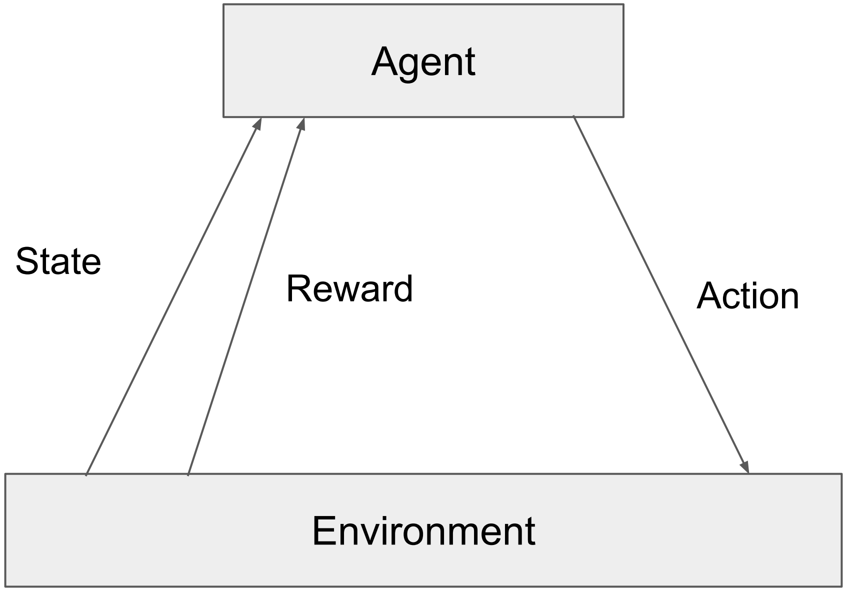
Generative Adversarial Imitation Learning

Jonathan Ho, Stefano Ermon - NIPS 2016

Presenter: Michael Park

Preliminaries: Reinforcement Learning



Goal: Find policy π that maximize rewards

Preliminaries: Imitation Learning

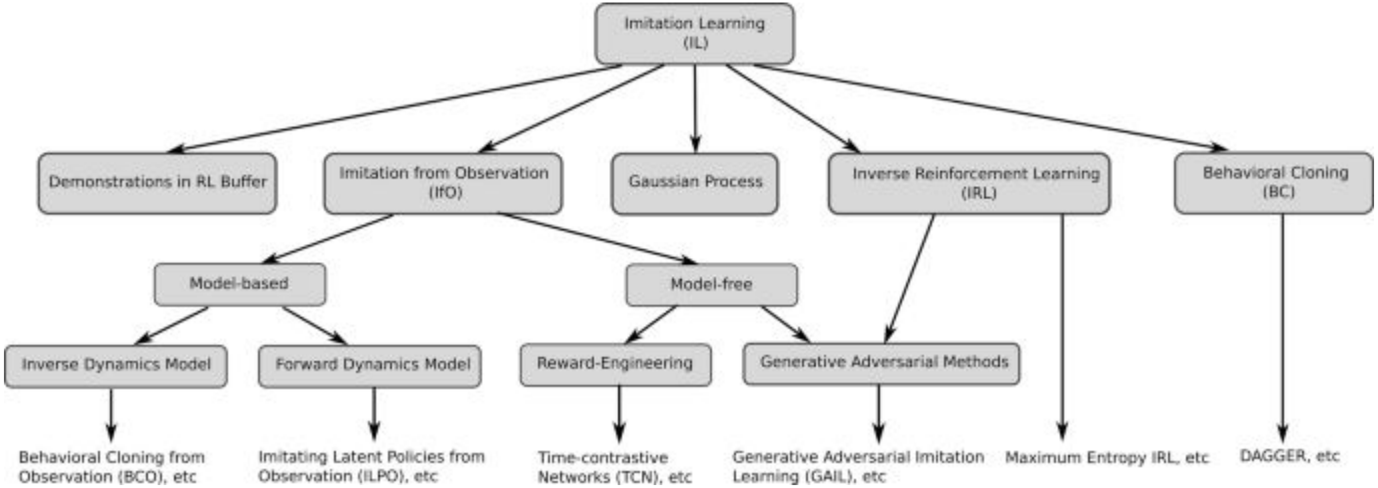
Learning from Demonstrations

- Expert provides a set of **demonstration trajectories**: a sequence of states and actions

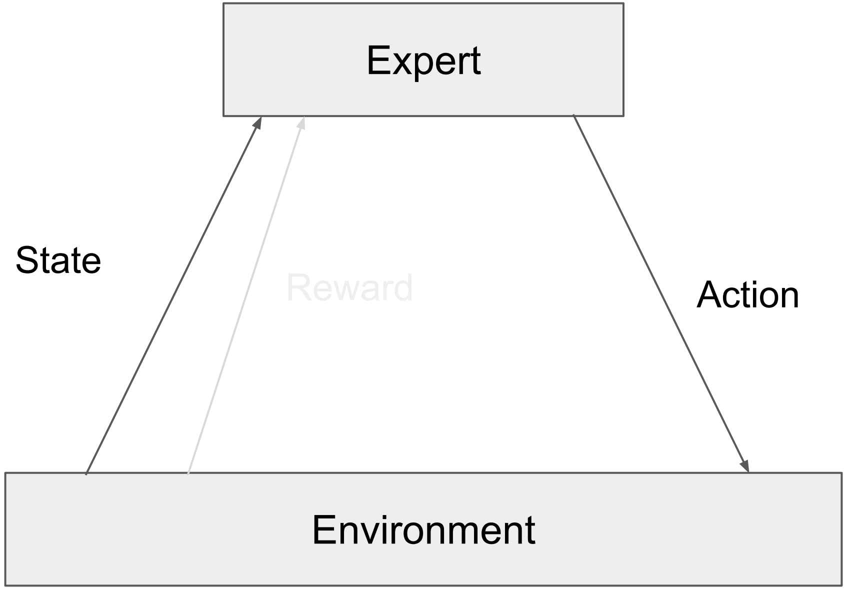
$$\{\langle s^1_1, a^1_1, s^1_2, a^1_2, \ldots, s^1_t, a^1_t\rangle, \langle s^2_1, a^2_1, s^2_2, a^2_2, \ldots, s^2_t, a^2_t\rangle, \ldots\}$$

- Imitation Learning is useful when it is easier for the expert to demonstrate the desired behavior rather than:
  - Specifying a **reward** that would generate such behavior (rewards are hard to define),
  - Specifying the desired **policy** directly

Preliminaries: Imitation Learning



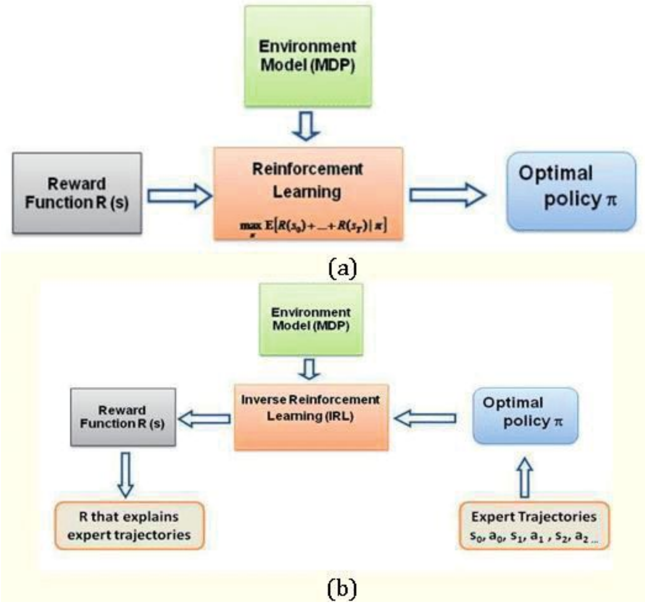
Preliminaries: Inverse Reinforcement Learning (IRL)



Goal: Find reward function that expert is implicitly optimizing

Preliminaries: Inverse Reinforcement Learning (IRL)

- Learns the **reward function** from expert trajectories
  that **prioritizes** entire trajectories over others, then derives the optimal policy
- Expensive to run (**inner loop has an RL iteration**)
- Indirectly learns optimal policy from the learned reward function (**using RL**)



Source

Proposed Model: **Generative Adversarial Imitation Learning** (GAIL)

Contributions

- **Directly extracting a policy** from data as if it were obtained by RL + IRL
- **Bypassing** any intermediate IRL step
- Draws an analogy between **imitation learning** and generative adversarial networks (GAN)
- Derive a model-free imitation learning algorithm with significant performance improvement and low sample & computational complexity

Inverse Reinforcement Learning

Maximum causal Entropy IRL (MaxEnt)

$$\underset{c\in\mathcal{C}}{\text{maximize}}\left(\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)]$$

$$\text{where } H(\pi) \triangleq \mathbb{E}_{\pi}[-\log \pi(a|s)]$$

- Try to find a cost function $c \in C$ that assigns **low cost** to the expert policy π_E and **high cost** to other policies π
- Using RL procedure, we can find the expert policy based on the cost c

$$\mathrm{RL}(c) = \underset{\pi\in\Pi}{\arg\min} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]$$

- Inner loop has RL; thus, **slow**

Generative Adversarial Imitation Learning: Proposed Framework

- Use the largest possible set of cost functions

$$\mathcal{C} = \mathbb{R}^{\mathcal{S}\times\mathcal{A}} = \{c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}\}$$

- Use Gaussian processes or **neural networks** to find the best cost function c among the large cost function class C
- To avoid overfitting, we use a "convex" regularizer for the cost function

$$\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R} \cup \infty$$

- With ψ, IRL procedure can be written as

$$\mathrm{IRL}_\psi(\pi_E) = \arg\max_{c\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}} -\psi(c) + \left[\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_\pi[c(s,a)]\right] - \mathbb{E}_{\pi_E}[c(s,a)]$$

- Let $\tilde{c} \in \mathrm{IRL}_\psi(\pi_E)$
- we are interested in π given by $\mathrm{RL}(\tilde{c}) = \pi$
- $\mathrm{RL}(\ \mathrm{IRL}_\psi(\pi_E)\ ) = \pi$

Generative Adversarial Imitation Learning: Occupancy Measure

- For a policy $\pi \in \Pi$, define its occupancy measure $\boldsymbol{\rho_\pi} : S \times A \rightarrow \mathbb{R}$ as

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

- In words, occupancy measure $\boldsymbol{\rho_\pi}$ is the distribution of state-action pairs with policy $\boldsymbol{\pi}$

$$\mathbb{E}_\pi[c(s,a)] = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t c(s_t,a_t)] = \sum_{s,a}\rho_\pi(s,a)c(s,a)$$

- The set of valid occupancy measure can be written as:

$$\mathcal{D} = \{\rho : \rho \geq 0 \& \sum_a \rho(s,a) = p_0(s) + \gamma\sum_{s',a} P(s|s',a)\rho(s',a) \forall s \in \mathcal{S}\}$$

- Note that there is 1-1 correspondence between $\boldsymbol{\Pi}$ and $\boldsymbol{D}$
- $\boldsymbol{\pi_\rho}$ to denote the unique policy for an occupancy measure $\boldsymbol{\rho}$

Generative Adversarial Imitation Learning: Convex Conjugate

- For a function f: $\mathbb{R}^{SxA} \rightarrow \mathbb{R} \cup \infty$, its convex conjugate f^*: $\mathbb{R}^{SxA} \rightarrow \mathbb{R} \cup \infty$ is

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

- Then, RL($\tilde{c}$) can be written as

$$\mathrm{RL} \odot \mathrm{IRL}_\psi(\pi_E) = \arg\min_{\pi \in \Pi} -H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E})$$

Generative Adversarial Imitation Learning: Overview

$$\mathrm{RL}(c) = \underset{\pi \in \Pi}{\arg\min} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)]$$

$$\mathrm{IRL}_{\psi}(\pi_E) = \arg \max_{c \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}} -\psi(c) + \left[\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right] - \mathbb{E}_{\pi_E}[c(s,a)]$$

$$\mathbb{E}_{\pi}[c(s,a)] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)] = \sum_{s,a} \rho_{\pi}(s,a) c(s,a)$$

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

$$\mathrm{RL} \odot \mathrm{IRL}_{\psi}(\pi_E) = \arg \min_{\pi \in \Pi} \max_{c} -H(\pi) - \psi(c) + \sum_{s,a} \rho(s,a) c(s,a) - \sum_{s,a} \rho_{\pi_E}(s,a) c(s,a)$$

convex conjugate form

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

(finding a cost function that makes the expert policy uniquely optimal) → (find a policy that matches the expert's occupancy measure)

Generative Adversarial Imitation Learning: proposed regularizer ψ

$$\psi_{\mathrm{GA}}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases} \quad \text{where } g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

Low penalty when x is far from 0; **High penalty** when x is close to 0.

Generative Adversarial Imitation Learning: proposed regularizer ψ

$$\psi_{\mathrm{GA}}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases} \quad \text{where } g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

Low penalty when x is far from 0; **High penalty** when x is close to 0.

$$\psi^*_{\mathrm{GA}}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))]$$

where the maximum ranges over discriminative classifiers $D : \mathcal{S} \times \mathcal{A} \to (0,1)$

The above equation is the **optimal negative log loss** of the binary classification problem of distinguishing between state-action pairs of π and π_E

Generative Adversarial Imitation Learning: proposed regularizer ψ

$$\psi_{\mathrm{GA}}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases} \quad \text{where } g(x) = \begin{cases} -x - \log(1-e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

Low penalty when x is far from 0; **High penalty** when x is close to 0.

$$\psi^*_{\mathrm{GA}}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))]$$

where the maximum ranges over discriminative classifiers $D : \mathcal{S} \times \mathcal{A} \to (0,1)$

The above equation is the **optimal negative log loss** of the binary classification problem of distinguishing between state-action pairs of π and π_E

$$D_{JS}(\rho_\pi, \rho_{\pi_E}) = D_{KL}(\rho_\pi || (\rho_\pi + \rho_E)/2) + D_{KL}(\rho_{\pi_E} || (\rho_\pi + \rho_{\pi_E})/2)$$

Generative Adversarial Imitation Learning: proposed regularizer ψ

$$\psi_{\text{GA}}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases} \quad \text{where } g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

Low penalty when x is far from 0; **High penalty** when x is close to 0.

$$\psi^*_{\text{GA}}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))]$$

where the maximum ranges over discriminative classifiers $D : \mathcal{S} \times \mathcal{A} \to (0,1)$

The above equation is the **optimal negative log loss** of the binary classification problem of distinguishing between state-action pairs of π and π_E

$$\underset{\pi}{\text{minimize}} \; \psi^*_{\text{GA}}(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) = D_{\text{JS}}(\rho_\pi, \rho_{\pi_E}) - \lambda H(\pi)$$

Generative Adversarial Imitation Learning: proposed algorithm

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

- We find the saddle point (π, D)

Generative Adversarial Imitation Learning: proposed algorithm

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

- Initialize the policy π_θ, and a discriminator $D_w : S \times A \rightarrow (0, 1)$
- Alternatively update w and θ:
  - **Adam** for gradient step on **w** to increase

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

  - **TPRO** step on **θ** to decrease

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

- Discriminator network is a **local cost function** providing learning signal to the policy
- Taking a policy step that decreases **expected cost** w.r.t. $c(s, a) = logD(s, a)$

Generative Adversarial Imitation Learning: proposed 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**

Generative Adversarial Imitation Learning: Experiments

- Settings
  - Run on OpenAI Gym
  - Low-dimensional control tasks: (e.g. Cartpole, Acrobat)
  - High-dimensional tasks: (e.g. 3D humanoid locomotion)
- Procedures
  - Use env where *reward* is known
  - Generate expert behavior for these tasks by running TPRO on the true cost functions to create expert policies
  - Run GAIL and other benchmarks on the generated expert policies
  - Evaluate imitation performance w.r.t. sample complexity of expert data
- Benchmarks
  - Behavior Cloning
  - Feature expectation matching (FEM): with **linear** cost function
  - Game-theoretic apprenticeship learning (GTAL): with **convex** cost function

Generative Adversarial Imitation Learning: Results

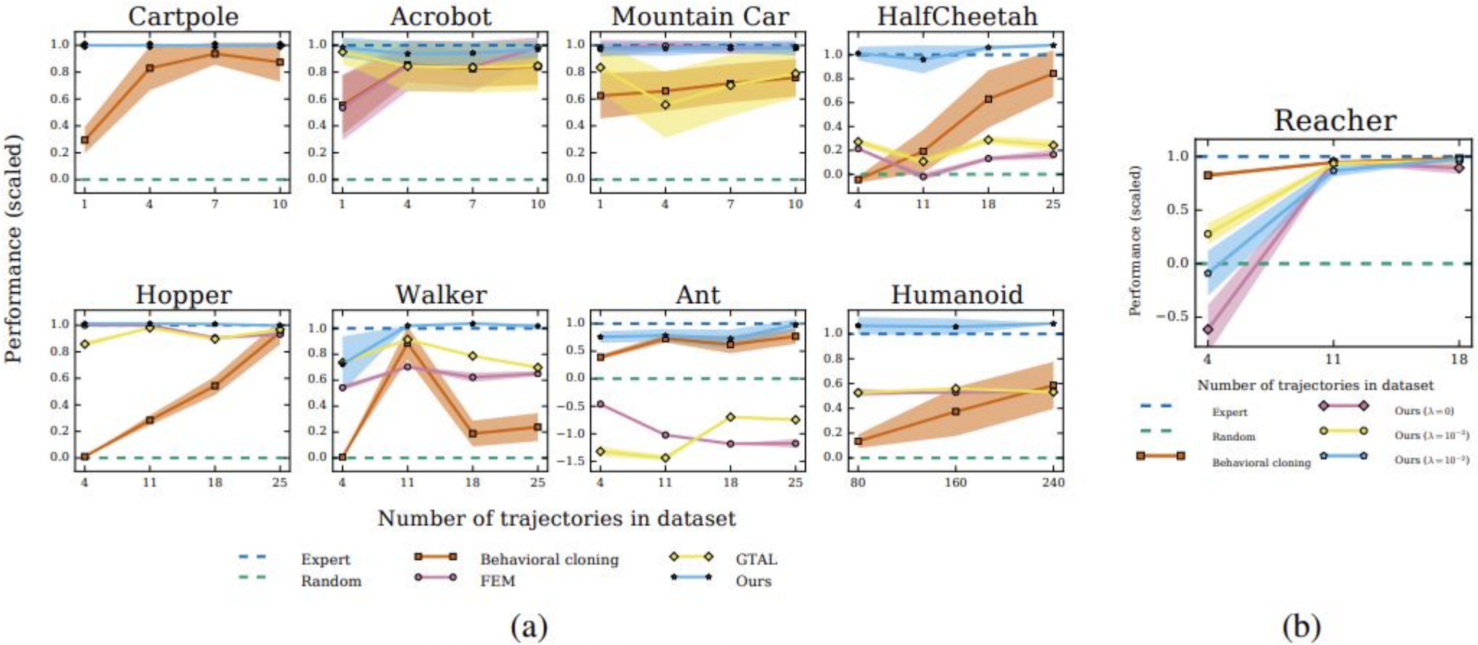


Figure 1: (a) Performance of learned policies. The y-axis is negative cost, scaled so that the expert achieves 1 and a random policy achieves 0. (b) Causal entropy regularization λ on Reacher.

Generative Adversarial Imitation Learning: Results

Table 3: Learned policy performance

Task	Dataset size	Behavioral cloning	FEM	GTAL	Ours
Cartpole	1	72.02 ± 35.82	200.00 ± 0.00	200.00 ± 0.00	200.00 ± 0.00
	4	169.18 ± 59.81	200.00 ± 0.00	200.00 ± 0.00	200.00 ± 0.00
	7	188.60 ± 29.61	200.00 ± 0.00	199.94 ± 1.14	200.00 ± 0.00
	10	177.19 ± 52.83	199.75 ± 3.50	200.00 ± 0.00	200.00 ± 0.00
Acrobot	1	-130.60 ± 55.08	-133.14 ± 60.80	-81.35 ± 22.40	-77.26 ± 18.03
	4	-93.20 ± 32.58	-94.21 ± 47.20	-94.80 ± 46.08	-83.12 ± 23.31
	7	-96.92 ± 34.51	-95.08 ± 46.67	-95.75 ± 46.57	-82.56 ± 20.95
	10	-95.09 ± 33.33	-77.22 ± 18.51	-94.32 ± 46.51	-78.91 ± 15.76
Mountain Car	1	-136.76 ± 34.44	-100.97 ± 12.54	-115.48 ± 36.35	-101.55 ± 10.32
	4	-133.25 ± 29.97	-99.29 ± 8.33	-143.58 ± 50.08	-101.35 ± 10.63
	7	-127.34 ± 29.15	-100.65 ± 9.36	-128.96 ± 46.13	-99.90 ± 7.97
	10	-123.14 ± 28.26	-100.48 ± 8.14	-120.05 ± 36.66	-100.83 ± 11.40
HalfCheetah	4	-493.62 ± 246.58	734.01 ± 84.59	1008.14 ± 280.42	4515.70 ± 549.49
	11	637.57 ± 1708.10	-375.22 ± 291.13	226.06 ± 307.87	4280.65 ± 1119.93
	18	2705.01 ± 2273.00	343.58 ± 159.66	1084.26 ± 317.02	4749.43 ± 149.04
	25	3718.58 ± 1856.22	502.29 ± 375.78	869.55 ± 447.90	4840.07 ± 95.36
Hopper	4	50.57 ± 0.95	3571.98 ± 6.35	3065.21 ± 147.79	3614.22 ± 7.17
	11	1025.84 ± 266.86	3572.30 ± 12.03	3502.71 ± 14.54	3615.00 ± 4.32
	18	1949.09 ± 500.61	3230.68 ± 4.58	3201.05 ± 6.74	3600.70 ± 4.24
	25	3383.96 ± 657.61	3331.05 ± 3.55	3458.82 ± 5.40	3560.85 ± 3.09
Walker	4	32.18 ± 1.25	3648.17 ± 327.41	4945.90 ± 65.97	4877.98 ± 2848.37
	11	5946.81 ± 1733.73	4723.44 ± 117.18	6139.29 ± 91.48	6850.27 ± 39.19
	18	1263.82 ± 1347.74	4184.34 ± 485.54	5288.68 ± 37.29	6964.68 ± 46.30
	25	1599.36 ± 1456.59	4368.15 ± 267.17	4687.80 ± 186.22	6832.01 ± 254.64
Ant	4	1611.75 ± 359.54	-2052.51 ± 49.41	-5743.81 ± 723.48	3186.80 ± 903.57
	11	3065.59 ± 635.19	-4462.70 ± 53.84	-6252.19 ± 409.42	3306.67 ± 988.39
	18	2597.22 ± 1366.57	-5148.62 ± 37.80	-3067.07 ± 177.20	3033.87 ± 1460.96
	25	3235.73 ± 1186.38	-5122.12 ± 703.19	-3271.37 ± 226.66	4132.90 ± 878.67
Humanoid	80	1397.06 ± 1057.84	5093.12 ± 583.11	5096.43 ± 24.96	10200.73 ± 1324.47
	160	3655.14 ± 3714.28	5120.52 ± 17.07	5412.47 ± 19.53	10119.80 ± 1254.73
	240	5660.53 ± 3600.70	5192.34 ± 24.59	5145.94 ± 21.13	10361.94 ± 61.28
Task	**Dataset size**	**Behavioral cloning**	**Ours ($\lambda = 0$)**	**Ours ($\lambda = 10^{-3}$)**	**Ours ($\lambda = 10^{-2}$)**
Reacher	4	-10.97 ± 7.07	-67.23 ± 88.99	-32.37 ± 39.81	-46.72 ± 82.88
	11	-6.23 ± 3.29	-6.06 ± 5.36	-6.61 ± 5.11	-9.26 ± 21.88
	18	-4.76 ± 2.31	-8.25 ± 21.99	-5.66 ± 3.15	-5.04 ± 2.22

References

- Paper
  - https://arxiv.org/pdf/1606.03476.pdf - 2016
- Code links
  - https://github.com/hollygrimm/gail-mujoco
  - https://github.com/andrewliao11/gail-tf
- References
  - https://efrosgans.eecs.berkeley.edu/CVPR18_slides/GAIL_by_Ermon.pdf
  - https://vitalab.github.io/article/2019/05/23/GAIL.html
  - https://hollygrimm.com/posts/rl_gail/
  - https://web.stanford.edu/class/cs234/CS234Win2019/slides/lecture7.pdf
  - https://web.stanford.edu/class/cs234/CS234Win2020/slides/lecture7.pdf
  - https://www.youtube.com/watch?v=6rZTaboSY4k