Sample-efficient Adversarial Imitation Learning from Observation

Faraz Torabi [1] **Sean Geiger** [1] **Garrett Warnell** [2] **Peter Stone** [1]

Abstract

Imitation from observation is the framework of learning tasks by observing demonstrated state-only trajectories. Recently, adversarial approaches have achieved significant performance improvements over other methods for imitating complex behaviors. However, these adversarial imitation algorithms often require many demonstration examples and learning iterations to produce a policy that is successful at imitating a demonstrator's behavior. This high sample complexity often prohibits these algorithms from being deployed on physical robots. In this paper, we propose an algorithm that addresses the sample inefficiency problem by utilizing ideas from trajectory centric reinforcement learning algorithms. We test our algorithm and conduct experiments using an imitation task on a physical robot arm and its simulated version in Gazebo and will show the improvement in learning rate and efficiency.

1. Introduction

Teaching new actions to robot actors through demonstration is one of the most attractive methods for behavior learning. While robots can learn new behaviors using reinforcement learning with a pre-specified reward function (Sutton & Barto, 1998), significant exploration is often required to extract the behavior from the reward. In some cases, denser reward functions can help speed up the exploration process, but designing them requires a certain level of skill and understanding of the reinforcement learning process, and can often result in unexpected behaviors when the reward function doesn't precisely guide the action. Instead, teaching a robot a behavior simply by demonstrating it removes the requirement of explicitly specifying a reward function altogether. Anyone who knows how to perform the task can demonstrate it without understanding the learning process, and the learning process requires much less exploration. This process–learning from demonstration (LfD)–aims to take a series of observed states (e.g. joint angles, position in space) and actions (e.g. decisions to move a joint at some speed) and extract a policy that approximates the demonstrated behavior (Argall et al., 2009).

While being able to imitate a behavior after observing the state and actions of a demonstrator is useful, there are many situations where the actions of the demonstrator are unknown. Common approaches to LfD require both the states and actions of the demonstrator to be recorded (Argall et al., 2009). In imitation from external observation (IfO) (Liu et al., 2018; Torabi et al., 2019c), on the other hand, just the observable states of the demonstrator are known--no action information is available. Imitating behaviors solely from observable data greatly expands the set of possible demonstrators: behaviors could be learned from in-person human demonstrators or even the vast collection of videos available online.

While imitation from external observation has been studied and performed with some success for two decades (Ijspeert et al., 2001), recent advances in deep neural networks have widened the set of behaviors that can be imitated and the ways that demonstration data can be collected. One way deep learning has been applied to IfO is through generative adversarial networks (Torabi et al., 2018b; Ho & Ermon, 2016; Chen et al., 2016). In this approach--generative adversarial imitation from observation (GAIfO)--one network learns a control policy for imitating the demonstrator while the other learns to discriminate between the demonstrator's behavior and that of the imitator. While GAIfO advanced the state of the art in imitation from observation, it comes with its own set of challenges. First, in comparison with simpler regressed models, deep networks are notorious for requiring orders of magnitude more training data, and GAIfO is no exception. Second, this algorithm uses model-free reinforcement algorithms which are usually very data inefficient. Some of the possible benefits of the applications of IfO break down when a high sample size is required. Therefore, in practice, this algorithm has been largely limited to being studied in simulation. In simulation, many experiences and large demonstration sets can be collected quickly. Physical demonstrations are more costly to perform, and real-time constraints limit the speed at which control policies can be

[1] University of Texas at Austin, Austin, USA [2] U.S. Army Research Laboratory. Correspondence to: Faraz Torabi <faraztrb@cs.utexas.edu>.

Appearing in the Proceedings of the ICML Workshop on Imitation, Intent, and Interaction (I3), Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

evaluated and thus behavior learned. For imitation from observation to work on a physical robot, a higher degree of sample efficiency is required.

Deep reinforcement learning has faced similar obstacles with learning with limited samples, especially in the context of robotic control policies with complex dynamics. However, recently, trajectory centric reinforcement learning algorithms are being used to guide neural network policy search which has been shown that is very sample-efficient (Levine & Koltun, 2013; Levine & Abbeel, 2014; Levine et al., 2015; 2016). These algorithms achieve this sample efficiency in part by gaining insight into dynamics through the iterative training of linear quadratic regulators (iLQR's) (Tassa et al., 2012) on a set of trajectory controllers.

In this paper, we propose an imitation from observation algorithm, LQR+GAIfO, that takes advantage of both (1) the high performance of the adversarial learning algorithms, and (2) the sample efficiency of trajectory centric reinforcement learning algorithms. We apply the proposed algorithm to a 6-degree-of-freedom robot arm to learn to imitate behaviors from a set of low-level state trajectories. We find that this new method results in successful imitation learning with fewer samples than the previous algorithms.

In Section 2 of this paper, we discuss previous work related to this topic. In Section 3, we cover the techniques involved in GAIfO and LQR. Section 4, describes our approach to combining LQR and GAIfO into one functional algorithm. In Section 5, we share our experimental setup and results, and we discuss results in Section 6. Finally, in Section 7, we summarize and discuss potential future work.

2. Related Work

Our approach to sample-efficient imitation learning is built upon previous works in the field of imitation learning and trajectory-centric reinforcement learning. In the following we discuss previous works on both topics.

Techniques for imitation learning differ in the way they approach the problem. Two popular approaches to imitation learning have been behavioral cloning (Pomerleau, 1991) and inverse reinforcement learning (IRL) (Ng et al., 2000; Russell, 1998). Behavioral cloning views the imitation learning problem as a supervised learning problem that attempts to learn a direct mapping from states to actions. On the other hand, inverse reinforcement learning works to find a cost function under which the expert demonstrator is optimal. One approach of this type is guided cost learning (Finn et al., 2016) which builds on maximum entropy IRL (Ziebart et al., 2008) and guided policy search algorithm (Levine & Abbeel, 2014) and achieves impressive results on physical robots. Later, in 2016, Ho & Ermon used generative adversarial networks to imitate policies when both states and actions are available using a technique called generative adversarial imitation learning (GAIL) (Ho & Ermon, 2016). One imitator network attempts to imitate the policy while another attempts to discriminate between the imitation and provided demonstration data(Goodfellow et al., 2014). Several follow-up works have improved upon this approach on different aspects (Fu et al., 2018; Song et al., 2018) and recently, there has been efforts to address sample efficiency of this algorithm by proposing approaches for unbiasing rewards and deriving an off-policy formulation of adversarial imitation learning algorithms (Kostrikov et al., 2019).

These approaches however, require access to the demonstrator's actions. Recently, on the other hand, imitation learning from observation (Torabi et al., 2018a; 2019c) is becoming more popular in which the agent only has access to state demonstrations of the expert. An algorithm of this type is generative adversarial learning from observation (GAIfO) (Torabi et al., 2018b; 2019a;b; Stadie et al., 2017) which uses a GANs like architecture to bring the state-transition distribution of the imitator closer to that of the demonstrator. While this technique has been shown to discover accurate imitation policies, to date, they have only been evaluated in a simulated experimental domain. Because experiments consist of thousands of iterations in which each iteration includes executing a policy several times, the time required for monitoring experiments is prohibitive.

On the other hand, in reinforcement learning–policy learning through environment-provided reward functions only–direct policy search in a large state-action space requires numerous samples and often can fall into poor local optima. Guided policy search (GPS) is a method to improve the sample efficiency of direct policy search and guide learning in a large space away from poor local optima (Levine & Koltun, 2013). The basis of GPS is to use trajectory optimization to focus policy learning on high-reward actions.

In guided policy search under unknown dynamics, time-varying linear Gaussian models of the dynamics for a small set of specific tasks are first trained to fit a small set of sample data through LQR (Levine & Abbeel, 2014). These Gaussian controllers are then sampled to generate samples to optimize a general policy for a model with thousands of parameters that would typically require much more training data. Specifically, samples in regions of trajectories that have been found to lead to higher reward are generated, guiding the policy learning.

GPS has had success in learning policies in reinforcement learning situations with complex dynamics and high-dimensional inputs, including training a policy that directly controls the torque on motors in a robot arm to perform a task like screwing a cap on a bottle solely from raw images of the system (Levine et al., 2016). Current applications of GPS have focused on reinforcement learning, and the technique's applications to IfO have not been adequately explored.

In this work, our goal is to resolve GAIfOs sample inefficiency with the help of linear quadratic regulators to the extent that it can be applied to learning a behavior on a real robot.

3. Preliminaries and Overview

In this section, we describe the notation considered throughout the paper, and the two methods that our proposed algorithm are based on, (1) adversarial imitation from observation, and (2) trajectory centric reinforcement learning.

3.1. Notation

We consider agents acting within the broad framework of Markov decision processes (MDPs). We denote a MDP using the 5-tuple $M = \{S, A, P, r, \gamma\}$, where S is the agent's state space, A is its action space, $P(s_{t+1}|s_t, a_t)$ is a function denoting the probability of the agent transitioning from state s_t to s_{t+1} after taking action a_t, $r : S \times A \to \mathbb{R}$ is a function specifying the immediate reward that the agent receives for taking a specific action in a given state, and γ is a discount factor. In this framework, agent behavior can be specified by a policy, $\pi : S \to A$, which specifies the action (or distribution over actions) that the agent should use when in a particular state.

In reinforcement Learning the goal is to learn a policy, π, by maximizing the accumulated reward, r, through interaction with the environment. However, imitation learning considers the setting of M\r, i.e. the reward function is excluded. Instead the agent has access to some demonstrated trajectories. The problem that we are interested in this paper is imitation from observation where these demonstrations only include state-trajectories of the expert $\tau_E = \{s_t\}$.

3.2. Adversarial Imitation from Observation

Generative adversarial imitation from observation (Torabi et al., 2018b) is an algorithm of this type in which attempts to learn tasks by bringing the state transition distribution of the imitator closer to that of the demonstrator. The algorithm works as follows. There is an imitator policy network, π_ϕ, that is initialized randomly. This policy is then executed in the environment to generate trajectories τ_π where each trajectory is a set of states $\{(s_0, s_1, ..., s_n)\}$. There is also a discriminator network parameterized by weights θ and maps input trajectories to a score between 0 and 1: $D_\theta : S \times A \to [0, 1]$, The discriminator is trained in a way to output values close to zero for the data coming from the expert and close to one for the data coming from the imitator. To do so, θ is updated by taking an iteration towards solving the following optimization problem.

$$\max_{D_\theta \in (0,1)^{S\times S}} \mathbb{E}_{\tau_\pi}[\log(D_\theta(s, s'))] + \mathbb{E}_{\tau_E}[\log(1 - D_\theta(s, s'))] \tag{1}$$

From a reinforcement learning point of view, the discriminator network provides a cost function that could change ϕ to move the distribution of trajectories created by π_ϕ towards the distribution of the demonstrated trajectories τ_E.

Therefore, following the update to D_θ, the imitator policy, π_ϕ, is updated using the technique of Trust Region Policy Optimization (Schulman et al., 2015) under the cost function

$$\log(D_\theta(s, s')) \tag{2}$$

where D_θ is the newly updated discriminator network. The whole process is repeated until convergence.

It is a quite well-known fact that model-free reinforcement learning algorithms (e.g. TRPO) often require a large number of environment interactions. Therefore, it is not practical to deploy these types of algorithms on physical robots. On the other hand, model-based RL algorithms have shown promising performance in the real world (Levine et al., 2015; 2016).

3.3. Trajectory Centric Reinforcement Learning

Linear quadratic regulators (LQR's) learn control policies under two assumptions (Bemporad et al., 2002):

1. The dynamics of the environment are linear. This means that the transition from a particular state given an action $f(s_t, a_t)$ can be represented as the product of the state/action and a matrix F_t plus a constant vector f_t:
$$f(s_t, a_t) = F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t$$
2. The cost is quadratic. The cost is represented by a quadratic term C_t and a linear vector c_t:
$$c(s_t, a_t) = \frac{1}{2} \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T C_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T c_t$$

The algorithm attempts to solve an optimization problem that returns the actions that have the highest return in the course of an episode. Solving this optimization problem, results in a linear controller:

$$a_t = K_t s_t + k_t \tag{3}$$

where the K_ts and k_ts are matrices and vectors which are combinations of F_ts, C_ts, f_ts, and c_ts that can be computed for each time-step.

In situations where the dynamics are assumed to be close to linear but are not completely known or are non-deterministic, the linear transition function is often replaced by a conditional probability specified under a normal Gaussian distribution, with a mean of the linear dynamics and a covariance:

$$p(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t, \sigma^2)$$

When the covariance is constant (independent of the state and action), the optimal policy is identical to the non-stochastic LQR.

In non-linear systems where the cost is not quadratic, the techniques of LQR can be used by approximating the dynamics with a first-order Taylor expansion and approximating the cost with a second-order Taylor expansion:

$$F_t = \nabla_{s_t,a_t} f(s_t, a_t), \quad C_t = \nabla^2_{s_t,a_t} c(s_t, a_t),$$
$$c_t = \nabla_{s_t,a_t} c(s_t, a_t)$$

Iterative linear quadratic regulators (iLQR's) can be used to find optimal controllers under non-linear models by running LQR with the approximated dynamics, then updating the dynamics fit on each iteration (Li & Todorov, 2004). The resulting controller is:

$$a_t = K_t(s_t - \hat{s}_t) + k_t + \hat{a}_t$$

Where $\hat{s}_t$ and $\hat{a}_t$ are the states and actions around which the Taylor expansion is computed.

LQR assumes that the dynamics of the environment are known. Learning dynamics for a given situation involves building a model to define $f(s_t, a_t)$ from a set of observed state/action transitions $\tau = \{(s_t, a_t, s_{t+1})\}$. A simple approach to this model building is to use linear regression to estimate the dynamics, finding some matrices X and Y that model the transition as $f(s_t, a_t) = Xs_t + Ya_t + c$, or in a stochastic environment, $p(s_{t+1}|s_t, a_t) = \mathcal{N}(Xs_t + Ya_t + c, \sigma^2)$. Modelling dynamics with a Gaussian approximation of the linear regression (often called linear Gaussian models) has the advantage of being very sample-efficient.

To avoid the erroneous pursuit of an incorrect global optimal, a set of local models can be used to replace a global model. The most expressive case of local models is a set of models with a single model for every time-step. In the linear regression approach, this amounts to fitting new X_t and Y_t for every time-step, often called time-varying controllers. Because dynamics are often highly correlated between time-steps, this approach can be refined by using a global model as a prior for a Bayesian linear regression at each time-step. For a better approximation of the local models it is shown that linear-Gaussian controllers, $p(a_t|s_t) = \mathcal{N}(K_t(s_t - \hat{s}_t) + k_t + \hat{a}_t, \Sigma_t)$, should be used for generating the training data (Levine et al., 2016). The covariance depends on the sensitivity of the total cost to the choice of action.

Because linear regression can overshoot optimals of non-linear dynamics, policy adjustment can be bounded so that each iteration's update to the model's transition distribution (or trajectory distribution) is not too large. This can be achieved with a bound on the Kullback–Leibler (KL) divergence–a relative measure of divergence between distributions–between the previous trajectory distribution and the current trajectory distribution.

4. Proposed Algorithm

In this section, we propose an imitation from observation algorithm, LQR+GAIfO, to learn an imitation policy from state only demonstrations, τ_E. Our algorithm takes advantage of the high performance of adversarial imitation from observation algorithms and the sample efficiency of trajectory-centric reinforcement learning algorithms. To do so, we build upon the methods described in Section 3. For LQR to be useful in an imitation learning scenario, it can no longer depend on a pre-specified reward function that defines the task. Instead, the trajectory optimization step in LQR should be based on the existing controller's ability to imitate the expert demonstration. To achieve this capability, we train a discriminator network on each iteration and use an approximate version of its loss on the sampled trajectories to optimize the controllers.

Our algorithm begins by initializing the linear Gaussian controller and executing it inside the environment to collect state-action trajectories $\{(s_t, a_t)\}$. Then it randomly initializes a time-varying model p to model the trajectory dynamics. p is specified as $p(s_{t+1}|s_t, a_t) = \mathcal{N}(F_t \begin{bmatrix} s_t \\ a_t \end{bmatrix} + f_t, \sigma^2)$. Given a set of state-action trajectories $\{s_t, a_t\}$, F_t, f_t, and σ^2 are fit to the sample data at each time-step using Bayesian linear regression with a normal-inverse-Wishart prior. For this prior, it fits the entire trajectory sample to a Gaussian mixture model (GMM), which previous research has found to be effective (Levine et al., 2016).

Following the dynamics model update, a randomly initialized neural network is considered as the discriminator, D_θ, which takes state-transitions (s_t, s_{t+1}) as input and outputs a value. Similar to Section 3.2, The goal is to train the discriminator to distinguish between the state-transitions coming from the controller and the demonstrator. However, in order to stabilize the learning, our algorithm uses Wasserstein loss (Arjovsky et al., 2017) and takes an iteration on the following optimization problem.

$$\min_{D_\theta^{S \times S}} \mathbb{E}_{p(a|s)}[D_\theta(s, s')] - \mathbb{E}_{\tau_E}[D_\theta(s, s'))]$$

Algorithm 1 LQR+GAIfO

```
Initialize controller $p(a|s)$
Initialize a neural network discriminator $D_\theta$ with random parameter $\theta$
Obtain state-only expert demonstration trajectories $\tau_E = \{s_t\}$
while Controller Improves do
    Execute the controller, $p(a|s)$, and store the resulting trajectories $\tau_{p(a|s)} = \{(s, a, s')\}$
    Learn dynamics model $p(s'|s, a)$ over $\tau$
    Update $D_\theta$ using loss
        $\min_{D_\theta^{S \times S}} \mathbb{E}_{\tau_{p(a|s)}}[D_\theta(s, s')] - \mathbb{E}_{\tau_E}[D_\theta(s, s'))]$
    Create the composite function $C(s_t, a_t) = (D_\theta \circ f_t)(s_t, a_t)$
    Compute the quadratically approximated cost function by taking the second order Taylor expansion of $C(s_t, a_t)$
        $c_q(s_t, a_t) = \frac{1}{2} \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T \nabla^2_{s,a} C(s_t, a_t) \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T \nabla_{s,a} C(s_t, a_t)$
    Improve controller $p(a|s)$ by LQR
end while
```

Gradient penalties are also used as the regularization for further stabilization of the learning process (Gulrajani et al., 2017). As discussed in Section 3, the discriminator—a function of state-transition (s_t, s_{t+1})—can be used as the cost function for training the controller. However, LQR requires the cost function to be a quadratic function of states and actions. Therefore, first, the discriminator is combined with the Gaussian dynamics models to create a composite cost function $C(s_t, a_t) = (D_\theta \circ f_t)(s_t, a_t)$. This composite function is then quadratically approximated by taking the second order Taylor expansions of the cost:

$$c_q(s_t, a_t) = \frac{1}{2} \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T \nabla^2_{s,a} C(s_t, a_t) \begin{bmatrix} s_t \\ a_t \end{bmatrix} + \begin{bmatrix} s_t \\ a_t \end{bmatrix}^T \nabla_{s,a} C(s_t, a_t)$$

Where $\nabla^2_{s,a}$ and $\nabla_{s,a}$ are the Hessian and gradient with respect to the concatenation of s and a vectors, respectively. Finally, an iteration of LQR uses this cost approximation c_q to optimize the trajectory to form a new linear-Gaussian controller. The step size of this update is bounded by the KL-Divergence compared to the previous iteration. The main components of this approach are depicted in Figure 1.

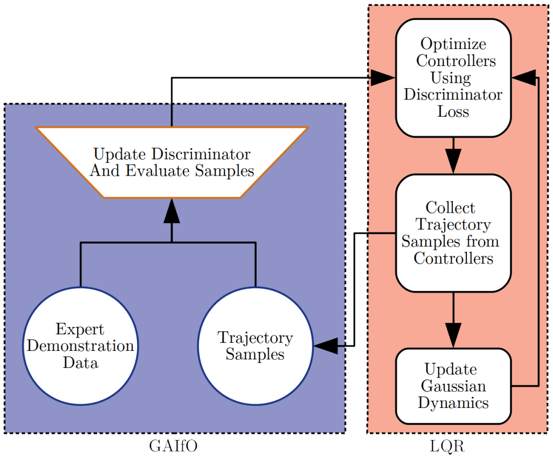

Optimize Controllers Using Discriminator Loss
Update Discriminator And Evaluate Samples
Collect Trajectory Samples from Controllers
Expert Demonstration Data
Trajectory Samples
Update Gaussian Dynamics
GAIfO
LQR

Figure 1. The proposed algorithm, LQR+GAIfO.

5. Experiments

To evaluate the performance of our algorithm, we studied its ability to imitate a reaching task on a robot arm–both on a physical arm and in a simulator.

5.1. Setup

For a testing platform, we used a Universal Robotics UR5, a 6-degree-of-freedom robotic arm (Figure 2). The task that is demonstrated is a reaching task in which the arm begins in a consistent, retracted position and reaches towards a point in Cartesian space. When the end effector (the gripper at the end of the arm) reaches this point, the arm stops moving. This task is shown in Figure 4. The expert is trained by iterating between iLQR and dynamics learning with a specified reward function until convergence. This policy is then executed and recorded a number of times to create the demonstration data.

We modified the software to record the state of the arm and the action chosen at every time-step of the trajectory execution. For the initial experiments, the state consisted of:

1. Joint angles (3 dimensional)
2. Joint velocities (3 dimensional)
3. Cartesian distance to the goal position from the end effector (3 dimensional)
4. Cartesian velocity of the end effector (3 dimensional)

For testing in simulation, we used the Gazebo simulation environment (Figure 3) with a model of the UR5. Each trial lasts for 100 timesteps (10 seconds) and ends regardless of the end effector reaching the goal state. At each iteration, the policy being evaluated is executed five times to collect five sample trajectories. The policy is also evaluated once without noise per iteration, and the performance according to the cost function is logged.

The cost function used takes into account the distance from the end effector to the target position, weighted linearly as the trial progresses. With the distance from the goal position to the end effector at a given time-step d_t, the cost of a trajectory with n time-steps is calculated as:

$$C(\tau) = d_{t_n} + \sum_{i=0}^{n} \frac{i}{n} d_{t_i}$$

The same cost function is used to train the expert through reinforcement learning as well as to evaluate the performance of the imitator. In this sense, the task of imitation learning can be seen as recovering the cost function that guided the expert (Torabi et al., 2018b). For a more complex task or more specific cost function than the one studied, it's possible that the imitator could recover the task behavior correctly while not performing well in the eyes of the cost function, or vice versa. However, for the arm reaching task, the cost function is simple and directly related to the task, making it appropriate as an evaluator of imitation performance. For the imitation tasks, this cost function was used to evaluate each trajectory sample at a given iteration. The results were normalized on a range from zero to one, with zero mapping to the average cost of a random policy, and one mapping to the cost achieved by the expert. A policy that performs as well as the expert would achieve a score of one on this normalized performance scale.

We compare our algorithm with GAIfO which is instrumented to interface with the arm control and simulation platform. Trials for the GAIfO also involved taking five samples per iteration, in the same way as ours. The GAIfO policy network was updated using Proximal Policy Optimization (PPO).

5.2. Experimental Design

We conducted three main experiments to evaluate our algorithm. In the first experiment, the learning rate is compared to learning under GAIfO. In the second experiment, we test our algorithm's ability to generalize to unseen target positions. Finally, we compare the performance of the algorithm in the simulated environment to the physical arm.

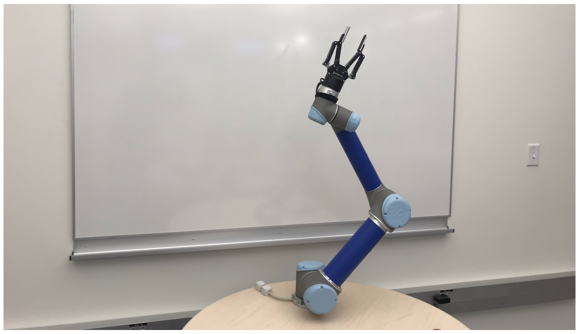
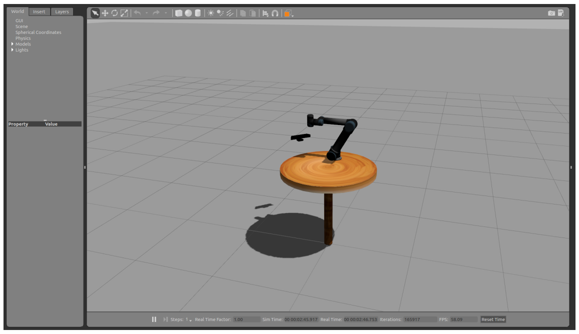


Figure 2. The UR5 Robot Arm



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:02:45.917 Real Time: 00 00:02:46.753 Iterations: 165917 FPS: 58.09 Reset Time

Figure 3. The UR5 Arm Modeled in the Gazebo Simulator

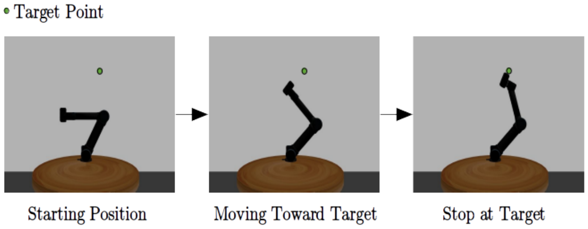


Figure 4. A depiction of the reaching task being demonstrated in the simulator. The arm starts in a retracted position and reaches the end effector toward the target point, stopping when the target point is reached.

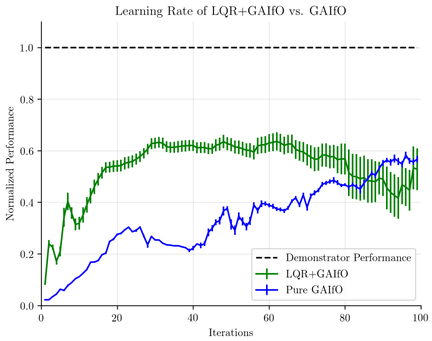


Figure 5. Learning rate comparison of LQR+GAIfO to GAIfO in simulation. The normalized performance is shown, with 0.0 denoting the performance of a random policy, and 1.0 denoting the performance of the demonstrator. The error bars show the mean standard error of the policy samples.

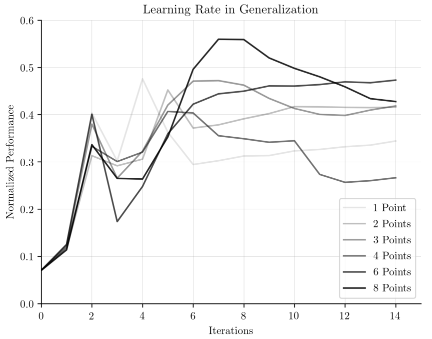


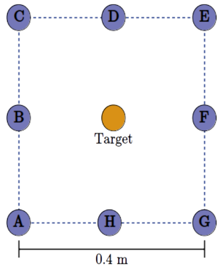
Figure 6. The learning rate of LQR+GAIfO in simulation when tasked with reaching an unseen point given a demonstration set with a varying number of demonstration points.

5.2.1. Comparison to GAIfO

To compare the learning rate of our algorithm to that of GAIfO, we ran trials for both algorithms for 100 iterations and tracked the policy's performance at each iteration using the cost function described in Section 5.1. This process was repeated for both algorithms (n=30 for ours, n=55 for GAIfO) to collect average performance data. The algorithms' performance along with the mean standard error is plotted in Figure 5. The performance of our algorithm quickly exceeds GAIfO and peaks around iteration 30.

5.2.2. Generalization

To test our algorithm's ability to generalize a policy for a point that is not in the expert demonstration data, we collected expert demonstration trajectories for 8 points on the edge of a square (shown in Figure 7). For each point, we trained the expert and recorded five sample trajectories when the expert converged. Then, after choosing a subset of the points on the square as $\{\tau_E\}$, we tasked the arm with moving to a point in the center of the square. Because the center point was not in $\{\tau_E\}$, the control policy was required to generalize the expert trajectories to this unseen point. We varied the number of points included in $\{\tau_E\}$, and tracked the normalized performance of our algorithm over 15 iterations. As shown in Figure 6, while performance was similar in the early iterations, our algorithm generally performed better in later iterations when more points were included in $\{\tau_E\}$.



# Points	Points Used for $\{\tau_E\}$
1	B
2	B, F
3	B, D, F
4	B, D, F, H
6	A, B, D, E, F, H
8	A, B, C, D, E, F, G, H

Figure 7. Points collected for expert data in the generalization experiment. A varying number of points were chosen from the edges of a square surrounding the target point. Demonstrated trajectories to these chosen points form the expert demonstration set.

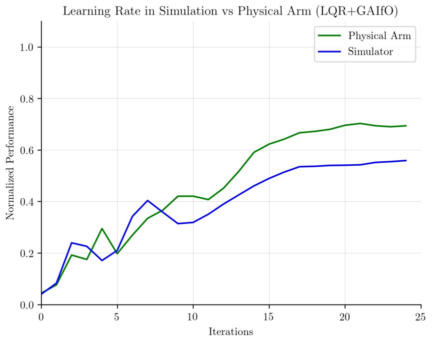


Figure 8. The normalized performance of LQR+GAIfO in simulation compared to the normalized performance of LQR+GAIfO on the physical UR5 arm over 25 iterations.

5.2.3. Performance on Physical Arm

Our algorithm was run on both the simulator and the physical arm to examine how closely simulated performance mapped to real-world performance. Over 25 iterations, the policy performance on the physical arm began to surpass the performance of the simulated arm, as shown in Figure 8.

6. Discussion

Our research began by asking if a combination of LQR and GAIfO could increase sample efficiency in imitation learning. The comparison of LQR+GAIfO to GAIfO suggests that LQR+GAIfO can indeed produce a policy that is better at imitating a behavior in a limited number of iterations, confirming our hypothesis. The steep initial learning curve of LQR+GAIfO indicates significantly higher sample efficiency compared to GAIfO alone. However, the performance of LQR+GAIfO seems to degrade around iteration 60. Without this performance degradation, LQR+GAIfO would outperform GAIfO past iteration 100. The reason for this degradation may be that in adversarial algorithms, improvement of the generator and the discriminator should occur at relatively similar rate. However, in our algorithm, since the controller's representation complexity is limited, after some number of iteration, the controller does not improve as fast as the discriminator. In addition, even without this degradation, the GAIfO approach would eventually surpass the performance of LQR+GAIfO, likely due to the ability of the generator network in GAIfO to produce more complex policies than those that can be represented with linear Gaussian controllers in LQR.

Although most of the ability for a policy to perform a task that is different from the expert trajectories in GAIfO and GPS result from a complex model considered for the policy (neural network), the linear Gaussian controllers in LQR+GAIfO still have the ability to generalize to some degree. As expected, the ability to successfully generalize increases with demonstration trajectories, as shown in Figure 6. The reason may be that the discriminator learns a general cost function that could be applied to new target points and as a result LQR can learn a relatively good controller. Future work integrating the full GPS approach would likely lead to better generalization.

We studied the performance of LQR+GAIfO on the physical arm to validate the tractability of this technique on a real robot and to establish a sense of how directly the performance studied in the simulator would translate to the physical arm. Our results, as seen in Figure 8, show that the policy performance seen in the simulator can be trusted to model policy performance on the real arm. Surprisingly, the performance of LQR+GAIfO on the physical arm exceeds the simulator performance. It is possible that the noise introduced by the physical arm as a result of actuator noise or other physical effects lead to wider exploration and faster policy improvement. If this is the case, it could be possible to achieve similar performance in the simulator by introducing more policy noise.

7. Conclusion and Future Work

We have found that combining generative adversarial imitation from observation with Linear Quadratic Regulators leads to faster learning of imitation behavior over fewer samples than with GAIfO alone, confirming our hypothesis. While LQR+GAIfO doesn't reach the absolute imitation performance of GAIfO over an extended training period with thousands of samples, achieving adequate imitation performance with limited samples opens the door to imitation research on physical robotic systems, for which imitation learning has posed logistical challenges in the past.

While LQR is a powerful technique by itself, a policy based solely on Gaussian controllers has limits in complexity. Work in GPS has already produced a method for combining sample-efficient Gaussian controllers with a deep network model that is trained through the controllers. Using a deep network as part of the policy offers increased performance in the long run and greatly increased generalization ability. Incorporating this deep network policy driven by importance-weighted samples of the linear Gaussian controllers is an obvious and promising next step for this work.

To validate the LQR+GAIfO technique, we represented the expert trajectories using low-level data like the Cartesian position of the arm's end effector. GAIfO has had success in using higher level data–like a visual recording of the demonstrator–as the state in trajectories. Additionally, GPS has been used in learning neural network policies from visual observation (Levine et al., 2016). Pursuing imitation learning from visual data alone would greatly widen the situations in which demonstration data could be collected. Adding a convolutional layer to the discriminator so that it can accept visual data is a natural next step for extending this research.

Acknowledgements

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-1637736, IIS-1651089, IIS-1724157), the Office of Naval Research (N00014-18-2243), Future of Life Institute (RFP2-000), Army Research Lab, DARPA, Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

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

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E. N. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002. ISSN 00051098. doi: 10.1016/S0005-1098(01)00174-1.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

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

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

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

Ijspeert, A. J., Nakanishi, J., and Schaal, S. Trajectory formation for imitation with nonlinear dynamical systems. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 2, pp. 752–757. IEEE, 2001.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. 2019.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

Levine, S. and Koltun, V. Guided Policy Search. *Proceedings of the 30th International Conference on Machine Learning*, 28:1–9, 2013. URL http://jmlr.org/proceedings/papers/v28/levine13.html.

Levine, S., Wagener, N., and Abbeel, P. Learning contact-rich manipulation skills with guided policy search. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015-June(June):156–163, 2015. ISSN 10504729. doi: 10.1109/ICRA.2015.7138994.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Li, W. and Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pp. 222–229, 2004.

Liu, Y., Gupta, A., Abbeel, P., and Levine, S. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125. IEEE, 2018.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Pomerleau, D. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural Computation*, 1991.

Russell, S. J. Learning agents for uncertain environments. In *COLT*, volume 98, pp. 101–103, 1998.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015.

Song, J., Ren, H., Sadigh, D., and Ermon, S. Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 7461–7472, 2018.

Stadie, B. C., Abbeel, P., and Sutskever, I. Third-person imitation learning. 2017.

Sutton, R. and Barto, A. *Reinforcement Learning: An introduction*. MIT Press, Cambridge, volume 1 edition, 1998.

Tassa, Y., Erez, T., and Todorov, E. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4906–4913. IEEE, 2012.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957. AAAI Press, 2018a.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018b.

Torabi, F., Warnell, G., and Stone, P. Adversarial imitation learning from state-only demonstrations. In *International Conference on Autonomous Agents and Multi-Agent Systems*, 2019a.

Torabi, F., Warnell, G., and Stone, P. Imitation learning from video by leveraging proprioception. In *International Joint Conference on Artificial Intelligence*, 2019b.

Torabi, F., Warnell, G., and Stone, P. Recent advances in imitation learning from observation. In *International Joint Conference on Artificial Intelligence*. AAAI Press, 2019c.

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