Generative Adversarial Imitation Learning*

Jonathan Ho, Stefano Ermon

2016

1 What

Using GAN approach for Inverse Reinforcement Learning (IRL) purposes, i.e. learning from expert behaviour without having access to the reward signal or interacting with the expert.

2 Why

Imitation Learning setting, when we want to learn from the expert data.

Two main approaches:

- behaviour cloning
- IRL

The first needs a lot of data and suffers from covariate shift. It's a supervised learning method when we fit the state-action pair mapping.

The latter finds a cost function under which the expert is uniquely optimal. Compounding error is not a problem here, but IRL is super data inefficiend and requires RL in the inner loop. Since we only interested in the final policy, why do we learn the cost function? So, GAIL was born.

3 How

The main idea is in having two models: discriminator and generator. The discriminator represents a reward function, assigning higher rewards to the expert trajectories and lower to generated ones. The goal of generator is in generating a trajectories so that their reward is maximized, i.e. the trajectories, that are close to expert trajectories. And, more formally:

$$\max_{c \in C} \big[\min_{\pi \in P} [-H(\pi) + \mathbb{E}_{\pi}(c(s,a))] - \mathbb{E}_{\pi_E}[c(s,a)] \big], \tag{1}$$

*Notes by Vitaly Kurin https://yobibyte.github.io/

where $H(\pi) \triangleq \mathbb{E}_\pi[-\log \pi(a|s)]$ is the γ-discounted causal entropy of the policy π. $\mathbb{E}_{\pi_E}(c(s,a))$ is an empirical expectation. $\arg\min_{\pi \in P}[-H(\pi) + \mathbb{E}_\pi(c(s,a))]$ is a typical RL loop.

All the tedious math leads to two insights:

- IRL is a dual of an occupancy measure matching problem
- The induced optimal policy is the primal optimum

Then the authors propose an algorithm, induced by a smart regularization function that leads to GAN-like approach.

Algorithm 1: GAIL

Input: expert trajs $\tau_E \sim \pi_E$, policy and discriminator init params θ_0, w_0

for $i \leftarrow 1$ **to** n_{epochs} **do**

Sample trajs $\tau_i \sim \pi_{\theta_i}$ Update discriminator (minimise):

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))] \tag{2}$$

Update the policy using TRPO rule with cost $\log(D_{w_{i+1}}(s,a))$, take a KL-constrained natural gradient step with

$$\begin{gathered} \hat{\mathbb{E}}_{\tau_i}[\nabla_\theta \log(\pi_\theta(a|s) Q(s,a))] - \lambda \nabla_\theta H(\pi_\theta), \\ \text{where } Q(\bar{s},\bar{a})) = \hat{\mathbb{E}}_{\tau_i}[\log(D_{wi+1}(s,a))|s_0 = \bar{s}, a_0 = \bar{a}] \end{gathered} \tag{3}$$

end

4 Evaluation

Typical classic control and MUJOCO tasks, e.g. Cartpole, Mountain Car, Hopper etc. Expert demonstrations are the trajectories generated by a trained TRPO agent.

Baselines:

- behavioural cloning
- Feature expecation matching (FEM)
- Game-theoretic apprenticeship learning (GTAL)

BC is the best for Reacher. GAIL is better on classic control tasks than FEM, GTAL, BC. Other MuJoCo envs also show boost in performance for GAIL.

5 Comments

- GAIL is efficient in terms of expert data, but not in terms of interaction with the environment (what if we init net with BC?).
- GAIL is model-free, hence less efficient in terms of env interaction, combining it with the model-based methods might be interesting.
- the further we train the generator, the more likely discriminator is to return 0.5 (since it almost impossible to distinguish), and when generator gets always constant rewards, it goes crazy
- what if generator has some (s, a) pairs which makes it think that these are generated ones (let's say he has never seen it in expert data)

References