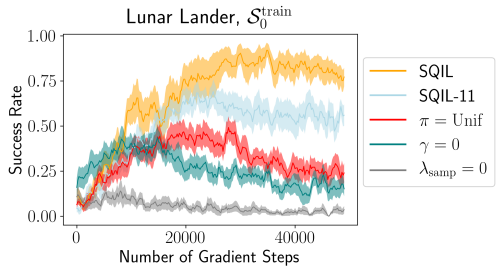


Figure 3: Error regions show standard error across five random seeds.

	$\mathcal{S}_0^{\text{train}}$	$\mathcal{S}_0^{\text{demo}}$
Random	0.10 ± 0.30	0.04 ± 0.02
$\lambda_{\text{samp}} = 0$	0.12 ± 0.02	$\mathbf{0.87 \pm 0.02}$
$\gamma = 0$	0.41 ± 0.02	$\mathbf{0.84 \pm 0.02}$
$\pi = \text{Unif}$	0.47 ± 0.02	$\mathbf{0.82 \pm 0.02}$
SQIL-11	0.66 ± 0.02	$\mathbf{0.89 \pm 0.01}$
SQIL	$\mathbf{0.89 \pm 0.02}$	$\mathbf{0.88 \pm 0.03}$
Expert	0.93 ± 0.03	0.89 ± 0.31

Table 3: Lunar Lander. Best performance on 100 consecutive episodes during training. Standard errors shown for five random seeds.

Analysis. The results in Figure 3 and Table 3 show that the original method performs significantly better than its hobbled counterparts when starting from $\mathcal{S}_0^{\text{train}}$, suggesting that SQIL relies on information about the environment dynamics encoded in state transitions (SQIL vs. $\gamma = 0$), and that SQIL requires on-policy sampling (SQIL vs. $\pi = \text{Unif}$ and $\lambda_{\text{samp}} = 0$). When starting from $\mathcal{S}_0^{\text{demo}}$, all methods perform equally well, since the task is easy enough that even behavioral cloning achieves a high success rate.

5 Discussion and Related Work

Related work. Our algorithm is inspired by the inverse soft Q-learning (ISQL) algorithm for internal dynamics estimation (Reddy et al., 2018), in that the objective in Equation 14 combines a behavioral cloning loss with soft Bellman error penalty terms, which is similar to the ISQL objective. The details of the two methods and their motivations are, however, completely different. ISQL is an internal dynamics estimation algorithm, while ours is meant for imitation learning. ISQL assumes that the demonstrations include observations of the expert's reward signal, while ours does not. Our algorithm also resembles the Deep Q-learning from Demonstrations (DQfD) algorithm (Hester et al., 2017), in that we also fill the agent's experience replay buffer with demonstrations and include an imitation loss in the agent's objective. The key difference between the two methods is that DQfD is a reinforcement learning algorithm that assumes access to a reward signal. Our method is an imitation learning algorithm and does not require an extrinsic reward signal from the environment. Instead, it automatically constructs a reward signal from the demonstrations.

Summary. We contribute a practical and general algorithm for learning to imitate an expert given action demonstrations and access to the environment. Simulation experiments demonstrate the effectiveness of our method at recovering an imitation policy that performs well in comparison to GAIL and behavioral cloning on tasks with high-dimensional, continuous observations.

Limitations. The analysis in Section 3, which proves that SQIL is equivalent to approximate inference of the expert policy under the MaxCausalEnt model of expert behavior, is limited in that does not show (1) that SQIL exactly recovers the expert policy in the limit of having infinite demonstrations, or (2) that the optimal imitation policy's state-action occupancy measure converges to the expert's. These are desirable properties for an imitation learning algorithm, and it is unclear if SQIL possesses them.

Future work. The ability to robustly recover an imitation policy from image observations, even when the initial state distribution differs between demonstration time and training time, enables open-world robotics experiments in which using sensors other than a camera would be expensive, and consistently resetting the environment to the same set of start states is difficult.

REFERENCES

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

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

Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Alessandro Giusti, Jérôme Guzzi, Dan C Ciresan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1(2):661–667, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *arXiv preprint arXiv:1809.01999*, 2018.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, et al. Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*, 2017.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Karol Kurach, Mario Lucic, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*, 2018.

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

Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems*, pp. 3812–3822, 2017.

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.

Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Aman Behal, and Ladislau Bölöni. Learning real manipulation tasks from virtual demonstrations using lstm. *arXiv preprint*, 2016.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. *arXiv preprint arXiv:1707.02920*, 2017.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. Where do you think you're going?: Inferring beliefs about dynamics from behavior. *arXiv preprint arXiv:1805.08010*, 2018.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 661–668, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.

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

Brian D Ziebart, J Andrew Bagnell, and Anind K Dey. Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1255–1262. Omnipress, 2010.