End-to-End Differentiable Adversarial Imitation Learning

Nir Baram[1] Oron Anschel [1] Itai Caspi[1] Shie Mannor[1]

[1]Technion Institute of Technology, Israel.

ICML, 2017

Presenter: Xueying Bai

Outline

Outline

Task

- **Task**: Learning a policy which imitates the expert policy.
- **Imitation is needed for**:
  - **Automation**: when the expert is human
  - **Distillation**: when the expert is too expensive to run in realtime
  - **Initialization**: when using an expert policy as an initial solution
- **To be more specific**:
  Assume that trajectories $\{s_0, a_0, s_1, ...\}_N^{i=0}$ of an expert policy π_E are given. The goal is to train a new policy π which imitates π_E without access to the original reward signal r_E that was used by the expert.

Outline

Related Work

There are two main approaches to solve imitation problems.

- **Behavioral Cloning (BC)**: directly learns $p(a|s)$ in a supervised learning fashion. But only uses single $\{s_t, a_t\}$, which ignores current action's affect to future state distribution. Also requires a significant amount of expert data for training.
- **A Two-stage Imitation Algorithm**: First, recover a reward signal under which the expert is uniquely optimal.

$$E\left[\sum_t \gamma^t \widehat{r}(s_t, a_t)|\pi_E\right] \geq E\left[\sum_t \gamma^t \widehat{r}(s_t, a_t)|\pi\right], \forall \pi$$

Then train a policy that maximizes the discounted cumulative expected reward.

$$E_\pi R = E_\pi\left[\sum_t \gamma^t \widehat{r}_t\right]$$

But the reward only comes from the observation. Better to be designed by hand.

Outline

Motivation

- Use GAN to imitate an expert. GAN can alleviate problems in imitation learning such as sample complexity.
- Problem: when training stochastic policies, the presence of stochastic elements breaks the flow of information, thus prohibits the use of backpropagation.
- This paper presents a model-based imitation learning algorithm (MGAIL), in which information propagates fluently. Also a forward process is proposed to approximate the environments' dynamics.

Outline

Markov Decision Process

- An infinite-horizon discounted MDP is defined by the tuple $(S, A, P, r, \rho_0, \gamma)$:
- S: a set of states
  A: a set of actions
  $P : S \times A \times S \to [0, 1]$: transition probability distribution
  $r : (S \times A) \to R$: reward function
  ρ_0: distribution over initial states
  $\gamma \in (0, 1)$: discount factor
  $\pi : S \times A \to [0, 1]$: a stochastic policy
  $R(\pi)$: expected discounted reward of π, $E_\pi R = E_\pi \left[\sum_t \gamma^t \widehat{r_t} \right]$
  $\tau = \{s_0, a_0, s_1, a_1\}$: trajectory of states and actions

Outline

Imitation Learning

Learning control policies directly from expert demonstrations.

- Train a policy π to minimize some loss function $l(s, \pi(s))$, under the discounted state distribution encountered by the expert: $d_\pi(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p(s_t)$. The learned policy is:

$$\pi = \arg\min_{\pi \in \Pi} E_{s\sim d_\pi}\left[l(s, \pi(s))\right]$$

  Π denotes the class of all possible policies.
  Problem:the policy's prediction will affect future state distribution.
- Forward Training to train a non-stationary policy (π_t for each time). π_t is induced by $\pi_0, ...\pi_{t-1}$, with actual state distribution at each time.
  Problem: Impractical.
- Stochastic Mixing Iterative Learning (SMILe): train a stochastic stationary policy over several iterations.

$$\pi_t = \pi_{t-1} + \alpha(1-\alpha)^{t-1}(\widehat{\pi}_t - \pi_0)$$

  π_0: expert's initial state, $\widehat{\pi}_t$: trained policy induced by π_{t-1}

Outline

GAN

- Objective function:

$$\arg\min_{G} \arg\max_{D\in(0,1)} E_{x\sim p_E}[\log D(x)] + E_{z\sim p_z}[\log(1 - D(G(z)))]$$

p_E is expert input distribution, p_z is the noise distribution.

- Gradients:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D_{\theta_d}(x^i) + \log(1 - D_{\theta_d}(G(z^i)))]$$

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G_{\theta_g}(z^i)))$$

GAIL

A model-free GAN approach for policy imitation learning.

- Objective function:

$$\arg\min_{\pi} \arg\max_{D\in(0,1)} E_{\pi}[\log D(s,a)] + E_{\pi_E}[\log(1-D(s,a)] - \lambda H(\pi)$$

$$H(\pi) = E_{\pi}[-\log \pi(a|s)]$$

- Gradients:

  Because the generator π is now stochasitic,

$$E_{\pi}[\log D(s,a)] = E_{s\sim\rho_{\pi}(s)} E_{a\sim\pi(\cdot|s)}[\log D(s,a)]$$

  if π is deterministic,

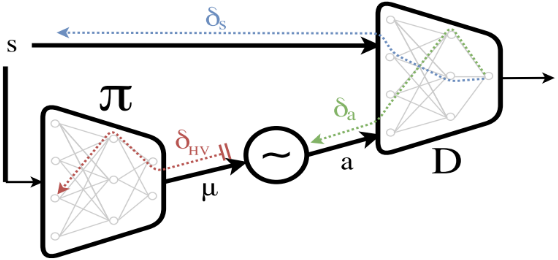
$$E_{\pi}[\log D(s,a)] = E_{s\sim\rho}[\log D(s,\pi(s))]$$

  So assume $\pi = \pi_{\theta}$, unknown how to differentiate the objective function w.r.t θ.

Score function methods

The method to obtain an unbiased gradient estimation.

- $$\nabla_\theta E_\pi[\log D(s,a)] \cong \widehat{E}_{\tau_i}[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)]$$
  $$Q(\widehat{s},\widehat{a}) = \widehat{E}_{\tau_i}[\log D(s,a)|s_0 = \widehat{s}, a_0 = \widehat{a}]$$
- Suffer from high variance.
- D will only give a score, G didn't access to the internal decision making logic of D.
- It's better to use the Jacobian of D when calculating θ



Outline

The Discriminator Network

For a good learner: what to inspect from the discriminator?

- $$D(s,a) = p(y|s,a), y \in \{\pi_E, \pi\}$$
- By Bayesian rule:
  $\varphi(s,a) = \frac{p(a|s,\pi_E)}{p(a|s,\pi)}$: policy likelihood ratio $\psi(s) = \frac{p(s|\pi_E)}{p(s|\pi)}$: state distribution likelihood ratio
  $$D(s,a) = \frac{1}{1 + \varphi(s,a) \cdot \psi(s)}$$
- A good learner should consider two effects: how its choice of actions stands against the expert? how it affects the future state distribution? Partial derivatives can reveal such information:
  $$\nabla_a D = -\frac{\varphi_a(s,a) \cdot \psi(s)}{(1 + \varphi(s,a) \cdot \psi(s))^2}$$
  $$\nabla_s D = -\frac{\varphi_s(s,a) \cdot \psi(s) + \varphi(s,a) \cdot \psi_s(s)}{(1 + \varphi(s,a) \cdot \psi(s))^2}$$

Outline

Coutinuous Action Distributions

Estimate the gradients of continuous stochastic elements by re-parametrization.

- Assume a stochastic policy with a Gaussian distribution:

$$\pi_\theta(a|s) \sim N(\mu_\theta(s), \sigma^2(s))$$

$$\pi_\theta(a|s) = \mu_\theta(s) + \xi\sigma_\theta(s), \xi \sim N(0,1)$$

- Mont-Carlo estimator of the derivative of the expected value of $D(s,a)$:

$$\nabla_\theta E_{\pi(a|s)}[\log D(s,a)] = E_{p(\xi)} \nabla_a D(s,a) \nabla_\theta \pi_\theta(a|s)$$
$$\cong \frac{1}{M} \sum_{i=1}^{M} \nabla_a D(s,a) \nabla_\theta \pi_\theta(a|s)|_{\xi=\xi_i}$$

Categorical Action Distributions

- Use softmax in the sampling procedure:

$$a_{soft\max} = \frac{\exp[\frac{1}{\tau}(g_i + \log \pi(a_i|s))]}{\sum_{j=1}^{k} \exp[\frac{1}{\tau}(g_i + \log \pi(a_i|s))]}$$

  $g_i \sim Gumbel(0, 1)$, τ is a hyper-parameter that trades bias with variance.

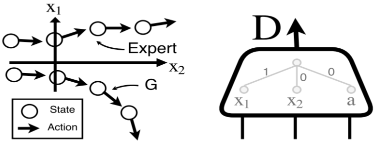
- To output action, apply *argmax* over $a_s oftmax$
- Use the continuous approximation in the backward pass:

$$\nabla_\theta a_{\arg\max} \approx \nabla_\theta a_{soft\max}$$

Outline

Backpropagating through a Forward model

- In the previous parts, only $\nabla_a D$ is used. However, $\nabla_s D$ is also important.

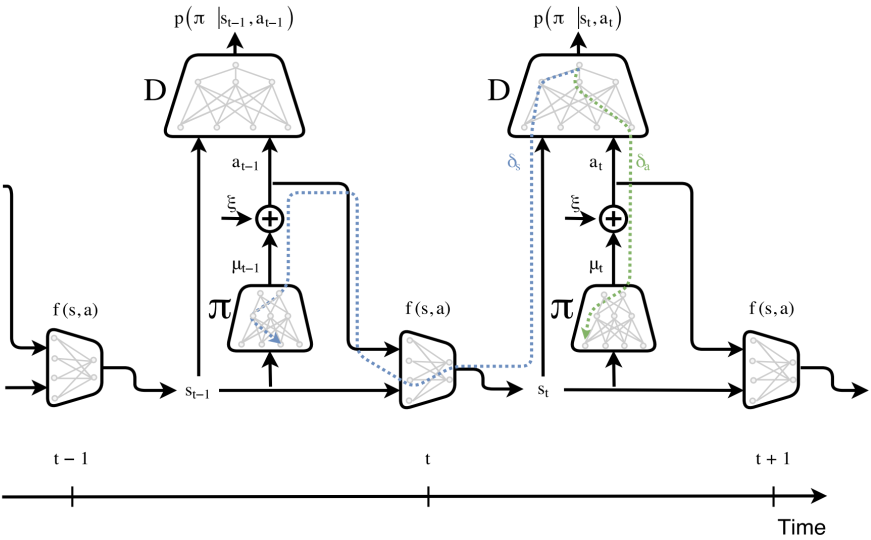


- Let $s_t = f(s_{t-1}, a_{t-1})$, f is the forward model. So the gradient of θ is:

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

$$\frac{\partial D}{\partial a}\frac{\partial a}{\partial \theta}\Big|_{a=a_t} + \frac{\partial D}{\partial s}\left(\frac{\partial f}{\partial s}\frac{\partial s}{\partial \theta}\Big|_{s=s_{t-1}} + \frac{\partial f}{\partial a}\frac{\partial a}{\partial \theta}\Big|_{a=a_{t-1}}\right).$$

Model Overview



Outline

MGAIL Algorithm

- For a multi-step transition, the objective function is:

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

- Gradient:

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

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

Algorithm

Algorithm 1 **Model-based Generative Adversarial Imitation Learning**

Input: Expert trajectories τ_E, experience buffer $\mathcal{B}$, initial policy and discriminator parameters θ_g, θ_d
for $trajectory = 0$ **to** ∞ **do**
  for $t = 0$ **to** T **do**
    Act on environment: $a = \pi(s, \xi; \theta_g)$
    Push (s, a, s') into $\mathcal{B}$
  end for
  train forward model f using $\mathcal{B}$
  train discriminator model D_{θ_d} using $\mathcal{B}$
  set: $j'_s = 0, j'_{\theta_g} = 0$
  for $t = T$ **down to** 0 **do**
    $j_{\theta_g} = [D_a \pi_{\theta_g} + \gamma(j'_{s'} f_a \pi_{\theta_g} + j'_{\theta_g})]\big|_{\xi}$
    $j_s = [D_s + D_a \pi_s + \gamma j'_{s'}(f_s + f_a \pi_{\theta_g})]\big|_{\xi}$
  end for
  Apply gradient update using $j^0_{\theta_g}$
end for

Data

- 3 discrete taskss (Cartpole, Mountain-Car, Acrobot), 5 continuous control tasks (Hopper, Walker, Half-Cheetah, Ant, and Humanoid)
- For each task, produce datasets with a different number of trajectories, where each trajectory is of length $N = 1000$.

Result

Task	Dataset size	Behavioral cloning	GAIL	MGAIL
Cartpole	1	72.02 ± 35.82	200.00 ± 0.00	200.00 ± 0.00
	4	169.18 ± 59.18	200.00 ± 0.00	200.00 ± 0.00
	7	188.60 ± 29.61	200.00 ± 0.00	200.00 ± 0.00
	10	177.19 ± 52.83	200.00 ± 0.00	200.00 ± 0.00
Mountain Car	1	-136.76 ± 34.44	-101.55 ± 10.32	-107.4 ± 10.89
	4	-133.25 ± 29.97	-101.35 ± 10.63	-100.23 ± 11.52
	7	-127.34 ± 29.15	-99.90 ± 7.97	-104.23 ± 14.31
	10	-123.14 ± 28.26	-100.83 ± 11.40	-99.25 ± 8.74
Acrobot	1	-130.60 ± 55.08	-77.26 ± 18.03	-85.65 ± 23.74
	4	-93.20 ± 35.58	-83.12 ± 23.31	-81.91 ± 17.41
	7	-96.92 ± 34.51	-82.56 ± 20.95	-80.74 ± 14.02
	10	-95.09 ± 33.33	-78.91 ± 15.76	-77.93 ± 14.78
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	[illegible]	[illegible]