Generative Adversarial Imitation Learning

Jonathan Ho and Stefano Ermon

Topic: Inverse RL
Presenter: Albert Hsueh

Problem Addressed

- Imitation learning for large and maybe complex environments is intractable
- Maybe cost function cannot capture the actual task or is too expensive to evaluate

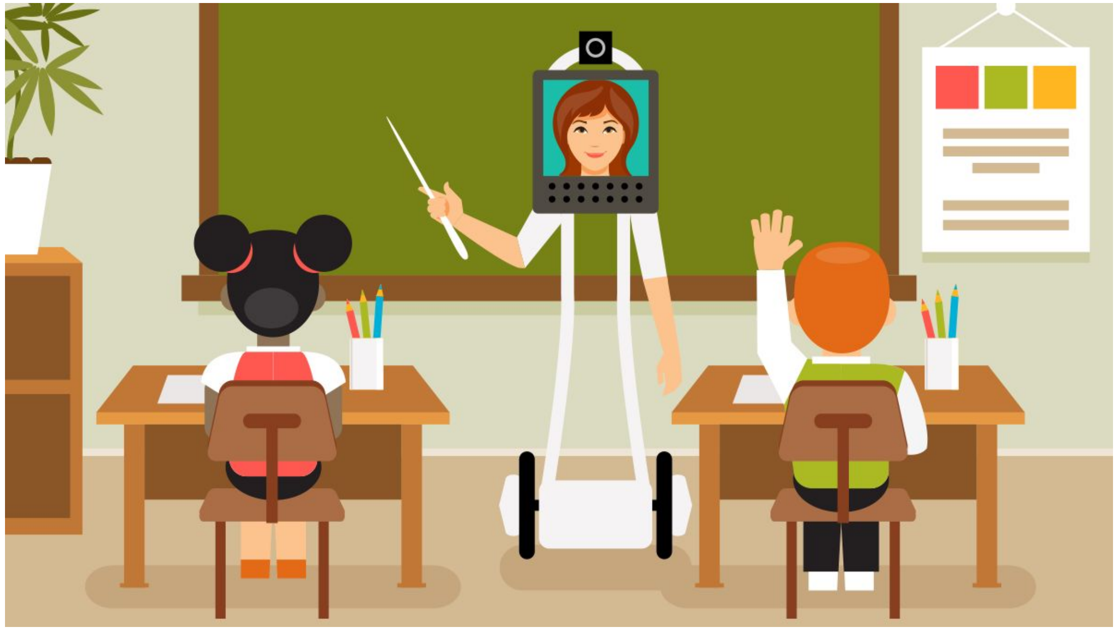


Image retrieved from:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fnewsroom.cisco.com%2Ffeature-content%3FarticleId%3D1873531&psig=AOvVaw1r35LJP5Qy0U6zokGWH-Q6&ust=1582651631781000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCJDby_7a6ucCFQAAAAAdAAAAABAJ

Limitations of Prior work

- Behaviour cloning require lots of data
- Inverse RL is computationally expensive
- Thus imitation learning for large scale learning environments remain unsolved research challenge

GAIL's Contribution

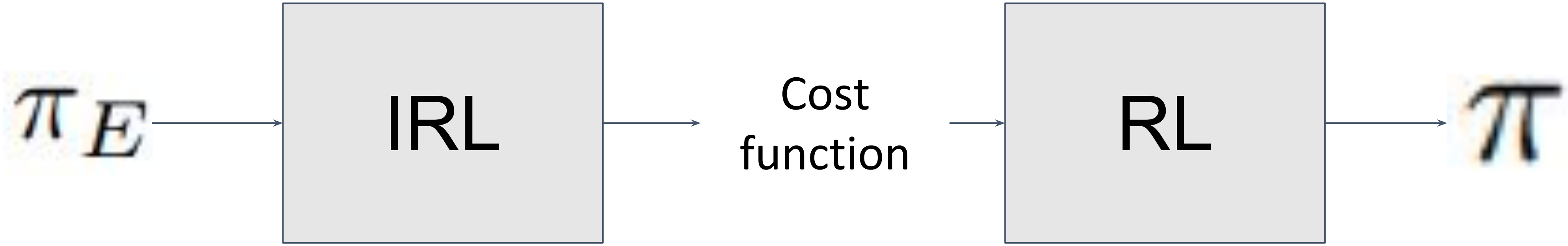
1. New algorithm using particular choice of regularizer, Ψ
2. Proved mathematically that the new algorithm is similar to GAN
3. State of the art performance -- beat 9 environment baselines and demonstrated feasibility of algo on increasingly larger environments

Background Concepts

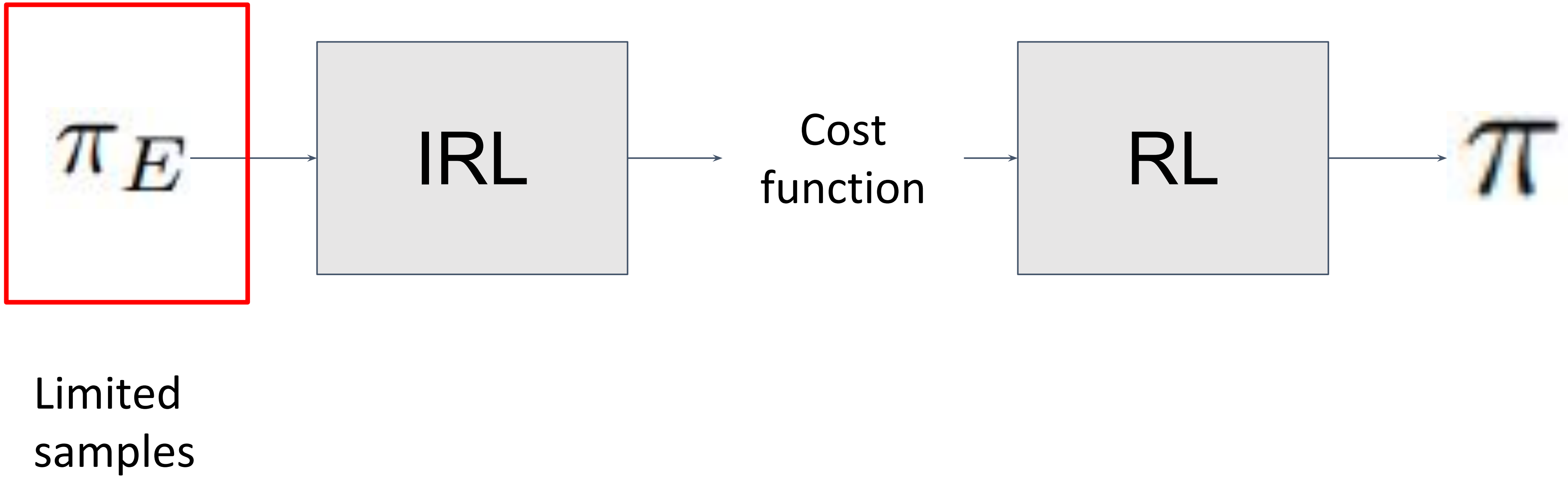
Important Background Concepts

- The vanilla learning process: RL(IRL(pi_E))
- IRL’s cross entropy formulation
- Regularization & model capacity
- Occupancy measure

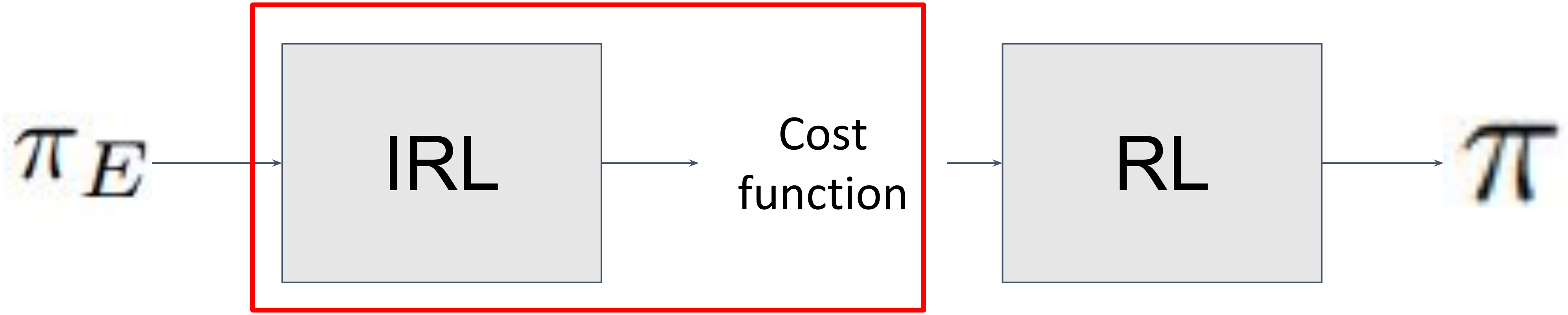
RL(IRL(pi_E))



RL(IRL(pi_E))

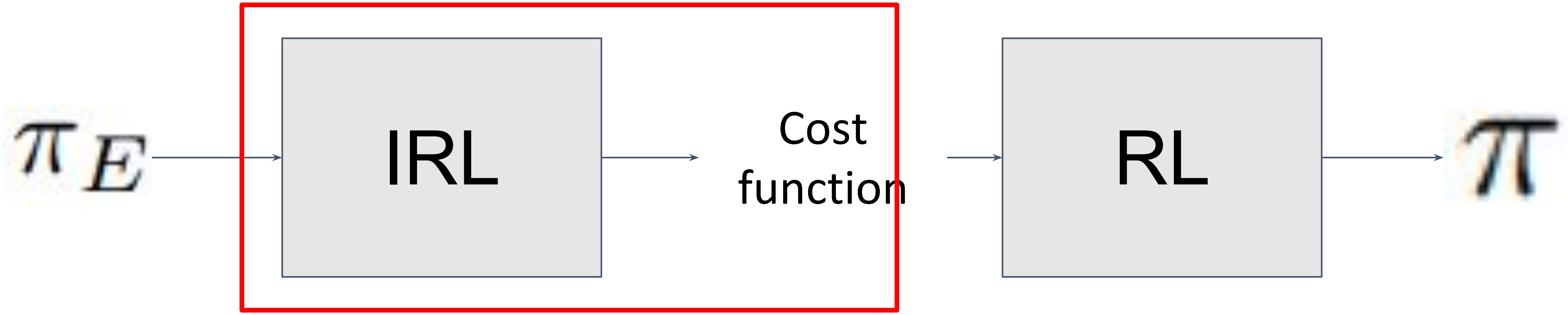


RL(IRL(pi_E))

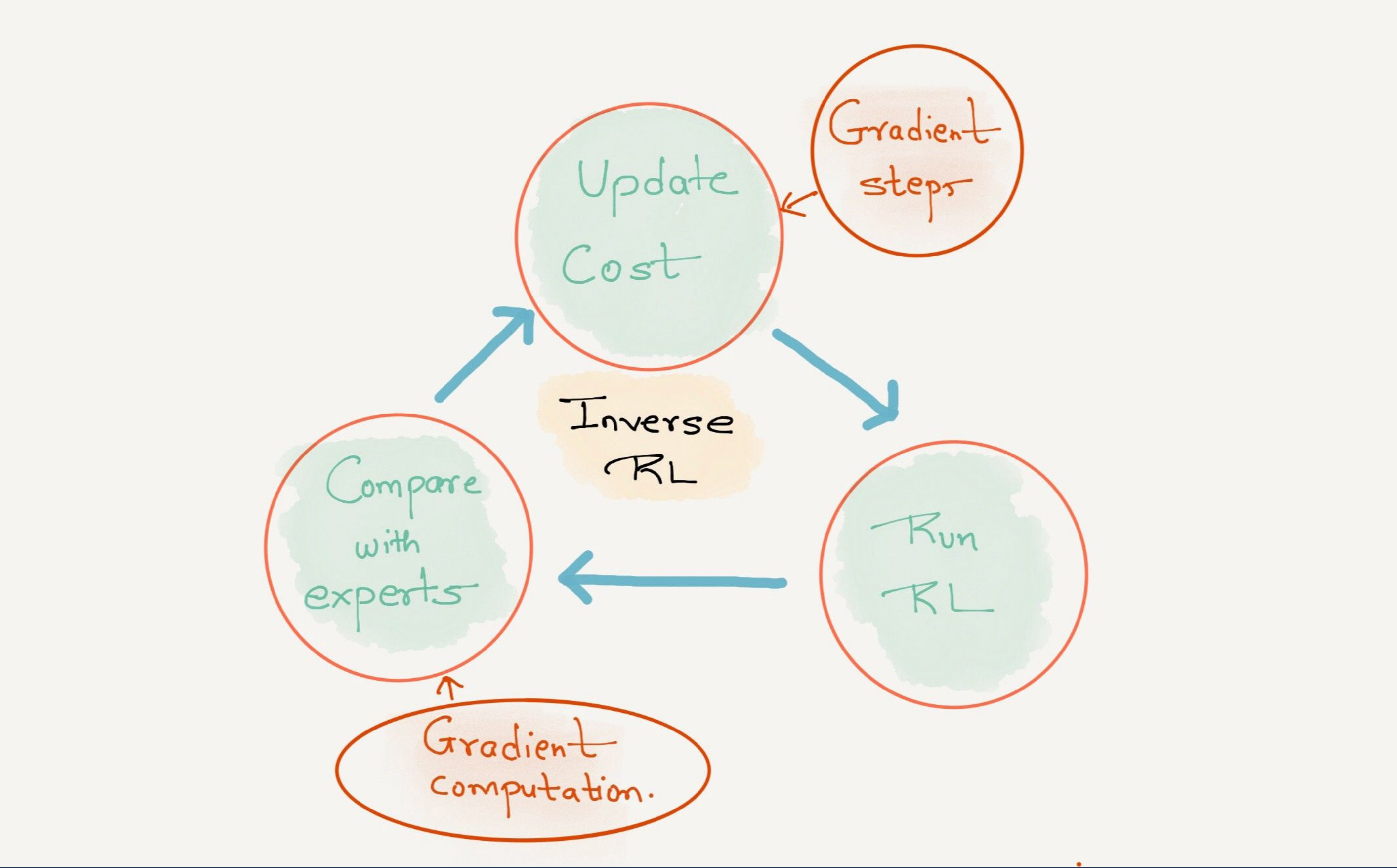


Slow, and perhaps not expressive enough

RL(IRL(pi_E))



Requires inner loop



https://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html
https://medium.com/@sanketgujar95/generative-adversarial-imitation-learning-266f45634e60

RL(IRL(pi_E))

$\pi_E \rightarrow$ One Step? $\rightarrow \pi$

Important Background Concepts

- ~~The vanilla learning process: RL(IRL(pi_E))~~
- IRL's cross entropy formulation
- Regularization & model capacity
- Occupancy measure

IRL's cross entropy formulation

- Key point here is that the H entropy term add noise to the system and enable exploration

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

Important Background Concepts

- ~~The vanilla learning process: RL(IRL(pi_E))~~
- ~~IRL's cross entropy formulation~~
- Regularization & model capacity
- Occupancy measure

Regularization & model capacity

- Use expressive, high capacity models (DNN) to learn cost functions may overfit easily given finite dataset (particularly since expert dataset usually small) so we need to regularize
    - High capacity models necessary for high us to rationalize expert behavior without hand crafting features

- Turns out that regularizer dictates what algorithm class gets recovered

Regularization & model capacity

- Use expressive … n cost functions may overfit ea… since expert dataset usually…
  - High capacity … e expert behavior without hand…

- Turns out that … class gets recovered

Important Background Concepts

- ~~The vanilla learning process: RL(IRL(pi_E))~~
- ~~IRL's cross entropy formulation~~
- ~~Regularization & model capacity~~
- Occupancy measure

Occupancy Measure

$$\mathbb{E}_\pi[c(s,a)] = \sum_{s,a} \rho_\pi(s,a)c(s,a) \text{ for any cost function } c.$$

- "Distribution of (s,a) that agent encounters" given its policy

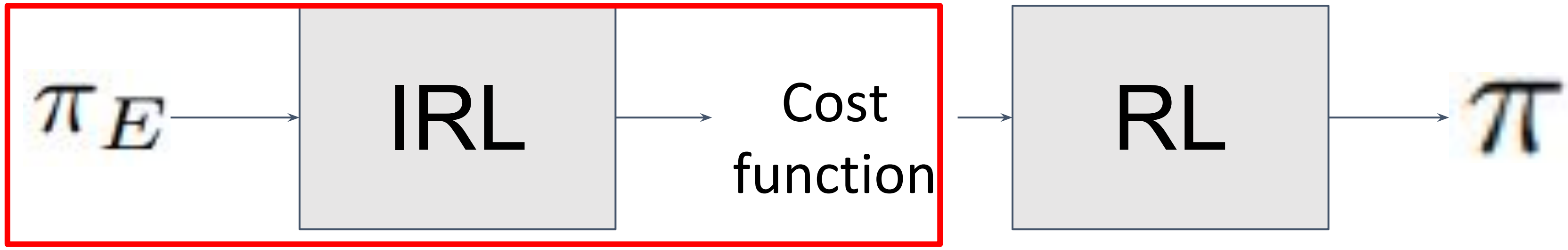
Proposition 3.1 (Theorem 2 of Syed et al. [29]). *If $\rho \in \mathcal{D}$, then ρ is the occupancy measure for $\pi_\rho(a|s) \triangleq \rho(s,a)/\sum_{a'} \rho(s,a')$, and $\boxed{\pi_\rho \text{ is the only policy whose occupancy measure is } \rho.}$*

Algorithm

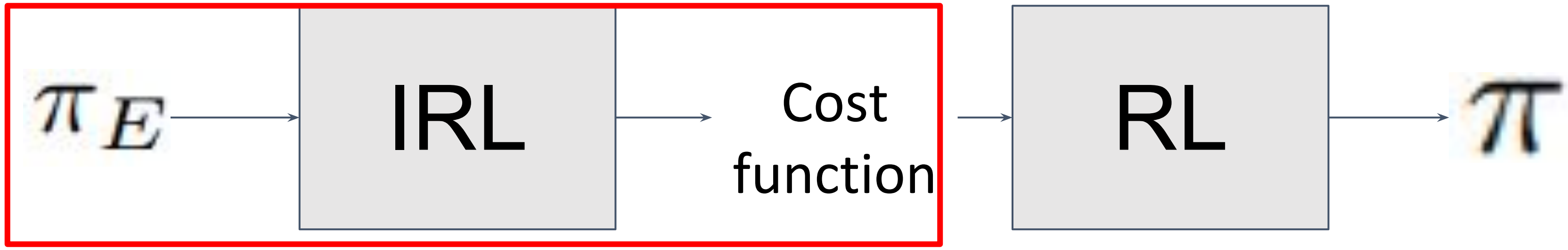
Algorithm Design Method (key takeaways)

- Proposition 3.1 + lemma 3.1 -> duality of IRL/occupancy matching
- Choice of regularizer gives rise to different known algorithm classes
- Lead to GAN loss function

Derivation

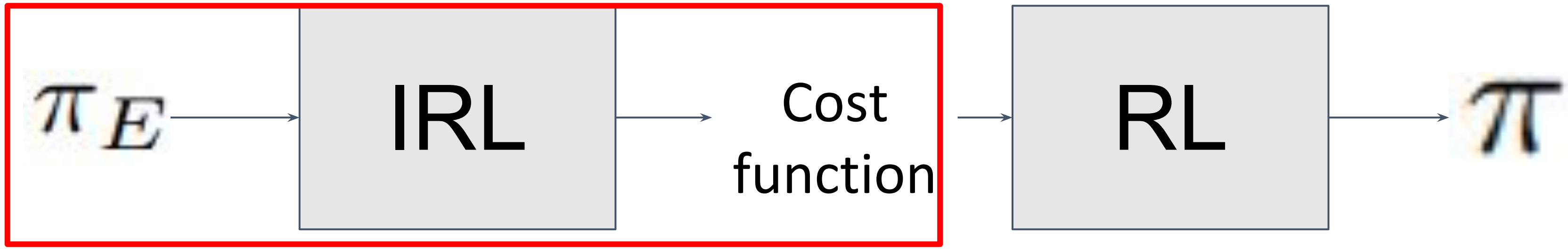


$$\underset{c \in \mathcal{C}}{\operatorname{maximize}} \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)] \tag{1}$$



$$\underset{c \in \mathcal{C}}{\text{maximize}} \left(\boxed{\text{E\_pi* [c(s,a)]}} \right) - \mathbb{E}_{\pi_E}[c(s,a)] \qquad (1)$$

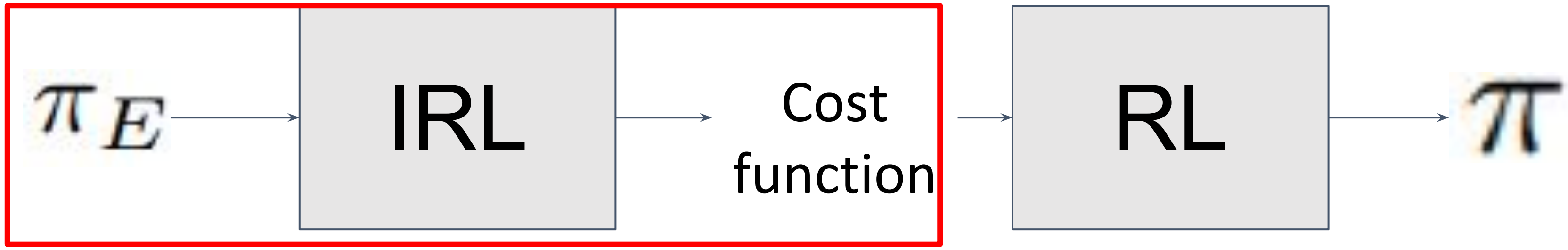
- Outer optimization wants to find the cost function that can distinguish between the expert and some “worst-case”/”most like expert” policy pi* that there is



E_pi* [c(s,a)]

$$\left(\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_\pi[c(s,a)]\right) \qquad (1)$$

- Inner optimization finds our pi* from all possible pi by selecting the pi that generate trajectories with the lowest average cost.

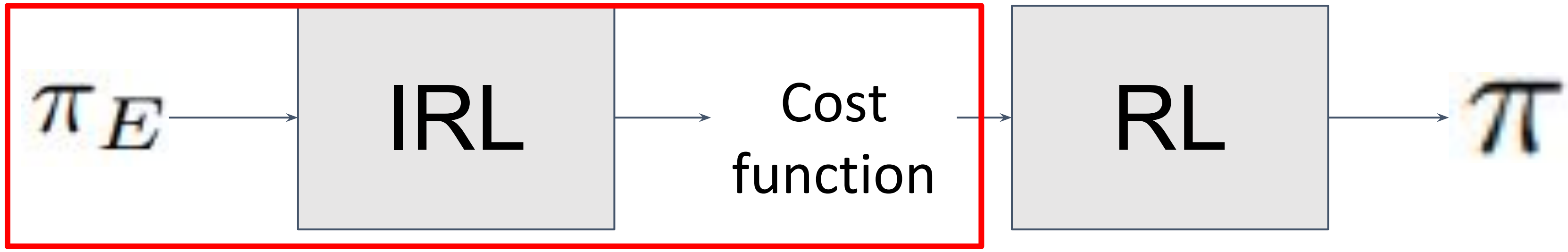


E_pi* [c(s,a)]

$$\left(\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_\pi[c(s,a)]\right) \tag{1}$$

- Inner optimization finds our pi* from all possible pi by selecting the pi that generate trajectories with the lowest average cost.
- By doing so we essentially are doing RL in the “inner loop”

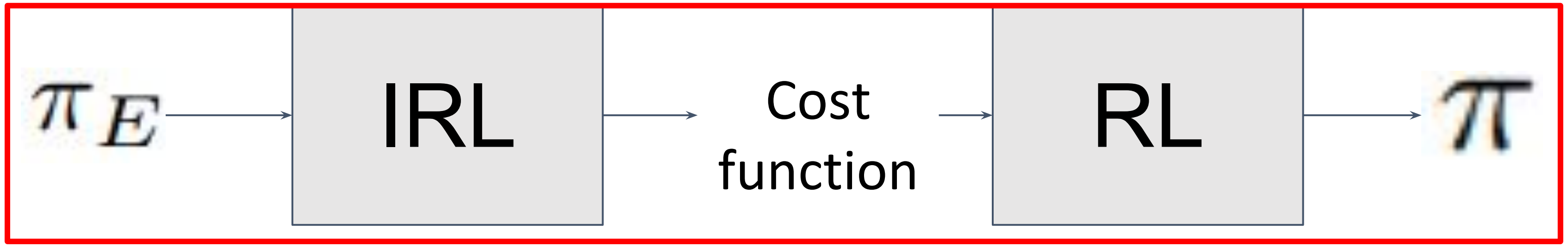
$$\mathrm{RL}(c) = \arg\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_\pi[c(s,a)] \tag{2}$$



$$\underset{c \in \mathcal{C}}{\text{maximize}} \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)] \qquad (1)$$

$$\mathrm{IRL}_{\psi}(\pi_E) = \underset{c \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}}{\arg\max} -\psi(c) + \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)] \qquad (3)$$

Add regularization to prevent overfitting



$$\underset{c \in \mathcal{C}}{\text{maximize}} \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)] \tag{1}$$

$$\mathrm{IRL}_{\psi}(\pi_E) = \underset{c \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}}{\arg\max} -\psi(c) + \left(\min_{\pi \in \Pi} -H(\pi) + \mathbb{E}_{\pi}[c(s,a)] \right) - \mathbb{E}_{\pi_E}[c(s,a)] \tag{3}$$

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

By Proposition 3.2's proof in the Appendix

Interlude -- Regularizer design

- Low penalty on functions that give negative cost to expert trajectories
- Heavily penalize otherwise

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

Interlude -- Regularizer design

- Low penalty on functions that give negative cost to expert trajectories
- Heavily penalize ot

WHY USE THIS?

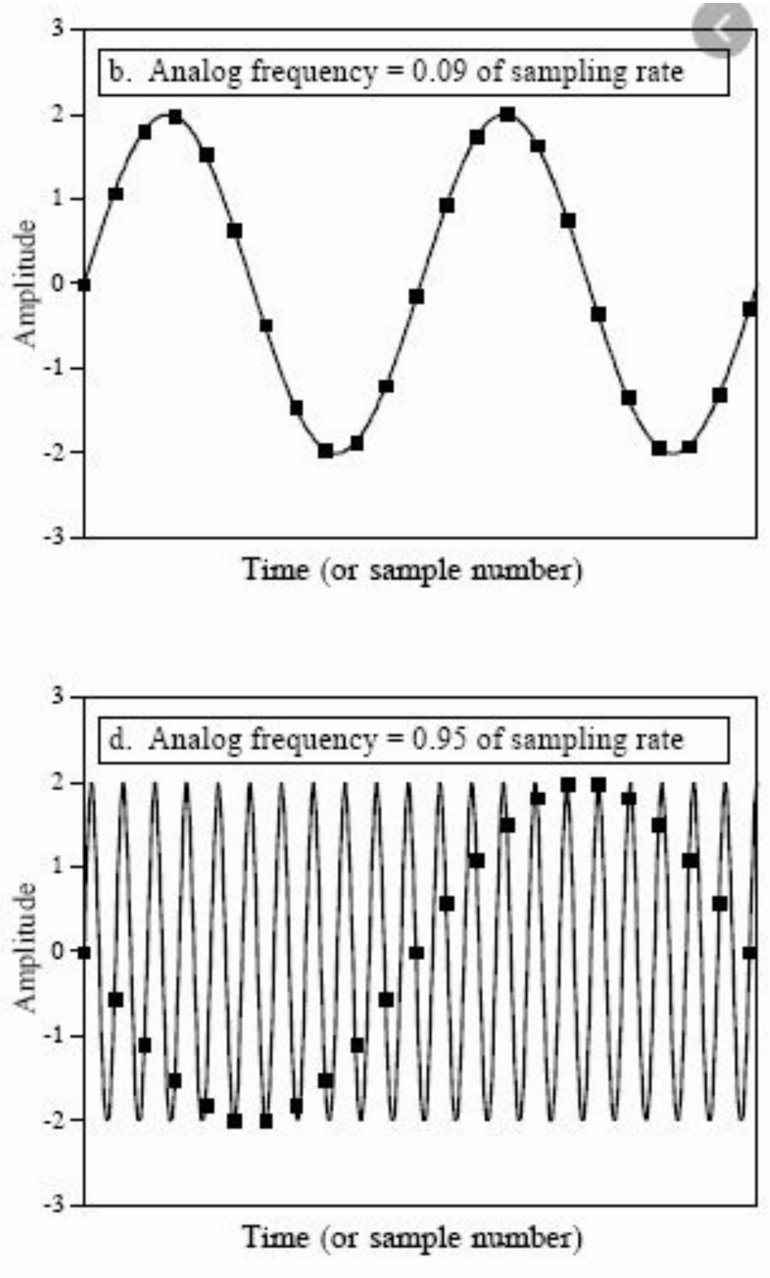
$$\psi_{\mathrm{GA}}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s & \\ +\infty & \text{otherwise} \end{cases} \quad g(x) = \begin{cases} \log(1-e^{x}) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

Interlude -- Regularizer design

- Constant regularizer -> Exact Occupancy Matching
    - large environments and few expert samples makes it intractable

Interlude -- Re

- Constant regularize ching
  - large environment es it intractable



Interlude -- Regularizer design

- Constant regularizer -> Exact Occupancy Matching
    - large environments and few expert samples makes it intractable
- Indicator regularizer -> apprenticeship learning
    - Scales well to large environments but require careful tuning to exact match

$\pi_E \longrightarrow$ One Step [done, eqn. 4] $\longrightarrow \pi$

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

π_E → One Step [done, eqn. 4] → π

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

$$\psi^*_{\mathrm{GA}}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] \tag{14}$$

Eqn. 14 is a fact from Corollary A.1.1

$\pi_E \longrightarrow$ One Step [done, eqn. 4] $\longrightarrow \pi$

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

$$\psi^*_{\mathrm{GA}}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] \tag{14}$$

$$\Downarrow$$

$$\mathbb{E}_\pi[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi) \tag{16}$$

GAN Loss

- Train a discriminator D to minimize this loss
- Train a generator to maximize this same loss

$$\mathbb{E}_{\pi}[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi) \qquad (16)$$

GAN Loss

- Train a discriminator D to minimize this loss
- Train a generator to maximize this same loss

$$\mathbb{E}_{\pi}[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi) \tag{16}$$

$D(s,a)$ = 0 if true distribution (ie drawn from expert)
= 1 if false distribution (ie done by learner's policy)

The algo

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

Include sample screenshot of environments

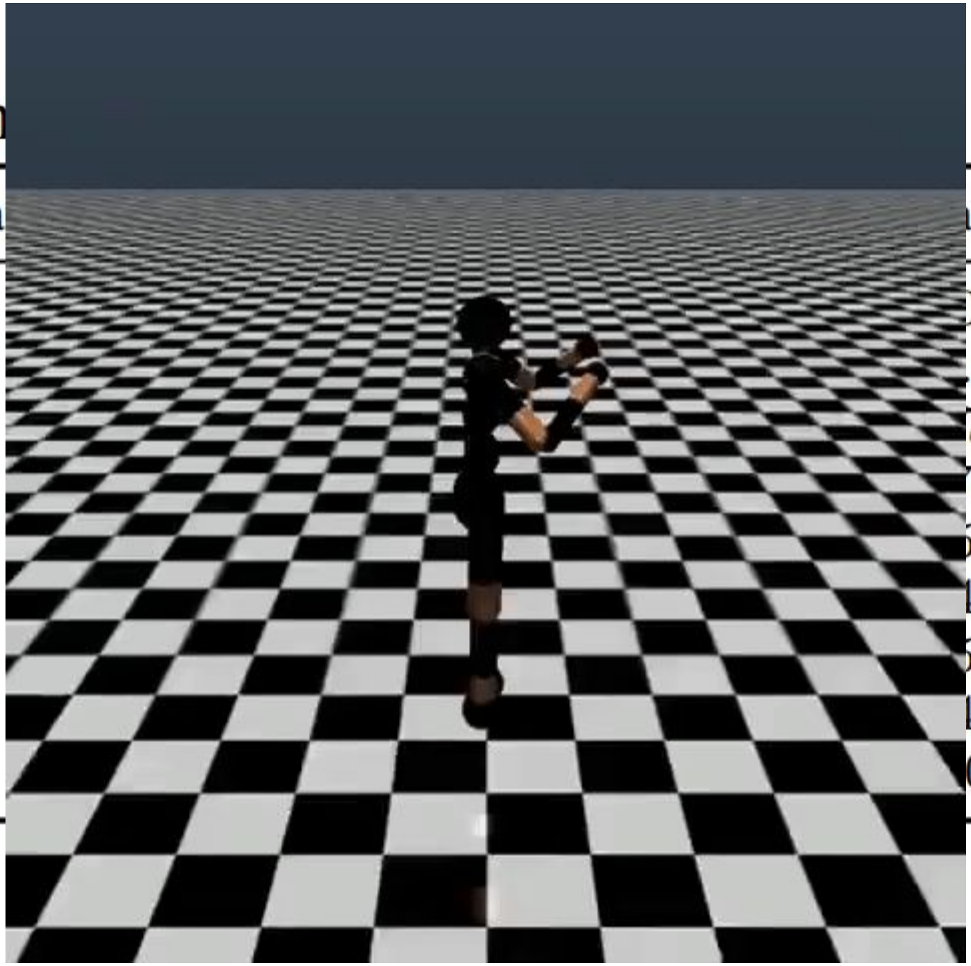
Table 1: Environments

Task	Observation space	Action space	Random policy performance	Expert performance
Cartpole-v0	4 (continuous)	2 (discrete)	18.64 ± 7.45	200.00 ± 0.00
Acrobot-v0	4 (continuous)	3 (discrete)	-200.00 ± 0.00	-75.25 ± 10.94
Mountain Car-v0	2 (continuous)	3 (discrete)	-200.00 ± 0.00	-98.75 ± 8.71
Reacher-v1	11 (continuous)	2 (continuous)	-43.21 ± 4.32	-4.09 ± 1.70
HalfCheetah-v1	17 (continuous)	6 (continuous)	-282.43 ± 79.53	4463.46 ± 105.83
Hopper-v1	11 (continuous)	3 (continuous)	14.47 ± 7.96	3571.38 ± 184.20
Walker-v1	17 (continuous)	6 (continuous)	0.57 ± 4.59	6717.08 ± 845.62
Ant-v1	111 (continuous)	8 (continuous)	-69.68 ± 111.10	4228.37 ± 424.16
Humanoid-v1	376 (continuous)	17 (continuous)	122.87 ± 35.11	9575.40 ± 1750.80

Include sample screenshot of environments

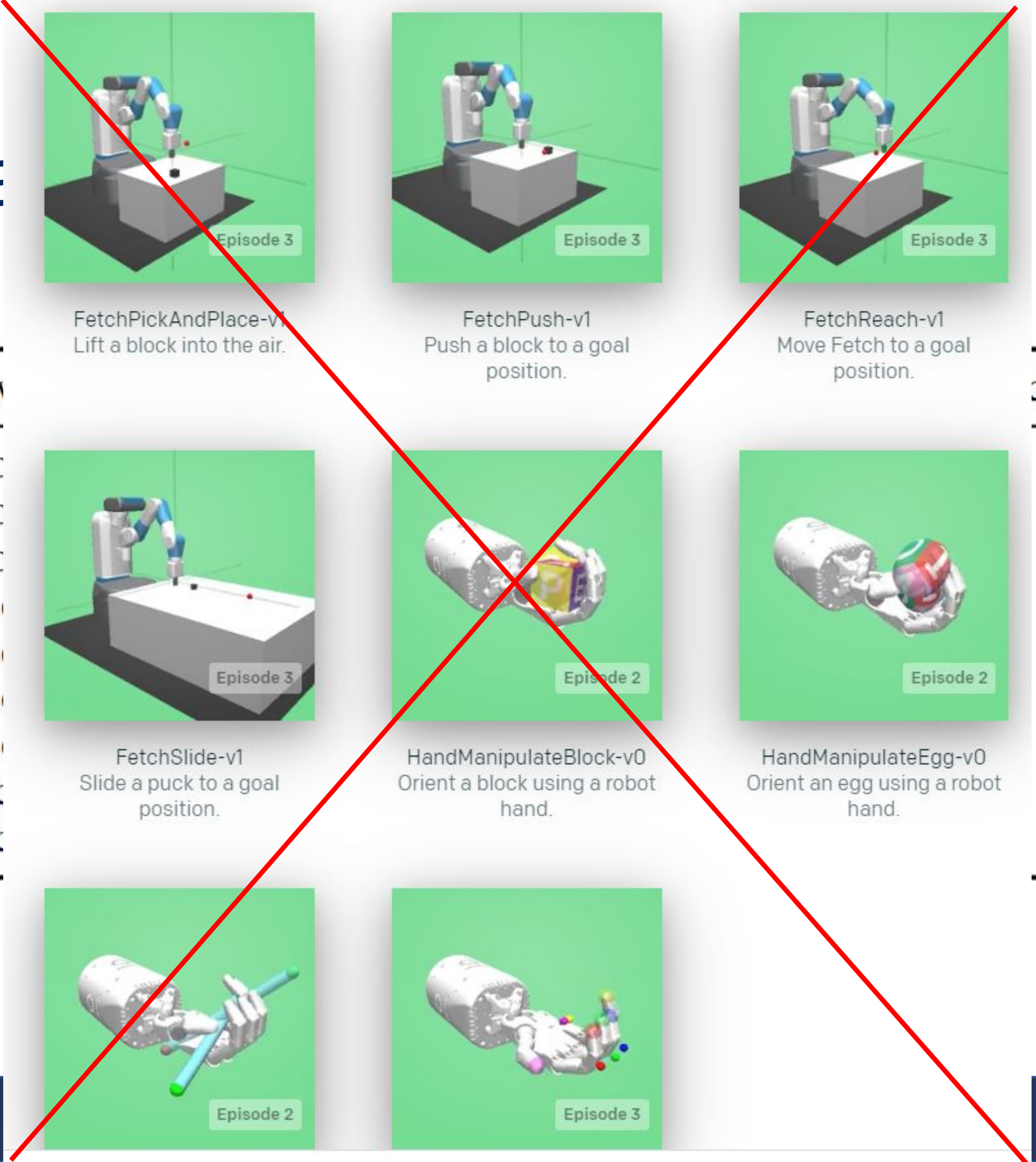
Table 1: Environ

Task	Observation space	Action space	Ra	nce
Cartpole-v0	4 (continuous)	2 (discrete)		00
Acrobot-v0	4 (continuous)	3 (discrete)		94
Mountain Car-v0	2 (continuous)	3 (discrete)		71
Reacher-v1	11 (continuous)	2 (continuous)		0
HalfCheetah-v1	17 (continuous)	6 (continuous)		.83
Hopper-v1	11 (continuous)	3 (continuous)		.20
Walker-v1	17 (continuous)	6 (continuous)		.62
Ant-v1	111 (continuous)	8 (continuous)		.16
Humanoid-v1	376 (continuous)	17 (continuous)		0.80

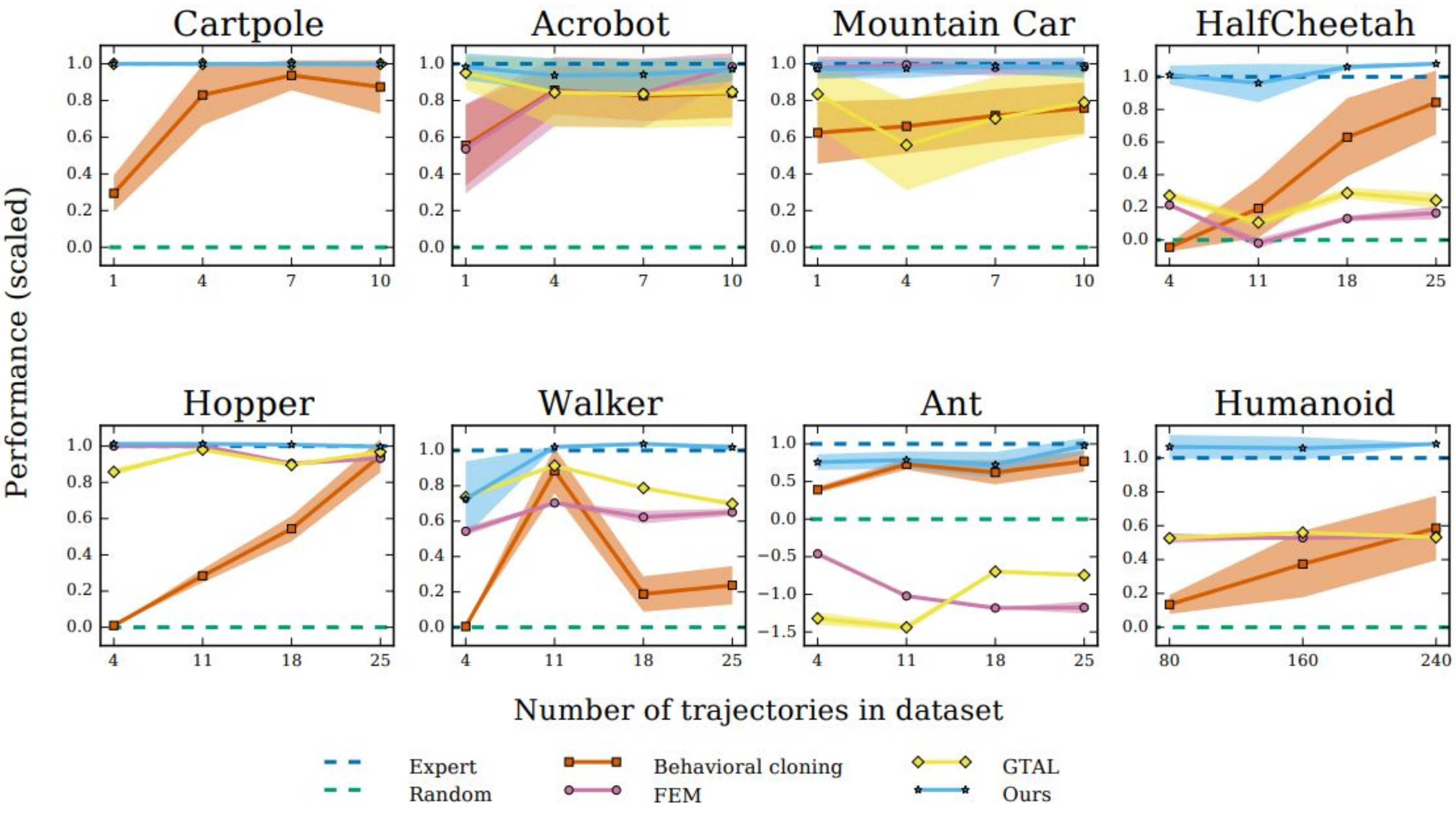


Include sa nmments

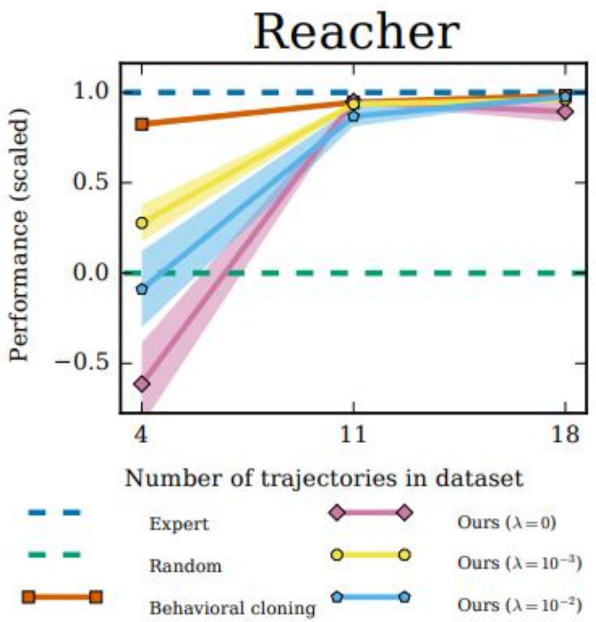
Task	Obser	ce	Expert performance
Cartpole-v0	4 (cc		200.00 ± 0.00
Acrobot-v0	4 (cc		-75.25 ± 10.94
Mountain Car-v0	2 (cc		-98.75 ± 8.71
Reacher-v1	11 (c		-4.09 ± 1.70
HalfCheetah-v1	17 (c		4463.46 ± 105.83
Hopper-v1	11 (c		3571.38 ± 184.20
Walker-v1	17 (c		6717.08 ± 845.62
Ant-v1	111 (c		4228.37 ± 424.16
Humanoid-v1	376 (c		9575.40 ± 1750.80



Results



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



Reacher
Performance (scaled)
Number of trajectories in dataset
Expert
Random
Behavioral cloning
Ours (λ=0)
Ours (λ=10⁻³)
Ours (λ=10⁻²)

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

Discussion of results

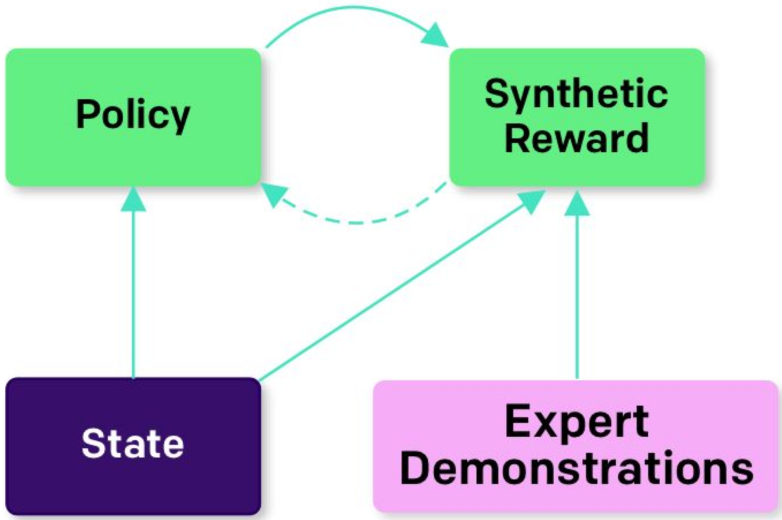
- Shows evidence that gail out performs benchmark on increasingly larger environments
- Which environments does GAIL fail?
    - Images
    - Stochastic Dynamics
    - Robotics gym env

Very good paper but -- Limitations

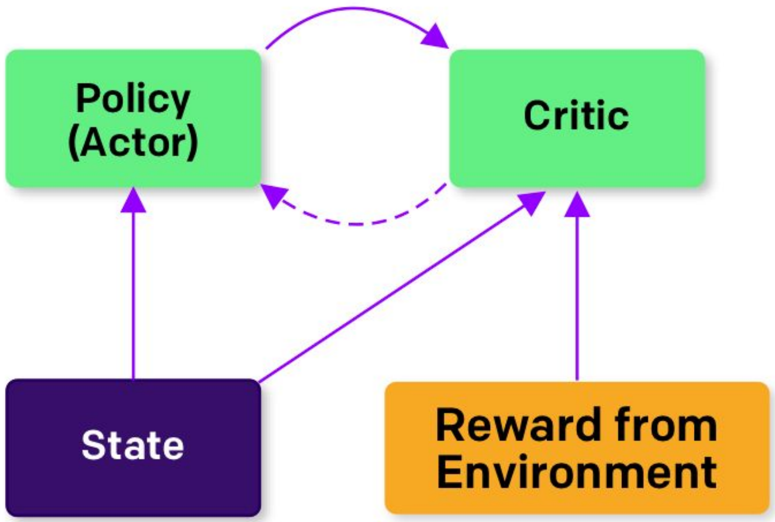
- Slow learning
- Qualitative performance not captured in metric
- Does not comment on whether there are better regularizers. Seems intuitive to apply gan to imitation learning but obviously the math proofs to show the equivalence is huge.
- Difficult paper to understand so less experimental insights is unfortunate

Recap

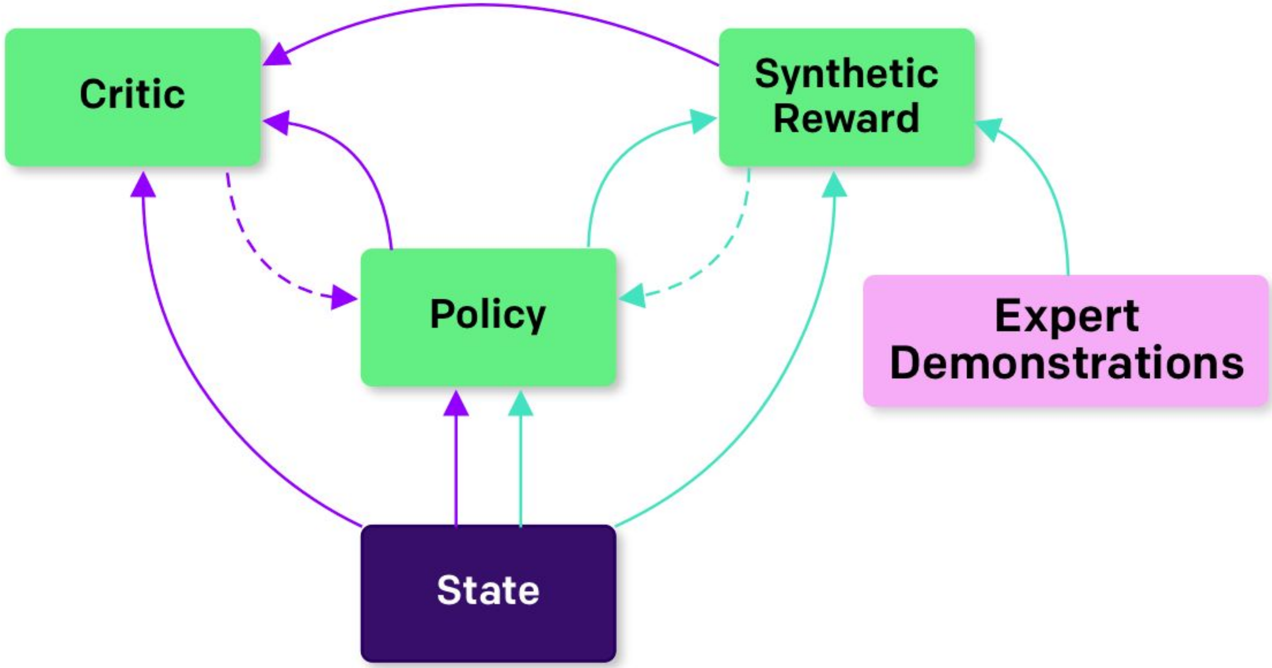
1. Occupancy measure is unique to a policy
2. Optimizing occupancy measure is the dual to optimizing over policy
3. We use this to reframe the vanilla imitation learning: RL(IRL(pi_E)) problem
4. Entropy allows for better exploration
5. Regularization allows us to recover different algorithm classes
6. Particular regularizer was chosen to allow scalability to large environments while also allowing exact matching of expert policy/occupancy measure
7. The conjugate of the regularizer + vanilla imitation learning resembles GAN loss



a) GAIL



b) Actor-Critic



c) SAM

http://dmml.ch/sample-efficient-imitation-learning-via-generative-adversarial-nets/