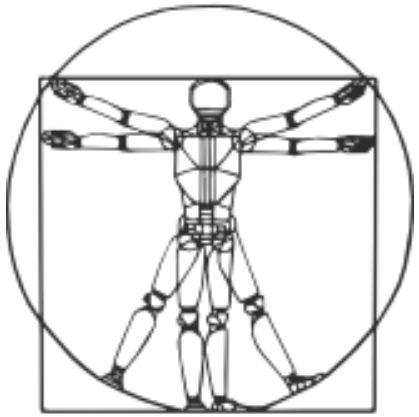


CSC2626
Imitation Learning for Robotics

Florian Shkurti

Week 8: Adversarial Imitation Learning

Today's agenda

- Adversarial optimization for imitation learning (GAIL)
- Adversarial optimization with multiple inferred behaviors (InfoGAIL)
- Model based adversarial optimization (MGAIL)

Acknowledgments
Today's slides are based on student presentations from 2019 by: Yeming Wen, Yin-Hung Chen , and Yuwen Xiong

Generative Adversarial Imitation Learning

Yeming Wen

2019-02-23

Backgroud

- **Definitions:** Action space $\mathcal{A}$ and sample space $\mathcal{S}$. Π is the set of all policies. Also assume $P(s'|s,a)$ is the dynamics model. In this paper, π_E denotes the expert policy.
- **Imitation Learning:** Learning to perform a task from expert demonstrations without querying the expert while training.
- **Behavioral cloning:** Its success depends on large amounts of data.
- **Inverse RL:** The paper adopts the maximum causal entropy IRL which fits a cost function c with the following problem.

$$\pi^* = \arg\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_\pi[c(s,a)]$$

$$\tilde{c} = \arg\max_{c \in \mathcal{C}} \mathbb{E}_{\pi^*}[c(s,a)]) - \mathbb{E}_{\pi_E}[c(s,a)]$$

where $H(\pi) = \mathbb{E}_\pi[-\log \pi(a|s)]$ is the entropy of the policy.

More on IRL

- The reason why we want to maximize the entropy is we want to make the least claim of the model while fitting the data.
- IRL learns a cost function that prioritizes entire trajectories.
- It doesn't have compounding error which occurs when the only fits single-timestep decisions, such as behavioral cloning.
- However, IRL is generally expensive because it requires reinforcement learning in the inner loop.
- Learning a cost function doesn't tell the learner how to act (policy).
- We hope to build a new algorithm based on IRL which can lead to an induced policy.

Formulation

- We first study the policies found by RL on costs learned by IRL on the largest possible set of cost functions $\mathcal{C} = \{c : \mathcal{S}X\mathcal{A} \rightarrow \mathbb{R}\}$.
- Also need to define a convex cost function regularizer $\psi : \mathbb{R}_{\mathcal{S}X\mathcal{A}} \rightarrow \mathbb{R}$, which turns out to be important in this paper.
- Re-write the Eq. 1 as the following:

$$IRL_{\psi}(\pi_E) = \arg\max_{c \in \mathcal{C}} -\psi(c) + (\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]) - \mathbb{E}_{\pi_E}[c(s,a)]$$

- Define $RL(c) = \arg\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]$.
  Let $\tilde{c} \in IRL_{\psi}(\pi_E)$. We are interested in characterizing the induced policy $RL(\tilde{c})$.

Derivations

- It is easier to characterize $RL(\tilde{c})$ if we transform optimization problems over policies into convex problems.
- So the paper introduces an occupancy measure $\rho_\pi : \mathcal{S}X\mathcal{A} \to \mathbb{R}$:

$$\rho_\pi(s,a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^y P(s_t = s|\pi) \tag{1}$$

  It can be interpreted as the distribution of state-action pairs when roll-out with policy π.
- There is an **one-to-one** correspondence between policy and occupancy measure. It also allows us to re-write the expected cost as

$$\mathbb{E}_\pi[c(s,a)] = \sum_{s,a} \rho_\pi(s,a)c(s,a) \tag{2}$$

Derivations

- Lemma 1: If we define

$$\hat{H}(\rho) = -\sum_{s,a} \rho(s,a) \log \left(\rho(s,a) / \sum_{a'} \rho(s,a')\right) \tag{3}$$

then we have $\hat{H}(\rho) = H(\pi_\rho)$ and $H(\pi) = \hat{H}(\rho_\pi)$. So we can represent the entropy of a policy π with the occupancy measure ρ_π.

- Lemma 2: If we define,

$$L(\pi, c) = -H(\pi) + \mathbb{E}_\pi[c(s,a)]$$
$$\hat{L}(\rho, c) = -\hat{H}(\rho) + \sum_{s,a} \rho(s,a) c(s,a)$$

then we have $L(\pi, c) = \hat{L}(\rho_\pi, c)$ and $\hat{L}(\rho, c) = L(\pi_\rho, c)$. The Lemma allows us to transform the problem from optimizing π to ρ.

Convex Conjugate

- Given a function f, it can be represented by the supremum of all affine functions that are majorized by f.
- For any given slope m, there may be many different constants b such that the affine function $\langle m, x \rangle - b$ is majorized by f. We only need the best such constant.
- That's what the convex conjugate f^* does. Given a slope m, f^* returns the best constant b such that $\langle m, x \rangle - b$ is majorized by f. Thus,

$$f^*(m) = \sup_x \langle m, x \rangle - f(x)$$

- Note that $f^{**} = f$.

Derivations

- By Lemma 2, if ψ is a constant regularizer and $\tilde{c} \in IRL_\psi(\pi_E)$ and $\tilde{\pi} \in RL(\tilde{c})$, then $\rho_{\tilde{\pi}} = \rho_{\pi_E}$.
- Furthermore, we can also get the main result of the paper

$$RL \circ IRL_\psi(\pi_E) = \arg\min_{\pi \in \Pi} -H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E}) \tag{4}$$

where ψ^* is the convex conjugate of ψ, which is defined as

$$\psi^*(m) = \sup_{x \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}} m^T x - \psi(x)$$

- It tells us that the ψ-regularized inverse RL seeks a policy whose occupancy measure is close to the expert's as measured by the convex function ψ^*.
- A good imitation learning algorithm boils down to a good choice of the regularizer ψ.

Occupancy Measure Matching

- As we showed previously, if ψ is a constant, then the resulting policy would have the same occupancy measures with expert at all states and actions.
- It is not practically useful because most of the occupancy measure of the expert values are exactly zero, due to the limited expert samples.
- Thus, exact occupancy measure matching will force the learned policy to never visit the unseen state-action pairs.
- If we restrict the class of cost function $\mathcal{C}$ to be convex and set the regularizer ψ to be the indicator function of the set $\mathcal{C}$. Then optimization problem in (6) can be written as

$$\min_{\pi} -H(\pi) + \max_{c \in \mathcal{C}} \mathbb{E}_{\pi}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \tag{5}$$

which is a entropy-regularized apprenticeship learning problem.

Apprenticeship Learning

- Policy gradient method can be used to update the parameterized policy π_θ to optimize the apprenticeship objective, Eq. 7.

$$\begin{aligned}\nabla_\theta \max_{c\in\mathcal{C}} \mathbb{E}_{\pi_\theta}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] &= \nabla_\theta \mathbb{E}_{\pi_\theta}[c^*(s,a)] \\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_{c^*}(s,a)]\end{aligned}$$

where

$$c^* = \arg\max_{c\in\mathcal{C}} \mathbb{E}_{\pi_\theta}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \tag{6}$$

$$Q_{c^*}(\bar{s},\bar{a}) = \mathbb{E}_{\pi_\theta}[c^*(\bar{s},\bar{a})|s_0=\bar{s}, a_0=\bar{a}] \tag{7}$$

- Fit c_i^* as defined above. Analytical solution is feasible if $\mathcal{C}$ is restricted to Convex or Linear cost classes.
- Given the c_i^*, compute the policy gradient and take a TRPO step to produce $\pi_{\theta_{i+1}}$.

GAIL

- Apprenticeship learning via TRPO is tractable in large environments but is incapable of exactly matching occupancy measures without careful tuning due to the restrictive cost classes $\mathcal{C}$.
- Constant regularizer ψ leads to exact matching but is intractable in large environments. Thus, GAIL is proposed to combine the best of both methods.

$$\psi_{GA}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases}$$

where

$$g(x) = \begin{cases} -x - \log\left(1 - e^{x}\right) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

GAIL

- The GAIL regularizer ψ_{GA} places low penalty on cost functions c that assign an amount of negative cost to expert state-action pairs; It havily penalizes c if it assigns large cost to the expert.
- ψ_{GA} is an average over expert data so it can adjust to arbitrary expert datasets.
- In comparison, if ψ is an indicator function (Apprenticeship Learning), then it's always fixed.
- Another property of ψ_{GA} is its convex conjugate $\psi^*_{GA}(\rho_\pi - \rho_{\pi_E})$ can be derived in the following form:

$$\max_{D \in (0,1)^{\mathcal{S} X \mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] \tag{8}$$

- It can be interpreted to find a discriminator that distinguishes trajectory between learned policy and expert policy. t

GAIL

- Combining with the main result Eq. (6) in the paper,

$$RL \circ IRL_{\psi}(\pi_E) = \arg\min_{\pi \in \Pi} -H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E})$$

The imitation learning problem is equivalent to find a sddle point (π, D) of the expression

$$\mathbb{E}_\pi[\log(D(s,a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi) \tag{9}$$

- In terms of implementation, we just need to fit a parameterized policy π_θ with weights θ and a discriminator network $D_w : \mathcal{S}X\mathcal{A} \to (0,1)$ with weights w.
- Update D_w with Adam and update π_θ with TRPO iteratively.

Algorithm

Algorithm 1 Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters θ_0, w_0
2: **for** $i = 0, 1, 2, \ldots$ **do**
3: Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4: Update the discriminator parameters from w_i to w_{i+1} with the gradient

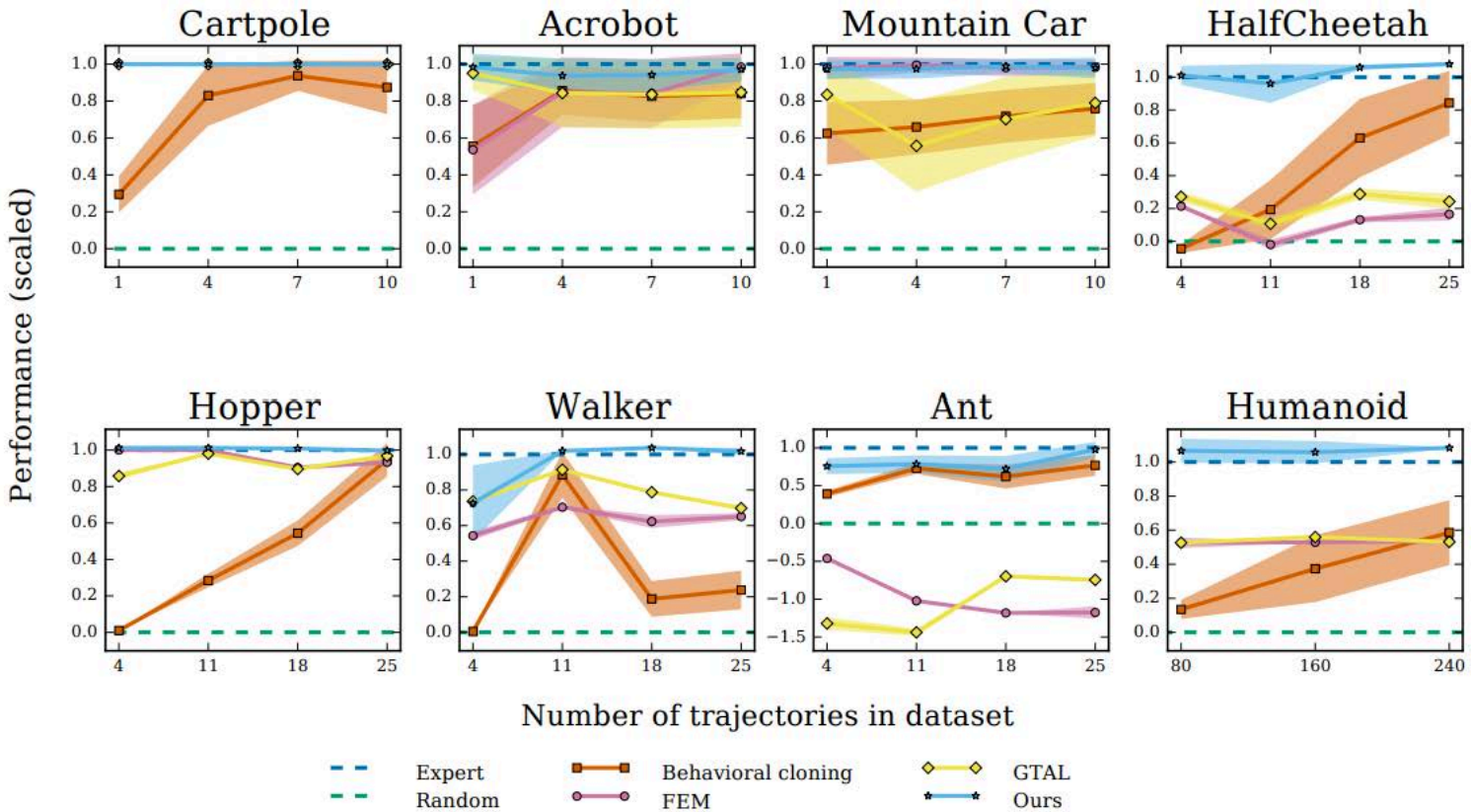
$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{17}$$

5: Take a policy step from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

Results


Cartpole
Acrobot
Mountain Car
HalfCheetah
Hopper
Walker
Ant
Humanoid
Performance (scaled)
Number of trajectories in dataset
Expert
Random
Behavioral cloning
FEM
GTAL
Ours

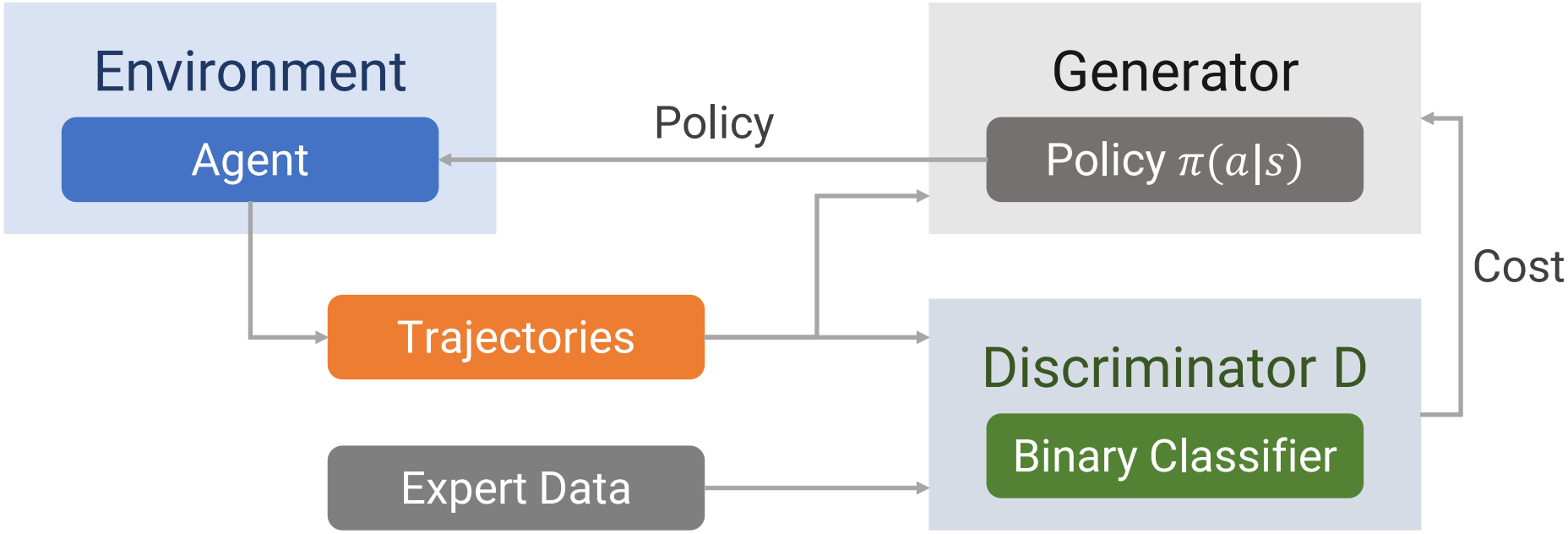
InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations

Presenter: Yin-Hung Chen

Motivation

GAIL

A generator producing a policy π competes with a discriminator distinguishing π and the expert.

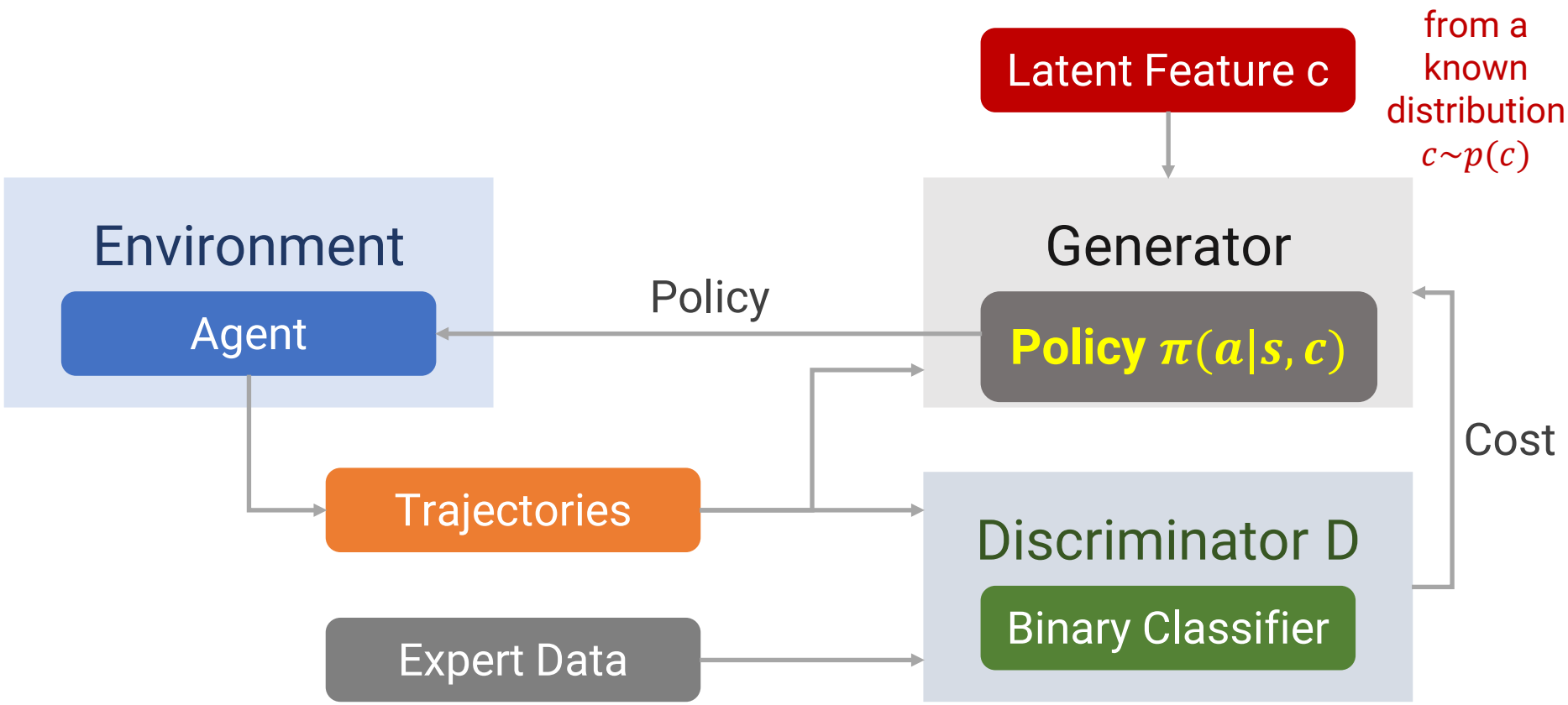


Drawbacks of GAIL

- Expert demonstrations can show significant **variability**.
- The observations might have been sampled from **different experts with different skills and habits**.
- **External latent factors** of variation are not explicitly captured by GAIL, but they can significantly affect the observed behaviors.

InfoGAIL

Modified GAIL



Objective Function

GAIL:

$$\min_{\pi} \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi}[logD(s,a)] + \mathbb{E}_{\pi_E}\big[log\big(1 - D(s,a)\big)\big] - \lambda H(\pi)$$

where π is learner policy, and π_E is expert policy.

InfoGAIL:

- Discriminator: same with GAIL
- Generator: simply introducing latent factor c into $\pi \rightarrow \pi(a|s,c)$
  However, applying GAIL to $\pi(a|s,c)$ could simply ignore c and fail to separate different expert behaviors → **adding more constraints over c**

Constraints over Latent Features

There should be high mutual information between the latent factor c and learner trajectory τ.

$$I(c;\tau) = \sum_{\tau} p(\tau) \sum_{c} p(c|\tau) log_2 \frac{p(c|\tau)}{p(c)}$$

Independence between c and trajectory tau:

$$p(c|\tau) = \frac{p(c)p(\tau)}{p(\tau)}, \qquad \frac{p(c|\tau)}{p(c)} = 1, \qquad log_2 \frac{p(c|\tau)}{p(c)} = 0$$

Maximizing mutual information $I(c;\tau)$
$\rightarrow$ hard to maximize directly as it requires the posterior $P(c|\tau)$
$\rightarrow$ using $Q(c|\tau)$ to estimate $P(c|\tau)$

Constraints over Latent Features

Introducing the lower bound $L_I(\pi, Q)$ of $I(c;\tau)$

$$
\begin{aligned}
& I(c;\tau) \\
& = H(c) - H(c|\tau) \\
& = \mathbb{E}_{a\sim\pi(\cdot|s,c)}\left[\mathbb{E}_{c'\sim P(c|\tau)}[logP(c'|\tau)]\right] + H(c) \\
& = \mathbb{E}_{a\sim\pi(\cdot|s,c)}\left[D_{KL}\big(P(\cdot\,|\tau) \parallel Q(\cdot\,|\tau)\big) + \mathbb{E}_{c'\sim P(c|\tau)}[logQ(c'|\tau)]\right] + H(c) \\
& \geq \mathbb{E}_{a\sim\pi(\cdot|s,\,c)}\left[\mathbb{E}_{c'\sim P(c|\tau)}[logQ(c'|\tau)]\right] + H(c) \\
& = \mathbb{E}_{c\sim P(c),a\sim\pi(\cdot|s,\,c)}[logQ(c|\tau)] + H(c) \\
& = L_I(\pi, Q)
\end{aligned}
$$

Constraints over Latent Features

There should be high mutual information between the latent factor c and learner trajectory τ.

Maximizing mutual information $I(c;\tau)$
$\rightarrow$ hard to maximize directly as it requires the posterior $P(c|\tau)$
$\rightarrow$ using $Q(c|\tau)$ to estimate $P(c|\tau)$

Maximizing $I(c;\tau)$ through maximize the lower bound $L_I(\pi, Q)$

Objective Function

GAIL:

$$\min_{\pi} \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi}[logD(s,a)] + \mathbb{E}_{\pi_E}\left[\log\left(1 - D(s,a)\right)\right] - \lambda H(\pi)$$

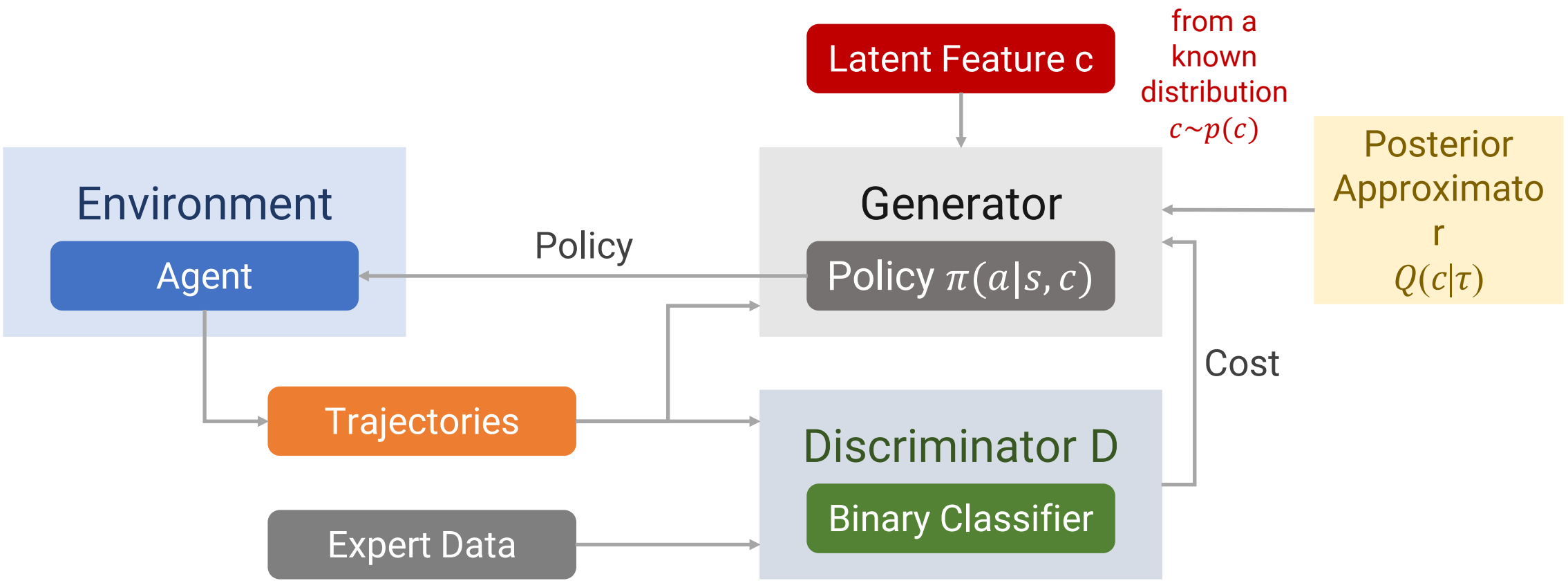
where π is learner policy, and π_E is expert policy.

InfoGAIL:

$$\min_{\pi,Q} \max_{D} \mathbb{E}_{\pi}[logD(s,a)] + \mathbb{E}_{\pi_E}\left[\log\left(1 - D(s,a)\right)\right] - \lambda_1 L_I(\pi, Q) - \lambda_2 H(\pi)$$

where $\lambda_1 > 0$ and $\lambda_2 > 0$.

InfoGAIL


Latent Feature c
from a known distribution $c \sim p(c)$
Posterior Approximator $Q(c|\tau)$
Environment
Agent
Policy
Generator
Policy $\pi(a|s,c)$
Cost
Trajectories
Discriminator D
Binary Classifier
Expert Data

Algorithm 1 InfoGAIL

Input: Initial parameters of policy, discriminator and posterior approximation $\theta_0, \omega_0, \psi_0$; expert trajectories $\tau_E \sim \pi_E$ containing state-action pairs.
Output: Learned policy π_θ
for $i = 0, 1, 2, ...$ **do**
Sample a batch of latent codes: $c_i \sim p(c)$
Sample trajectories: $\tau_i \sim \pi_{\theta_i}(c_i)$, with the latent code fixed during each rollout.
Sample state-action pairs $\chi_i \sim \tau_i$ and $\chi_E \sim \tau_E$ with same batch size.
Update ω_i to ω_{i+1} by ascending with gradients

$$\Delta_{\omega_i} = \hat{\mathbb{E}}_{\chi_i}[\nabla_{\omega_i} \log D_{\omega_i}(s, a)] + \hat{\mathbb{E}}_{\chi_E}[\nabla_{\omega_i} \log(1 - D_{\omega_i}(s, a))]$$

Update ψ_i to ψ_{i+1} by descending with gradients

$$\Delta_{\psi_i} = -\lambda_1 \hat{\mathbb{E}}_{\chi_i}[\nabla_{\psi_i} \log Q_{\psi_i}(c|s, a)]$$

Take a policy step from θ_i to θ_{i+1}, using the TRPO update rule with the following objective:

$$\hat{\mathbb{E}}_{\chi_i}[\log D_{\omega_{i+1}}(s, a)] - \lambda_1 L_I(\pi_{\theta_i}, Q_{\psi_{i+1}}) - \lambda_2 H(\pi_{\theta_i})$$

end for

Additional Optimization

Reward Augmentation

If the expert is performing sub-optimally, then any policy trained under the recovered rewards will be also suboptimal.

Reward augmentation: providing **additional incentives to incorporate prior knowledge** to the agent without interfering with the imitation learning process.

$\rightarrow$ specifying a **surrogate state-based reward** $\boldsymbol{\eta(\pi_\theta) = \mathbb{E}_{s\sim\pi_\theta}[r(s)]}$ that reflects our bias over the desired agent's behavior.

$$\min_{\theta,\psi}\max_{\omega}\mathbb{E}_{\pi_\theta}[logD_\omega(s,a)]+\mathbb{E}_{\pi_E}\left[\log\left(1-D_\omega(s,a)\right)\right]-\lambda_0\eta(\pi_\theta)-\lambda_1 L_I\left(\pi_\theta,Q_\psi\right)-\lambda_2 H(\pi_\theta)$$

where $\lambda_0 > 0$.

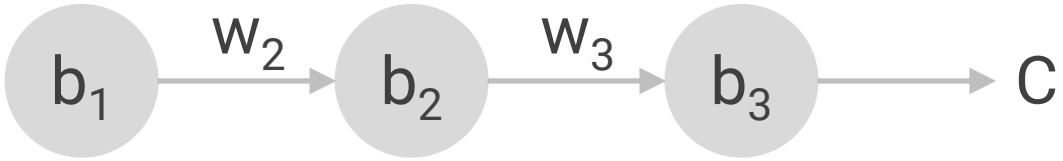
Improved Optimization

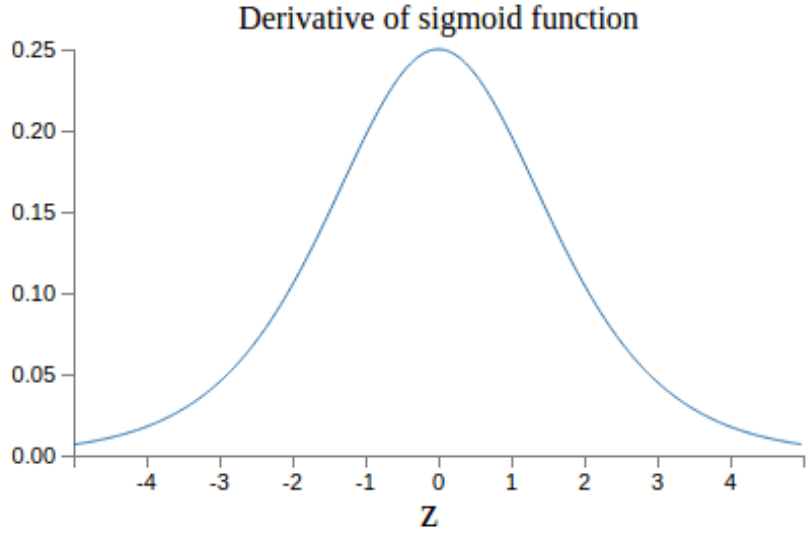
The traditional GAN objective suffers from vanishing gradient and mode collapse problems.

Vanishing gradient

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial y_3}\frac{\partial y_3}{\partial z_3}\frac{\partial z_3}{\partial x_3}\frac{\partial x_3}{\partial z_2}\frac{\partial z_2}{\partial x_2}\frac{\partial x_2}{\partial z_1}\frac{\partial z_1}{\partial b_1}$$

$$= \frac{\partial C}{\partial y_3}\sigma'(z_3)w_3\sigma'(z_2)w_2\sigma'(z_1)$$





Improved Optimization

The traditional GAN objective suffers from vanishing gradient and mode collapse problems.

Mode collapse: generator tends to produce the same type of data
→ generator yields the same G(z) for different z

Improved Optimization

The traditional GAN objective suffers from vanishing gradient and mode collapse problems.

→ using the Wasserstein GAN (WGAN)

$$\min_{\theta,\psi} \max_{\omega} \mathbb{E}_{\pi_\theta}[D_\omega(s,a)] - \mathbb{E}_{\pi_E}[D_\omega(s,a)] - \lambda_0 \eta(\pi_\theta) - \lambda_1 L_I\left(\pi_\theta, Q_\psi\right) - \lambda_2 H(\pi_\theta)$$

Algorithm 2 InfoGAIL with extensions

Input: Expert trajectories $\tau_E \sim \pi_E$; initial policy, discriminator and posterior parameters $\theta_0, \omega_0, \psi_0$; replay buffer $B = \varnothing$;
Output: Learned policy π_θ
for $i = 0, 1, 2, ...$ **do**
Sample a batch of latent codes: $c_i \sim P(c)$
Sample trajectories: $\tau_i \sim \pi_{\theta_i}(c_i)$, with the latent code fixed during each rollout.
Update the replay buffer: $B \leftarrow B \cup \tau_i$.
Sample $\chi_i \sim B$ and $\chi_E \sim \tau_E$ with same batch size.
Update ω_i to ω_{i+1} by ascending with gradients

$$\Delta_{\omega_i} = \hat{\mathbb{E}}_{\chi_i}[\nabla_{\omega_i} D_{\omega_i}(s, a)] - \hat{\mathbb{E}}_{\chi_E}[\nabla_{\omega_i} D_{\omega_i}(s, a)]$$

Clip the weights of ω_{i+1} to $[-0.01, 0.01]$.
Update ψ_i to ψ_{i+1} by descending with gradients

$$\Delta_{\psi_i} = -\lambda_1 \hat{\mathbb{E}}_{\chi_i}[\nabla_{\psi_i} \log Q_{\psi_i}(c|s, a)]$$

Take a policy step from θ_i to θ_{i+1}, using the TRPO update rule with the following objective (without reward augmentation):

$$\hat{\mathbb{E}}_{\chi_i}[D_{\omega_{i+1}}(s, a)] - \lambda_1 L_I(\pi_{\theta_i}, Q_{\psi_{i+1}}) - \lambda_2 H(\pi_{\theta_i})$$

or (with reward augmentation):

$$\hat{\mathbb{E}}_{\chi_i}[D_{\omega_{i+1}}(s, a)] - \lambda_0 \eta(\pi_{\theta_i}) - \lambda_1 L_I(\pi_{\theta_i}, Q_{\psi_{i+1}}) - \lambda_2 H(\pi_{\theta_i})$$

end for

Experiments

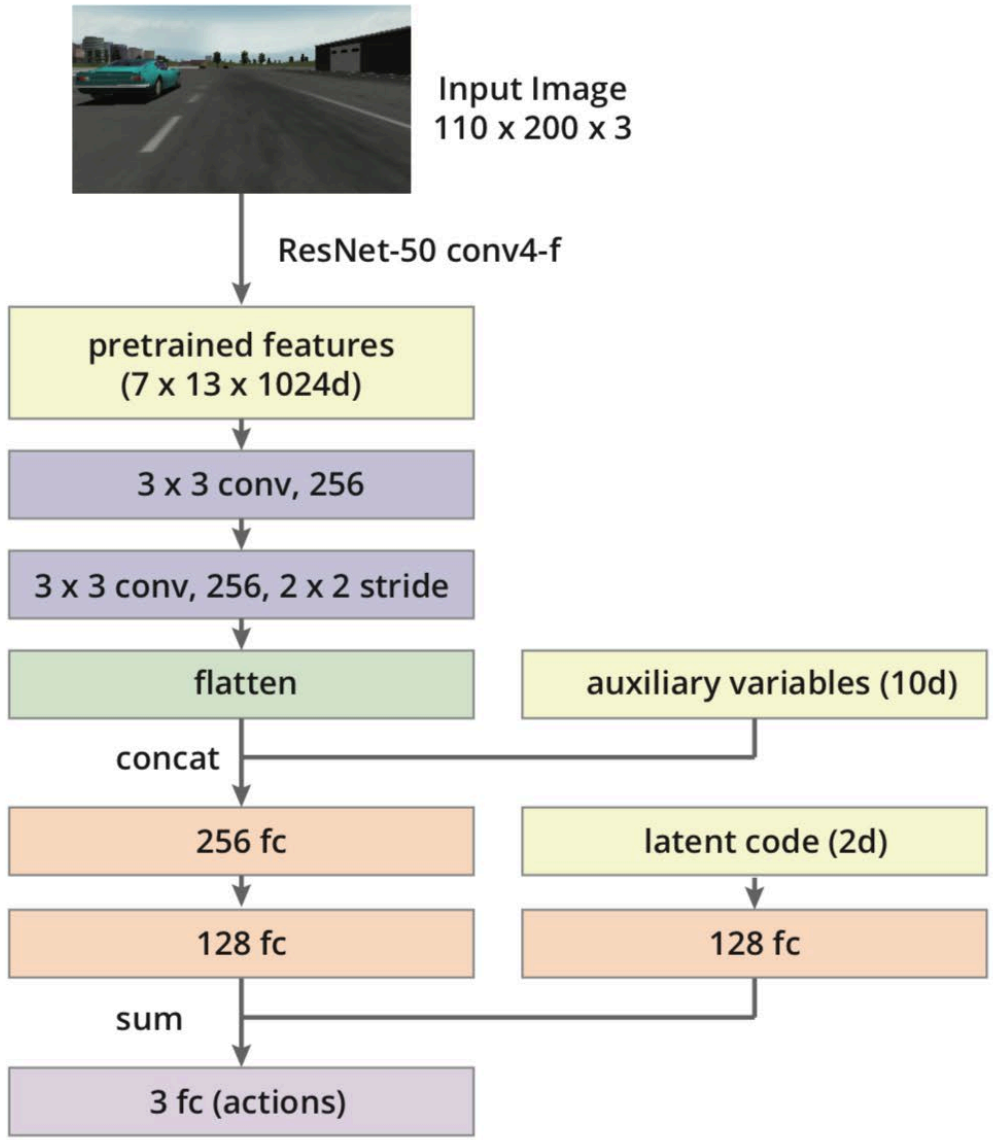
Learning to Distinguish Trajectories

- The observations at time t are positions from t − 4 to t.
- The latent code is a one-hot encoded vector with 3 dimensions and a uniform prior.

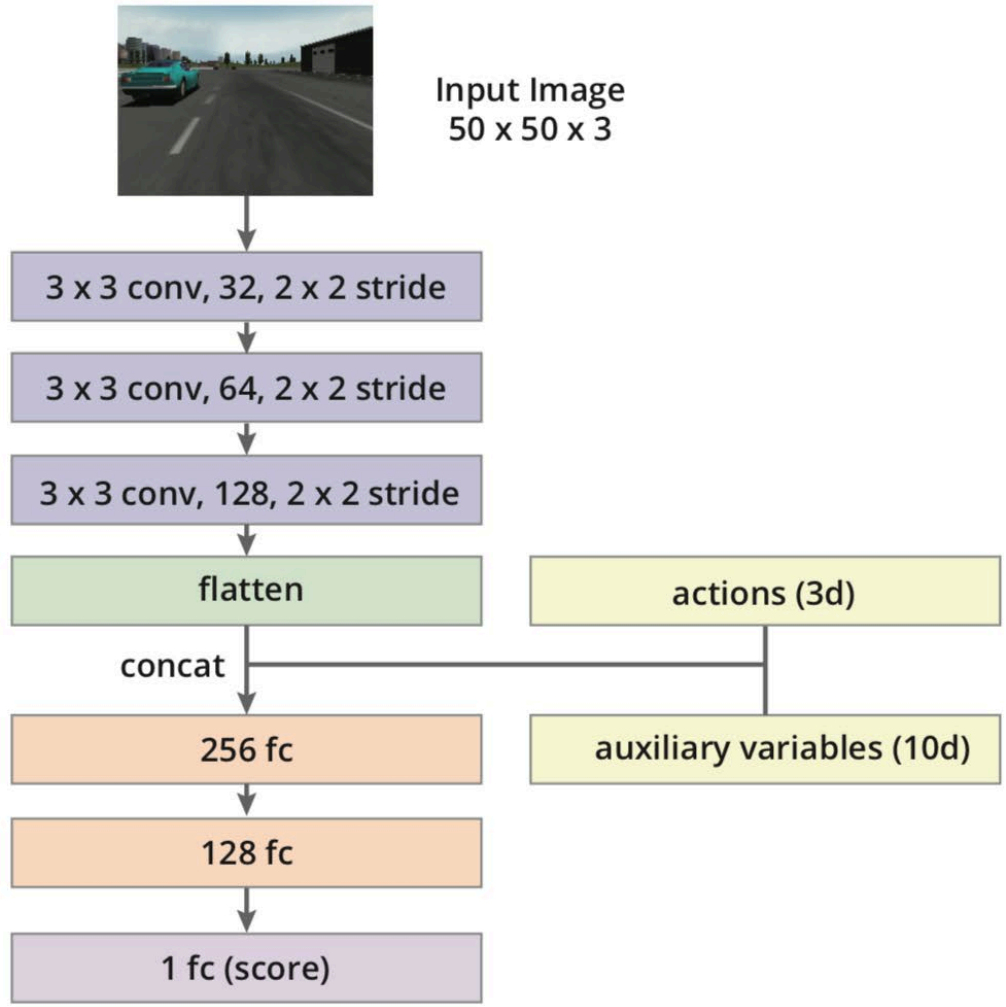
(a) Expert (b) Behavior cloning (c) GAIL (d) Ours

Self-driving car in the TORCS Environment

- The demonstrations collected by manually driving
- Three-dimensional continuous action composed of *steering*, *acceleration*, and *braking*
- Raw visual inputs as the only external inputs for the state
- Auxiliary information as internal input, including velocity at time *t*, actions at time $t - 1$ and t − 2, and damage of the car
- Pre-trained ResNet on ImageNet



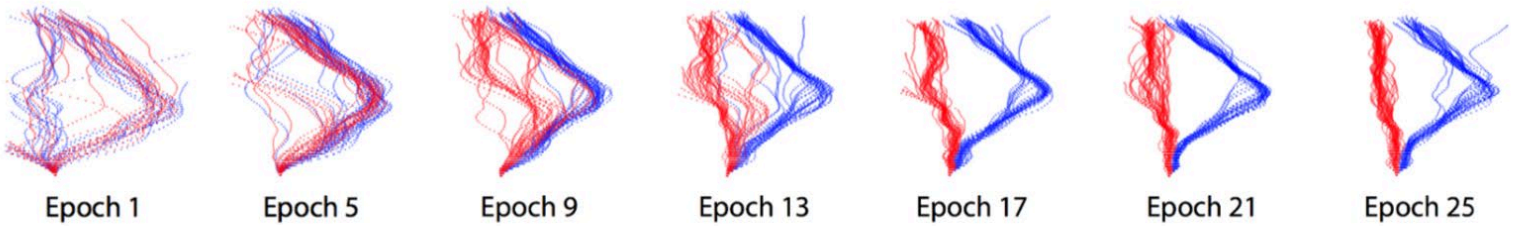
(a) Network architecture for the policy/generator π_θ.

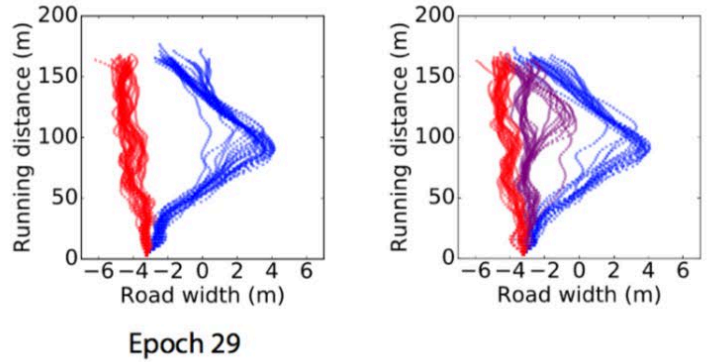


(b) Network architecture for the discriminator D_ω.

Performance

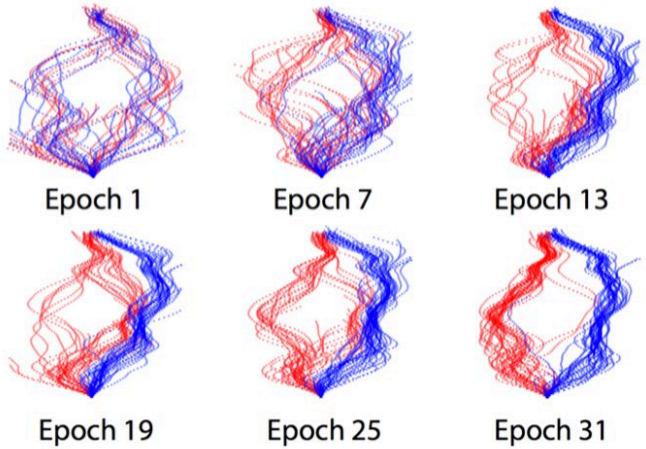
Turn
[0, 1] corresponds to using the inside lane (blue lines), while [1, 0] corresponds to the outside lane (red lines).

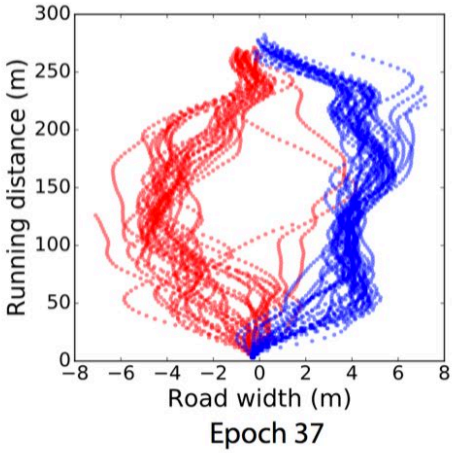




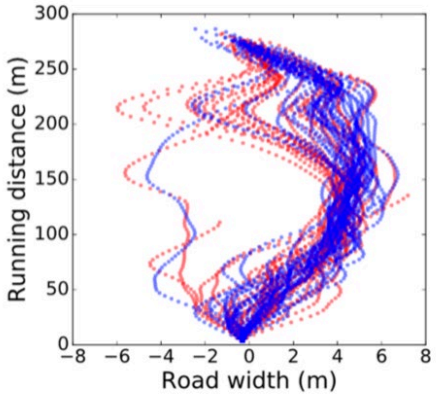
Performance

Pass
[0, 1] corresponds to passing from right (red lines), while [1, 0] corresponds to passing from left (blue lines).





InfoGAIL



GAIL

Performance

- Classification accuracies of $Q(c|\tau)$
- Reward augmentation encouraging the car to drive faster

Table 1: Classification accuracies for *pass*.

Method	Accuracy
Chance	50%
K-means	55.4%
PCA	61.7%
InfoGAIL (Ours)	**81.9%**
SVM	85.8%
CNN	**90.8%**

Table 2: Average rollout distances.

Method	Avg. rollout distance
Behavior Cloning	701.83
GAIL	914.45
InfoGAIL \ RB	1031.13
InfoGAIL \ RA	1123.89
InfoGAIL \ WGAN	1177.72
InfoGAIL (Ours)	**1226.68**
Human	1203.51

Model-based Adversarial Imitation Learning

Nir Baram, Oron Anschel, Shie Mannor

Presented by Yuwen Xiong, Mar 1st

Recap: GAIL algorithm

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

- 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

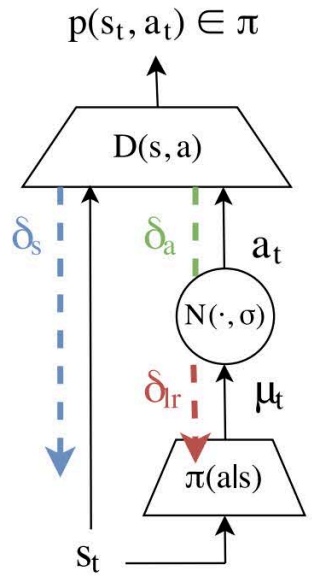
Motivation

- 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.

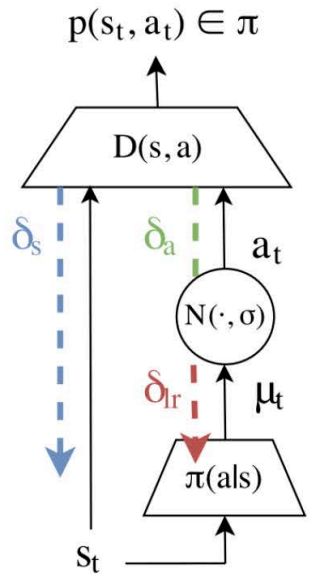


Algorithm - overview

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:

$$D(s,a) = p(\pi|s,a) = \frac{p(s,a|\pi)p(\pi)}{p(s,a)}$$

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

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:

$$D(s,a) = p(\pi|s,a) = \frac{p(s,a|\pi)p(\pi)}{p(s,a)}$$

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

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

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

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

Algorithm - re-parameterization of distribution

Assuming the policy is given by

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

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

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

We can rewrite it to

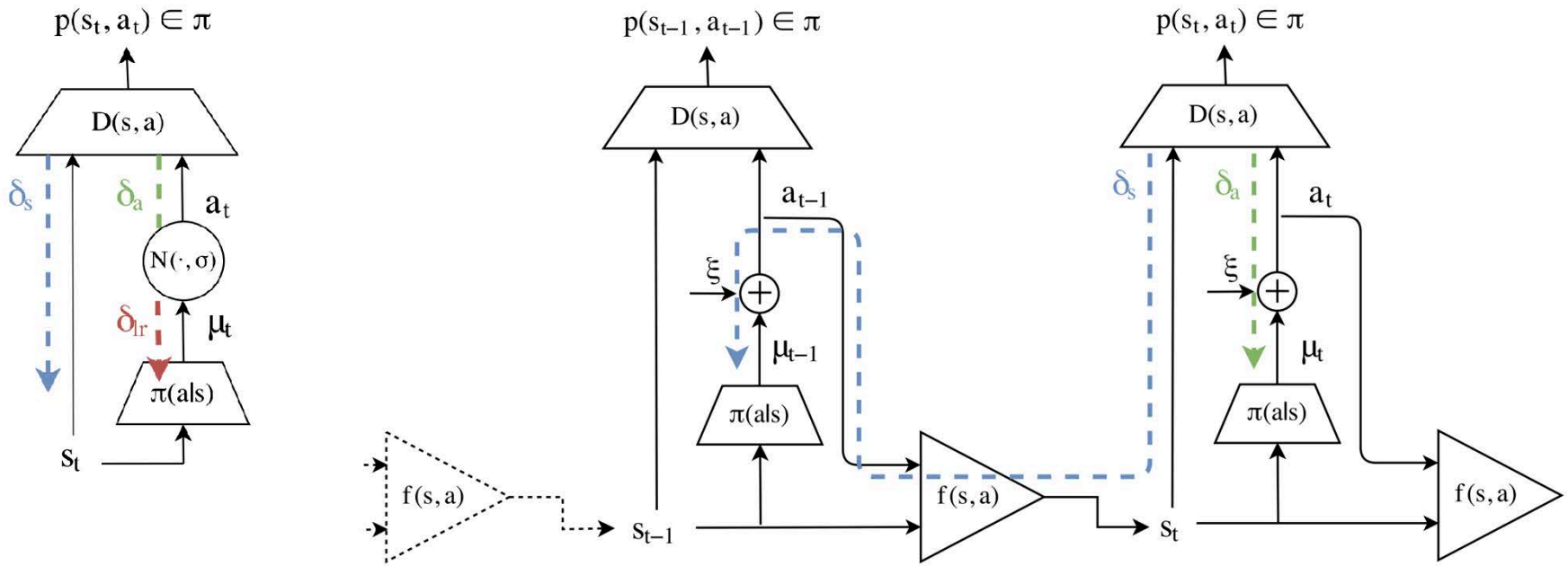
$$\pi_\theta(a|s) = \mu_\theta(s) + \xi\sigma_\theta(s), \text{ where } \xi \sim \mathcal{N}(0,1)$$

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

$$\nabla_\theta \mathbb{E}_{\pi(a|s)} D(s,a) = \mathbb{E}_{\rho(\xi)} \nabla_a D(a,s) \nabla_\theta \pi_\theta(a|s)$$

$$\cong \frac{1}{M} \sum_{i=1}^{M} \nabla_a D(s,a) \nabla_\theta \pi_\theta(a|s) \Big|_{\xi=\xi_i}$$

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:

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

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

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

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

Algorithm

Algorithm 1 Model-based Adversarial Imitation Learning

Given empty experience buffer $\mathcal{B}$
for $trajectory = 0$ **to** ∞ **do**
  for $t = 0$ **to** T **do**
    Act on environment: $a = \pi(s, \xi; \theta)$
    Push (s, a, s') into $\mathcal{B}$
  end for
  train forward model f using $\mathcal{B}$
  train discriminator model D using $\mathcal{B}$
  set: $j'_s = 0, j'_\theta = 0$
  for $t = T$ **down to** 0 **do**
    $j_\theta = [D_a \pi_\theta + \gamma(j'_{s'} f_a \pi_\theta + j'_\theta)]\big|_\xi$
    $j_s = [D_s + D_a \pi_s + \gamma j'_{s'}(f_s + f_a \pi_\theta)]\big|_\xi$
  end for
  Apply gradient update using j^0_θ
end for

Experiments

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

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 (~ 2x) in comparison to the policy network
- Discriminator network should be trained with a large lr 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