Notes for: Generative Adversarial Imitation Learning

M. Nef. *

June 14, 2020

1 Goal

Imitation Learning is the name of the game. The end goal of GAIL is to have a way to produce a policy π_θ which tries to copy another policy π_E by observing samples of this policy.

E.g. behaviours which are hard to assign a reward to.

2 High Level

We have a discriminator which attempts to determine if a series of trajectories comes from the optimal policy π_E or from some other policy π_θ.

Once the discriminator is no longer able to distinguish the trajectories we've likely got a very similar policy.

The discriminator's confidence is used as a cost function to update the parameters θ.

3 Other approaches

3.1 Behavioural Cloning (BC)

We treat the problem as a supervised learning problem. We want a way to predict the action $\pi*$ would take given an observation.

Covariate Shift: The agent enters states which it was not exposed to during training. E.g. Car driving off the road.

Because of slight inaccuracies In BC, the policy will likey find itself in a novel state (covariate shift), it would be highly unclear what the agent might do (since it's just supervised learning). This can lead it to getting more off-course

*ipaper: https://arxiv.org/pdf/1812.10613.pdf

and into even more novel states. This is called compounding (or cascading) error.

This is because BC learns to fit single-timestep decisions as opposed to fitting to the behaviour of π_E on the whole trajectory.

3.2 Insert more alternatives here...

Todo

4 Inverse Reinforcement Learning

Produce a reward function which best explains the actions taken by an agent.

The paper uses 'Maximum Causal Entropy IRL'. Intuitively, it will find a cost function $c \in C$ which gives a low cost to the expert policy and a high cost to all other policies. After finding the cost function we can train an RL agent on it.

$$\max_{c \in C} \left(\min_{\pi \in \Pi} -H(\pi) + E_\pi[c(s,a)] \right) - E_{\pi_E}[c(s,a)]$$

(Not quite sure *why* the entropy term is in there) I'll probably need to read the maximum causal entropy IRL paper to fully understand its purpose.

Once we have this cost function which maximises this difference between all π and π_E we can just do RL to find a good policy.

4.1 Selecting C

The largest set of possible cost function $\Re^{S \times A} = \{c : S \times A \to \Re\}$. In order to prevent the cost function overfitting to our finite dataset it we need to add a convex function regularizer ψ.

The IRL procedure which finds a cost function such that the expert outperforms all other policies is now:

$$IRL_\psi(\pi_E) = c \in \Re^{S \times A} - \psi(c) + (\min_{\pi \in \Pi} -H(\pi) + E_\pi[c(s,a)]) - E_{\pi_E}[c(s,a)]$$

$$\tilde{c} \in IRL_\psi(\pi_E)$$

we want $RL(\tilde{c})$.

4.2 Occupancy Measure Matching

The occupancy measure ρ_π measures the distribution of state-action pairs that an agent encounters when navigating the environment. Since a policy uniquely

defines an occupancy measure and a valid [1] occupancy measure uniquely defines a policy

With this fact, they show:

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

Why's this important? Because it shows that the IRL procedure tries to minimise the distance between the expert's occupancy measure and the new policy as measured by the convex function ψ^*.

They come up with two conclusions: IRL is a dual of an occupancy measure matching problem. I *think* this means that occupancy matching and IRL complete the same outcome.

$$\min_{\rho \in D} -\bar{H}(\rho) \; subject \; to \; \rho(s,a) = \rho_E(s,a) \forall s \in S, a \in A$$

The induced optimal policy is the 'primal optimum' Again, I don't know what a primal optimum is but they basically say that: IRL usually is thought of as the process of finding a cost function which explains the expert policy however it can alternatively be seen as a procedure which tries to induce a policy that matches the expert's occupancy measure.

5 Apprenticeship Learning

Classic apprenticeship algorithms usually restrict C to convex sets given by linear combinations of basis functions. To clarify that. If we have a space and each basis of that space is a function. Then, every point in that space is defined by a linear combination of the basis functions.

So:

$$C_{linear} = \{\sum_i w_i f_i : ||w||_2 \leq 1\}$$

$$C_{convex} = \{\sum_i w_i f_i : \sum_i w_i = 1, w_i \geq 0 \forall i\}$$

where f_i is a "basis function" and w_i is the scalar.

The objective of an apprenticeship learning algorithm is as follows:

$$\min_\pi \max_{c \in C} E_\pi[c(s,a)] - E_{\pi_E}[c(s,a)]$$

Apprenticeship Learning is equivalent to performing RL after IRL.

If our definition of C is too restrictive and doesn't contain the real cost function of the expert then it's pretty obvious that we can't recover a policy which matches the expert. In this case there's no guarantee that π_E will outperform π for each cost function in C.

[1] Positive probabilities + some more constraints

Apprenticeship objective optimisation Since we have a paramaterised policy π_θ the policy gradient formula will be:

$= \nabla_\theta \max_{c \in C} E_{\pi_\theta}[c(s,a)] - E_{\pi_E}[c(s,a)]$

$= \nabla_\theta E_{\pi_\theta}[c^*(s,a)]$

$= E_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q_{c^*}(s,a)]$

where:

$c^* = \arg\max_{c \in C} E_{\pi_\theta}[c(s,a)] - E_{\pi_E}[c(s,a)]$

$Q_{c^*}(\bar{s}, \bar{a}) = E_{\pi_\theta}[c^*(\bar{s}, \bar{a})|s_0 = \bar{s}, a_0 = \bar{a}]$

This is just an RL objective with cost c^*

SOooO, here's what we can do:

1. Sample some trajectories from the current policy π_θ
2. Fit a c^* (shown above). In the case where we have C_{linear} or C_{convex} "this cost fitting amounts to evaluating simple analytical expressions".
3. Use TRPO to produce $\pi_{\theta_{i+1}}$

6 GAIL

The first thing the authors do is create a new regulariser:

$$\psi_{GA}(C) := \begin{cases} E_{\pi_E}[g(c(s,a))] & c < 0 \\ +\infty & \text{otherwise} \end{cases}$$

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

https://www.desmos.com/calculator/osicwgrejv

- Low penalty on cost functions that assign negative cost to expert state-action pairs.
- High penalty on cost functions that assign high (close to 0) cost to expert state-action pairs.

They then show that the regulariser was selected because:

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

Is the negative log loss of the binary classification problem of distinguishing between state-action pairs of π and π_E. They show that this optimal loss is very similar to the Jesnon-Shannon divergence.

$$\min_\pi \psi_{GA}(\rho_\pi - \rho_{\pi_E}) - \lambda H(\pi) = D_{JS}(\rho_\pi, \rho_{\pi_E}) - \lambda H(\pi)$$

.

This connects imitation learning and GANS. Since we train a generative model by having it confuse a discriminator. ρ_π is the data distribution "generated" and the expert's, ρ_{π_E} is the true distribution.

7 Final Algorithm

π_θ parameterised policy (dnn)

$D_w : S \times A \to (0, 1)$ parameterised discriminator

Now, we alternate between an Adam gradient step on w and a TRPO step on θ.

This can be intuitively thought as the discriminator being a cost function, as the TRPO algorithm decreases this "cost" it's moving towards parameter spaces where the discriminator is unable to distinguish it from the expert.