# IL, RL & GANs

王政飞

2018.11.9

# Table of Contents

# Foreword

- Suppose you are playing a computer game, but at the very beginning, it seems that whatever you do, you will die in a short time. What will you do?
- You are desperate and decide to search for a video of this game on YouTube. What will you focus on this video?
- Coincidentally, now one of your friend who has already pass this game passes by and notice you are getting crazy. What will you ask from him?

# Expert Trajectories -> Your Trajectories


Expert Trajectories
(Training Distribution)


Behavioral Cloning
Makes mistakes, enters new states
Cannot recover from new states

# Imitation Learning

- Behavioral Cloning, *an example from Disney Research*
  - Advantages: simple and efficient
  - Disadvantages: distribution mismatch, no long term planning
  - Consider: 1-step deviations affection, state-action cover area, long term objective
- If you have interactive demonstrator:
  1. Supervised learning
  2. Rollout in environment
  3. Collect Demonstrations (BC is 1-step special case), back to Step 1

# Imitation Learning cont.

- Reward engineering can be **REALLY** hard…
- So here comes Inverse RL with expert behaviors
  1. Run full RL
  2. Compare with expert
  3. Update reward, back to Step 1
- However for Inverse RL
  - Many reward function corresponding to the same policy
  - Really difficult to train for double loop

# Generative Adversarial Imitation Learning

- OpenAI & Stanford. NIPS 2016
- Intuition:
  - Behavioral Cloning (BC): simple, need large amount of data; else compounding error
  - Inverse RL: learn a cost function that prioritizes entire trajectories over others; expensive to run
  - Desire an algorithm that tells us explicitly how to act by **directly** learning a policy
- **THICK** math in this paper…

# Simplified Math

- We want a policy $\pi$ and hope it could act like our demonstration policy $\pi_{\mathrm{E}}$, and if possible, get more reward
- So takes it into 2 steps:
  1. Act like demonstration policy: use Jensen-Shannon divergence
  2. Better policy: keep regularization
- Therefore, our new imitation learning algorithm is:

$$\min_{\pi} D_{JS}\left(\rho_{\pi} - \rho_{\pi_E}\right) - \lambda H(\pi)$$

# Practical Algorithm

**Algorithm 1** Generative adversarial imitation learning

1: **Input:** Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminator parameters $\theta_0, w_0$
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 $\theta_i$ to $\theta_{i+1}$, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$. Specifically, take a KL-constrained natural gradient step with

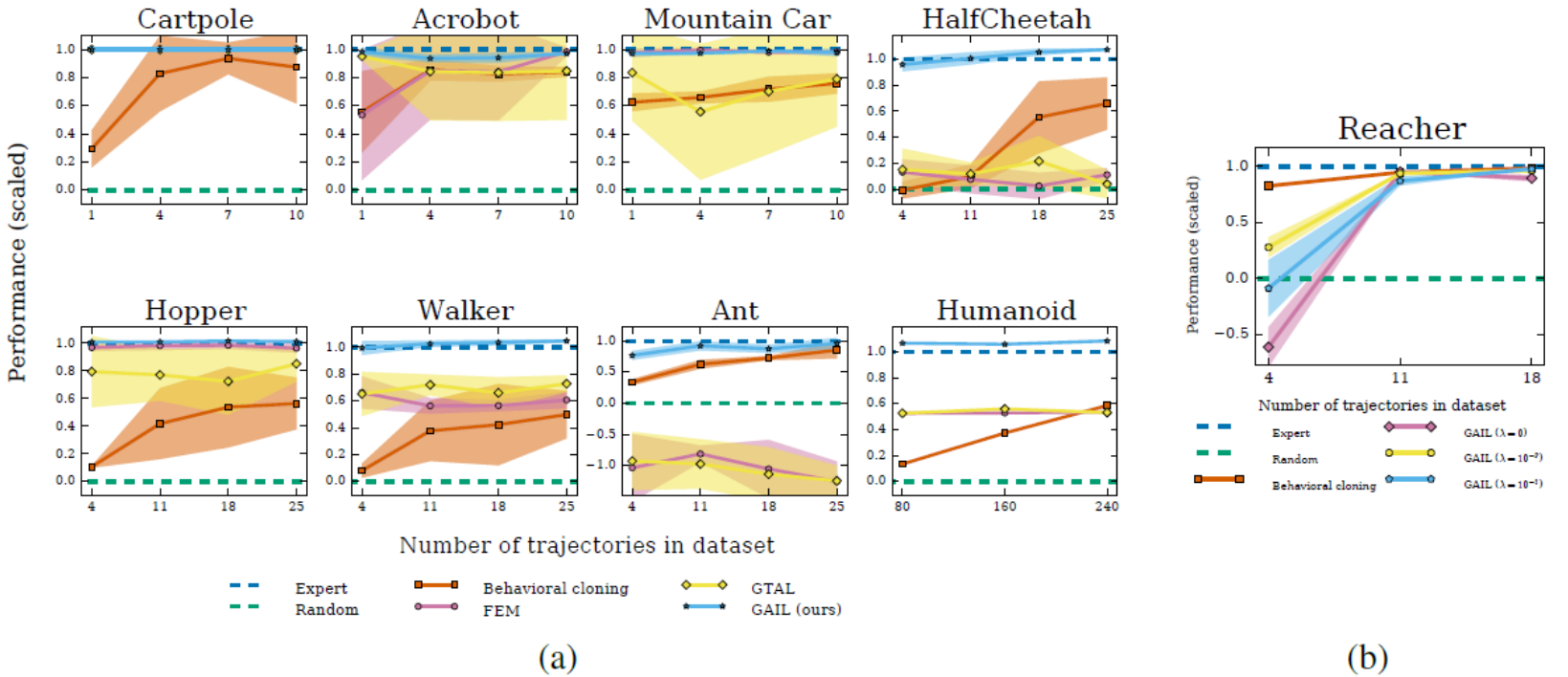
$$\begin{aligned} &\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta H(\pi_\theta), \\ &\text{where } Q(\bar{s},\bar{a}) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{w_{i+1}}(s,a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \tag{18}$$

6: **end for**

# Experiments

- Expert polices comes from TRPO, 50 state-action pairs per task
- Tasks: 9 physics-based control tasks
  - cartpole, acrobot, mountain car
  - 3D humanoid locomotion
- Baselines:
  - Behavioral cloning (BC)
  - Feature expectation matching (FEM, Inverse RL)
  - Game-theoretic apprenticeship learning (GTAL, Inverse RL)

# Results




(a) (b)

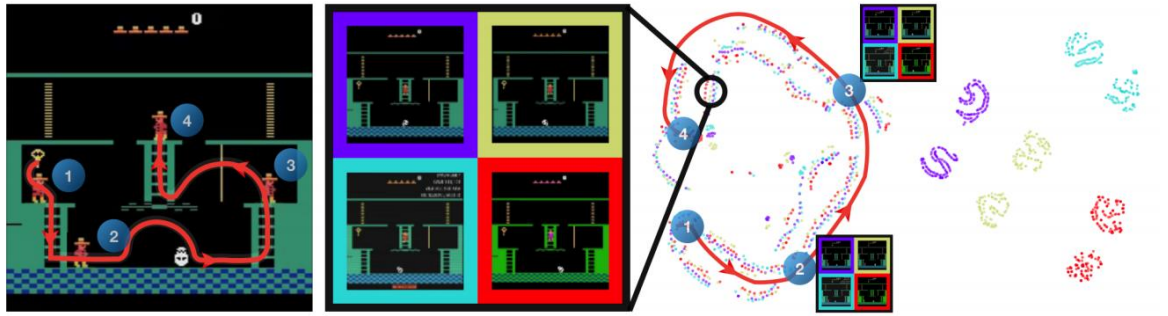
Figure 1: (a) Performance of learned policies. The $y$-axis is negative cost, scaled so that the expert achieves 1 and a random policy achieves 0. (b) Causal entropy regularization $\lambda$ on Reacher. Except for Humanoid, shading indicates standard deviation over 5-7 reruns.

# Some Thoughts

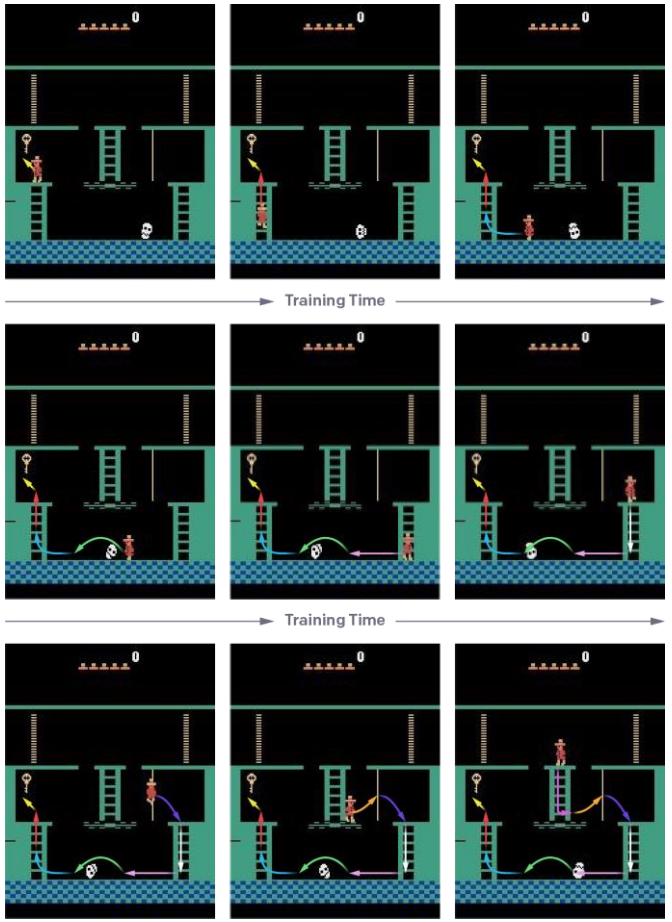
- Demonstration trajectories come from TRPO, because it was the best source of expert trajectories. (by Jonathon Ho)
- GAIL or imitation learning should be adopted in environment without specific reward function, otherwise naïve reinforcement learning algorithm should be better and efficient. (except for sparse and delayed reward function, like *Montezuma's Revenge*)

# Solutions for MR

- DeepMind
- Playing hard exploration games by watching YouTube

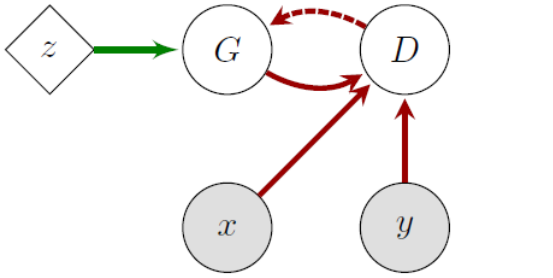



- OpenAI
- Learning Montezuma's Revenge from a Single Demonstration (blog)

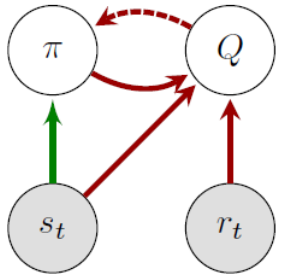


# Connecting Generative Adversarial Networks and Actor-Critic Methods

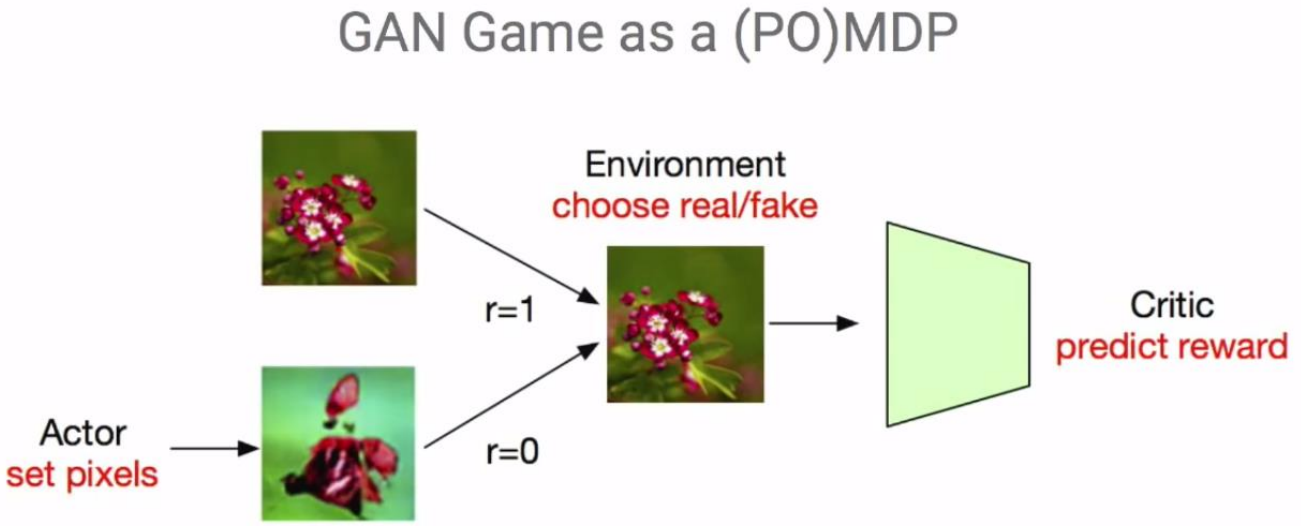
- DeepMind. NIPS 2016
- Intuition:
  - Both optimize two models with different objectives
  - Both suffer stability issues during training
  - Not a coincidence and tricks from one field may be useful to the other!




(a) Generative Adversarial Networks [4]




(b) Deterministic Policy Gradient [7] / SVG(0) [8] / Neurally-Fitted Q-learning with Continuous Actions [9]



# Stabilizing Strategies

| Method | GANs | AC |
|---|---|---|
| Freezing learning | yes | yes |
| Label smoothing | yes | no |
| Historical averaging | yes | no |
| Minibatch discrimination | yes | no |
| Batch normalization | yes | yes |
| Target networks | n/a | yes |
| Replay buffers | no | yes |
| Entropy regularization | no | yes |
| Compatibility | no | yes |

# Some other papers about GANs & RL

- A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. UC Berkeley. NIPS 2016.
- Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow. UC Berkeley. ICLR 2019, 6-10-8.

# Reference


- A Deep Learning Approach for Generalized Speech Animation. SIGGRAPH 2017.
- Generative Adversarial Imitation Learning. NIPS 2016.
  - Code: https://github.com/openai/imitation
  - Usage: https://blog.csdn.net/c2a2o2/article/details/77387637
- Connecting Generative Adversarial Networks and Actor-Critic Methods. NIPS 2016.