

Model-based Adversarial Imitation Learning

Nir Baram, Oron Anschel, Shie Mannor

Presented by Yuwen Xiong, Mar 1st

Recap: GAIL algorithm

$$\operatorname{argmin}_{\pi} \operatorname{argmax}_{D \in (0,1)} \mathbb{E}_{\pi} [\log D(s, a)] + \mathbb{E}_{\pi_E} [\log(1 - D(s, a))] - \lambda H(\pi)$$

- We use Adam to optimize the discriminator and use TRPO to optimize the policy

Recap: GAIL algorithm

$$\operatorname{argmin}_{\pi} \operatorname{argmax}_{D \in (0,1)} \mathbb{E}_{\pi} [\log D(s, a)] + \mathbb{E}_{\pi_E} [\log(1 - D(s, a))] - \lambda H(\pi)$$

- We use Adam to optimize the discriminator and use TRPO to optimize the policy
- The optimization of the discriminator can be done by using backpropagation, but this is not the case for the optimization of the policy

Recap: GAIL algorithm

$$\operatorname{argmin}_{\pi} \operatorname{argmax}_{D \in (0,1)} \mathbb{E}_{\pi} [\log D(s, a)] + \mathbb{E}_{\pi_E} [\log(1 - D(s, a))] - \lambda H(\pi)$$

- We use Adam to optimize the discriminator and use TRPO to optimize the policy
- The optimization of the discriminator can be done by using backpropagation, but this is not the case for the optimization of the policy
- π affects the data distribution but do not appear in the objective itself
- We use these two equations to get gradient estimation for π_{θ}

$$\nabla_{\theta} \mathbb{E}_{\pi} [\log D(s, a)] \cong \hat{\mathbb{E}}_{\tau_i} [\nabla_{\theta} \log \pi_{\theta}(a|s) Q(s, a)]$$

$$Q(\hat{s}, \hat{a}) = \hat{\mathbb{E}}_{\tau_i} [\log D(s, a) \mid s_0 = \hat{s}, a_0 = \hat{a}]$$

Motivation

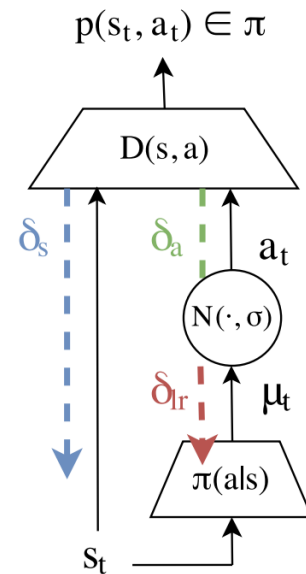
- A model-free approach like GAIL has its limitations
 - The generative model can no longer be trained by simply backpropagating the gradient from the loss function defined over the discriminator
 - Has to resort to high-variance gradient estimations

Motivation

- A model-free approach like GAIL has its limitations
 - The generative model can no longer be trained by simply backpropagating the gradient from the loss function defined over the discriminator
 - Has to resort to high-variance gradient estimations
- If we have a model-based version of adversarial imitation learning
 - The system can be easily trained end-to-end using regular backpropagation
 - The policy gradient can be derived directly from the gradient of the discriminator
 - Policies can be more robust and training requires fewer interactions with the environment

Algorithm - overview

The model-free approach treats the state s as fixed and only tries to optimize the behavior.



Algorithm - overview

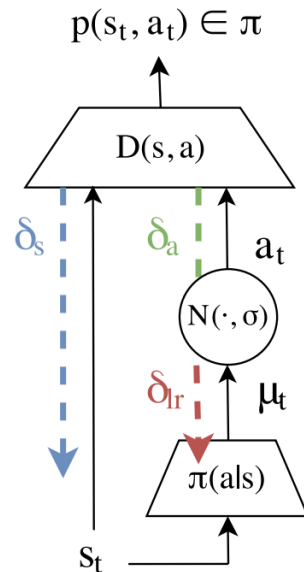
The model-free approach treats the state s as fixed and only tries to optimize the behavior.

Instead, we treat s as a function of the policy:

$$s' = f(s, a)$$

So that, by using the law of total derivative we can get:

$$\begin{aligned} \nabla_{\theta} D(s_t, a_t) \Big|_{s=s_t, a=a_t} &= \frac{\partial D}{\partial a} \frac{\partial a}{\partial \theta} \Big|_{a=a_t} + \frac{\partial D}{\partial s} \frac{\partial s}{\partial \theta} \Big|_{s=s_t} \\ &= \frac{\partial D}{\partial a} \frac{\partial a}{\partial \theta} \Big|_{a=a_t} + \frac{\partial D}{\partial s} \left(\frac{\partial f}{\partial s} \frac{\partial s}{\partial \theta} \Big|_{s=s_{t-1}} + \frac{\partial f}{\partial a} \frac{\partial a}{\partial \theta} \Big|_{a=a_{t-1}} \right) \end{aligned}$$



Algorithm - preparation

First, we know that $D(s, a) = p(y|s, a)$, where $y = \{\pi_E, \pi\}$

Algorithm - preparation

First, we know that $D(s, a) = p(y|s, a)$, where $y = \{\pi_E, \pi\}$

By using Bayes rule and the law of total probability we can get:

$$\begin{aligned} D(s, a) &= p(\pi|s, a) = \frac{p(s, a|\pi)p(\pi)}{p(s, a)} \\ &= \frac{p(s, a|\pi)p(\pi)}{p(s, a|\pi)p(\pi) + p(s, a|\pi_E)p(\pi_E)} \end{aligned}$$

Algorithm - preparation

First, we know that $D(s, a) = p(y|s, a)$, where $y = \{\pi_E, \pi\}$

By using Bayes rule and the law of total probability we can get:

$$\begin{aligned} D(s, a) &= p(\pi|s, a) = \frac{p(s, a|\pi)p(\pi)}{p(s, a)} \\ &= \frac{p(s, a|\pi)p(\pi)}{p(s, a|\pi)p(\pi) + p(s, a|\pi_E)p(\pi_E)} \\ &= \frac{p(s, a|\pi)}{p(s, a|\pi) + p(s, a|\pi_E)} \end{aligned}$$

Algorithm - preparation

Re-writing it as following:

$$D(s, a) = \frac{1}{\frac{p(s, a | \pi) + p(s, a | \pi_E)}{p(s, a | \pi)}} = \frac{1}{1 + \frac{p(s, a | \pi_E)}{p(s, a | \pi)}} = \frac{1}{1 + \frac{p(a | s, \pi_E)}{p(a | s, \pi)} \cdot \frac{p(s | \pi_E)}{p(s | \pi)}}$$

Algorithm - preparation

Re-writing it as following:

$$D(s, a) = \frac{1}{\frac{p(s, a | \pi) + p(s, a | \pi_E)}{p(s, a | \pi)}} = \frac{1}{1 + \frac{p(s, a | \pi_E)}{p(s, a | \pi)}} = \frac{1}{1 + \frac{p(a | s, \pi_E)}{p(a | s, \pi)} \cdot \frac{p(s | \pi_E)}{p(s | \pi)}}$$

Let $\varphi(s, a) = \frac{p(a | s, \pi_E)}{p(a | s, \pi)}$ and $\psi(s) = \frac{p(s | \pi_E)}{p(s | \pi)}$, we can get:

$$D(s, a) = \frac{1}{1 + \varphi(s, a) \cdot \psi(s)}$$

Algorithm - preparation

Here $\varphi(s, a) = \frac{p(a|s, \pi_E)}{p(a|s, \pi)}$ stands for policy likelihood ratio

And $\psi(s) = \frac{p(s|\pi_E)}{p(s|\pi)}$ stands for state distribution likelihood ratio

Algorithm - preparation

Here $\varphi(s, a) = \frac{p(a|s, \pi_E)}{p(a|s, \pi)}$ stands for policy likelihood ratio

And $\psi(s) = \frac{p(s|\pi_E)}{p(s|\pi)}$ stands for state distribution likelihood ratio

By using differentiation rule we can easily get:

$$\nabla_a D = - \frac{\varphi_a(s, a) \psi(s)}{(1 + \varphi(s, a) \psi(s))^2}$$

$$\nabla_s D = - \frac{\varphi_s(s, a) \psi(s) + \varphi(s, a) \psi_s(s)}{(1 + \varphi(s, a) \psi(s))^2}$$

Algorithm - preparation

Here $\varphi(s, a) = \frac{p(a|s, \pi_E)}{p(a|s, \pi)}$ stands for policy likelihood ratio

And $\psi(s) = \frac{p(s|\pi_E)}{p(s|\pi)}$ stands for state distribution likelihood ratio

By using differentiation rule we can easily get:

$$\nabla_a D = - \frac{\varphi_a(s, a) \psi(s)}{(1 + \varphi(s, a) \psi(s))^2}$$

$$\nabla_s D = - \frac{\varphi_s(s, a) \psi(s) + \varphi(s, a) \psi_s(s)}{(1 + \varphi(s, a) \psi(s))^2}$$

Recall what we need: $\nabla_{\theta} D(s_t, a_t) \Big|_{s=s_t, a=a_t} = \frac{\partial D}{\partial a} \frac{\partial a}{\partial \theta} \Big|_{a=a_t} + \frac{\partial D}{\partial s} \frac{\partial s}{\partial \theta} \Big|_{s=s_t}$

Algorithm - re-parameterization of distribution

Assuming the policy is given by

$$\pi_{\theta}(a|s) = \mathcal{N}(a|\mu_{\theta}(s), \sigma_{\theta}^2(s))$$

Algorithm - re-parameterization of distribution

Assuming the policy is given by

$$\pi_{\theta}(a|s) = \mathcal{N}(a|\mu_{\theta}(s), \sigma_{\theta}^2(s))$$

We can rewrite it to

$$\pi_{\theta}(a|s) = \mu_{\theta}(s) + \xi\sigma_{\theta}(s), \text{ where } \xi \sim \mathcal{N}(0, 1)$$

Algorithm - re-parameterization of distribution

Assuming the policy is given by

$$\pi_{\theta}(a|s) = \mathcal{N}(a|\mu_{\theta}(s), \sigma_{\theta}^2(s))$$

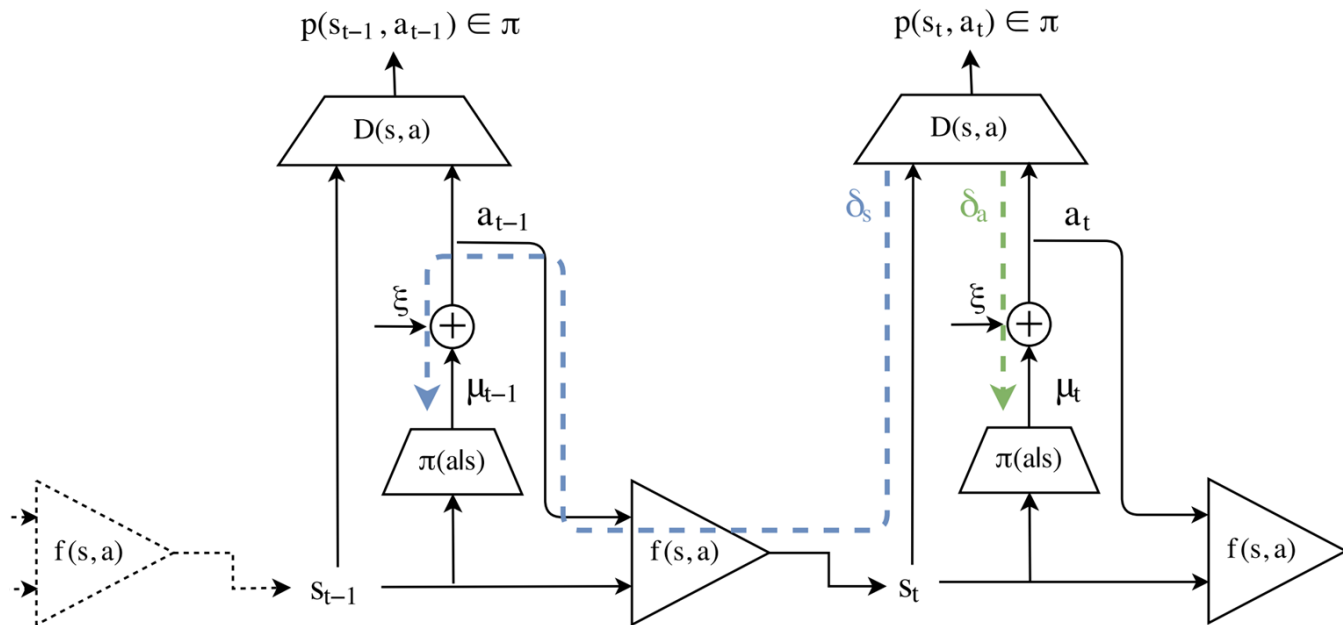
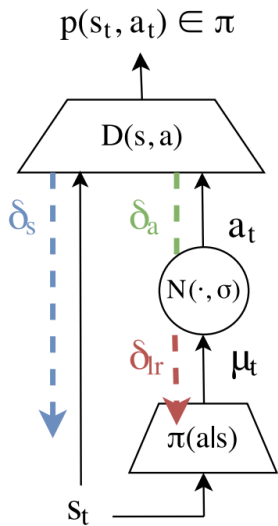
We can rewrite it to

$$\pi_{\theta}(a|s) = \mu_{\theta}(s) + \xi\sigma_{\theta}(s), \text{ where } \xi \sim \mathcal{N}(0, 1)$$

So that we can get a Monte-Carlo estimator of the derivative

$$\begin{aligned}\nabla_{\theta}\mathbb{E}_{\pi(a|s)}D(s, a) &= \mathbb{E}_{\rho(\xi)}\nabla_a D(a, s)\nabla_{\theta}\pi_{\theta}(a|s) \\ &\cong \frac{1}{M}\sum_{i=1}^M \nabla_a D(s, a)\nabla_{\theta}\pi_{\theta}(a|s)\Big|_{\xi=\xi_i}\end{aligned}$$

Algorithm



Model-free block diagram

Model-based block diagram

Algorithm

To maximize the reward function, we can view reward as $r(s, a) = -D(s, a)$, and then maximizing the total reward is equivalent to minimizing the total discriminator beliefs along a trajectory.

Algorithm

To maximize the reward function, we can view reward as $r(s, a) = -D(s, a)$, and then maximizing the total reward is equivalent to minimizing the total discriminator beliefs along a trajectory.

So that we can define:

$$J(\theta) = \mathbb{E} \left[\sum_{t=0} \gamma^t D(s_t, a_t) | \theta \right]$$

And write down the derivatives: (this follows SVG paper [Heess et al. 2015])

$$J_s = \mathbb{E}_{p(a|s)} \mathbb{E}_{p(s'|s,a)} \mathbb{E}_{p(\xi|s,a,s')} \left[D_s + D_a \pi_s + \gamma J'_{s'} (f_s + f_a \pi_s) \right]$$

$$J_\theta = \mathbb{E}_{p(a|s)} \mathbb{E}_{p(s'|s,a)} \mathbb{E}_{p(\xi|s,a,s')} \left[D_a \pi_\theta + \gamma (J'_{s'} f_a \pi_\theta + J'_\theta) \right]$$

Algorithm

Algorithm 1 Model-based Adversarial Imitation Learning

```
1: Given empty experience buffer  $\mathcal{B}$ 
2: for  $trajectory = 0$  to  $\infty$  do
3:   for  $t = 0$  to  $T$  do
4:     Act on environment:  $a = \pi(s, \xi; \theta)$ 
5:     Push  $(s, a, s')$  into  $\mathcal{B}$ 
6:   end for
7:   train forward model  $f$  using  $\mathcal{B}$ 
8:   train discriminator model  $D$  using  $\mathcal{B}$ 
9:   set:  $j'_s = 0, j'_\theta = 0$ 
10:  for  $t = T$  down to  $0$  do
11:     $j_\theta = [D_a \pi_\theta + \gamma(j'_{s'}, f_a \pi_\theta + j'_\theta)]|_\xi$ 
12:     $j_s = [D_s + D_a \pi_s + \gamma j'_{s'}(f_s + f_a \pi_\theta)]|_\xi$ 
13:  end for
14:  Apply gradient update using  $j_\theta^0$ 
15: end for
```

Experiments

Task	Dataset size	Behavioral cloning	GAIL	Ours
Hopper	4	50.57 ± 0.95	3614.22 ± 7.17	3669.53 ± 6.09
	11	1025.84 ± 266.86	3615.00 ± 4.32	3649.98 ± 12.36
	18	1949.09 ± 500.61	3600.70 ± 4.24	3661.78 ± 11.52
	25	3383.96 ± 657.61	3560.85 ± 3.09	3673.41 ± 7.73
Walker	4	32.18 ± 1.25	4877.98 ± 2848.37	6916.34 ± 115.20
	11	5946.81 ± 1733.73	6850.27 ± 91.48	7197.63 ± 38.34
	18	1263.82 ± 1347.74	6964.68 ± 46.30	7128.87 ± 141.98
	25	1599.36 ± 1456.59	6832.01 ± 254.64	7070.45 ± 30.68

Conclusion

- A model-based method for adversarial imitation learning
- Pros:
 - Requires fewer interactions with the environment
 - Enable using the partial derivatives of the discriminator when calculating the policy gradient
- Cons:
 - Requires learning a forward model, which could be difficult for some problems since forward model is also affected by distribution changing of the policy's data
 - Inaccurate forward model will lead to noisy gradients and will impede convergence
 - Experiment part in this paper is not very solid

Other take-home messages

- Discriminator network should be large ($\sim 2x$) in comparison to the policy network
- Discriminator network should be trained with a large l_r that slowly decays (to adapt to the changing distribution of the policy data)
- Adding noise to the expert data helps convergence (otherwise it is easy for discriminator to distinguish the expert)
- The discriminator holds valuable information and can be used such as a confidence measure for the policy's performance at inference time