Maximum Entropy Inverse Reinforcement Learning and Generative Adversarial Imitation Learning

Sanket Gujar

Worcester Polytechnic Institute, MA, United States

Abstract

Instead of relying on drivers to communicate their intentions, which they often will not or cannot do, we take the opposite perspective; the navigation system itself should learn to predict the intentions and future behaviors of the driver based on past observations and the current situation. One approach to route prediction is to assume the driver will also try to take this most expedient route.

Two important applications of this problem are route recommendation, where a driver requests a desirable route connecting two specified points, and unanticipated hazard warning, where an application can predict the driver will encounter some hazard he is unaware of and provide a warning beforehand so that the hazard can be avoided. We explore the problem of Imitation learning with two popular approaches: Maximum Entropy Inverse Reinforcement Learning and Generative Adversarial Imitation Learning and compare the advantages and drawbacks of each.

Keywords: Inverse Reinforcement Learning, Entropy

1. Literature Survey

1.1. ***Maximum Entropy Inverse Reinforcement Learning*** *[1]*

Uses the principle of Maximum entropy to resolve ambiguity when each policy can be optimal for reward function, and many policies lead to the same feature count in IRL and feature matching. Maximize the Log likelihood over the demonstrations (Expert collected data).

Notations:

τ : Trajectories $(s_1, a_1, s_2, ...s_\tau)$

τ_D : Trajectories collected from expert.

$r_\theta(\tau)$: Reward with parameter θ for trajectory τ

The reward of a trajectory is expressed as linearly combinations with feature counts. $r(\tau) = \theta^T f_\tau$

The constraint to solve this problem is

$$\sum_\tau p_\tau f_\tau = \overline{f}$$

and we have to maximise the log likelihood,

$$maximise \sum_\tau p(\tau) \log p(\tau)$$

Now taking Lagrange to solve this optimization problem.

$$d[-\sum_\tau p(\tau) \log p(\tau) - \lambda(\sum_\tau p_\tau f_\tau - \overline{f}) - \mu(\sum_\tau P_\tau - 1)] = 0$$

$$[-\sum_\tau \log p(\tau) - \sum_\tau 1 - \lambda(\sum_\tau f_\tau) - \mu(\sum_\tau 1]d(p) = 0$$

$$\sum_\tau [\log p(\tau) + 1 + \lambda(f_\tau) + \mu]d(p) = 0$$

$$let \mu + 1 = \lambda_0$$

$$\sum_\tau [\log p(\tau) + \lambda(f_\tau) + \lambda_0]d(p) = 0$$

Since this is zero for all probability distribution, all τ braces should be zero

$$\log p(\tau) + \lambda(f_\tau) + \lambda_0 = 0$$

$$\log p(\tau) = -\lambda_0 - \lambda f_\tau$$

$$p(\tau) = e^{-\lambda_0} e^{-\lambda f_\tau}$$

The sum of probabilities should be equal to 1

$$\sum_\tau p(\tau) = 1 = \sum_\tau e^{-\lambda_0} e^{-\lambda f_\tau}$$

Now we get

$$e^{-\lambda_0} = \frac{1}{\sum_\tau e^{-\lambda f_\tau}}$$

let

$$Z = \sum_\tau e^{-\lambda f_\tau}$$

Z is the Partition Function

$$e^{-\lambda_0} = \frac{1}{Z}$$

Now, we have the constraint

$$\sum_\tau p_\tau f_\tau = \overline{f}$$

Substituting the probability value

$$\overline{f} = \frac{1}{Z} \sum_\tau f_\tau e^{-\lambda f_\tau}$$

When we take derivative of Z

$$\frac{dZ}{d\lambda} = -\sum_\tau e^{-\lambda f_\tau} f_\tau$$

Substituting the derivative of Z in the equation

$$\overline{f} = -\frac{1}{Z} \frac{dZ}{d\lambda}$$

$$\overline{f} = -\frac{d \log Z}{d\lambda}$$

Now if we solve this we get,

$$\overline{f} = f$$

Principle of Maximum entropy (Jaynes 1957): Probability of a demonstrated trajectory is proportional to its exponential of reward of the trajectory.

$$p(\tau) \propto exp(r(\tau))$$

And the **objective** is to maximise the log likelihood of the demonstrated trajectories.

$$\theta^* = argmax_\theta L(\theta) = argmax_\theta \frac{1}{m} \sum_{\tau_d \in D} \log p(r(\tau_d))$$

$$\theta^* = argmax_\theta \frac{1}{m} \sum_{\tau_d \in D} \log \frac{1}{Z} e^{r(\tau_d)}$$

Where Z is the partition function $Z = \sum_\tau e^{r(\tau)}$ and m is the total number of trajectories. So Now,

$$\theta^* = argmax_\theta \frac{1}{m} \sum_{\tau_d \in D} (r(\tau_d) - \log Z)$$

Taking the derivative of the objective function as it is convex:

$$\frac{dL(\theta)}{d\theta} = \frac{1}{m} (\sum_{\tau_d} \frac{dr(\tau_d)}{d(\theta)} - \frac{1}{\sum_\tau e^{r(\tau)}} \sum_\tau e^{r(\tau)} \frac{r(\tau)}{\theta})$$

Now here if we take the summation inside we get

$$p(\tau) = \frac{e^{r(\tau)}}{\sum_\tau e^{r(\tau)}}$$

$$\frac{dL(\theta)}{d\theta} = \frac{1}{m}(\sum_{\tau_d} \frac{dr(\tau_d)}{d(\theta)} - \sum_{\tau} p(\tau) \frac{dr(\tau)}{d\theta})$$

We can change the trajectory with the states, we get

$$\frac{dL(\theta)}{d\theta} = \frac{1}{m}(\sum_{s\in\tau_d} \frac{dr(s_d)}{d(\theta)} - \sum_{s\in\tau} p(s|\theta, T) \frac{dr(s)}{d\theta})$$

This equations hold for any dimension of reward function,
But for Linear reward function : $r(\tau) = \theta f_\tau$
So, $\frac{dr(\tau)}{d\theta} = f_\tau$
So Now the equations becomes for linear reward function,

$$\frac{dL(\theta)}{d\theta} = \overline{f} - \sum_{s\in\tau} \frac{p(s|\theta, T)}{m} f_s$$

Here $\sum_{s\in\tau} \frac{p(s|\theta,T)}{m}$ is the state visitation which cab be denoted by D_s. So now the derivative reduces to

$$\frac{dL(\theta)}{d\theta} = \overline{f} - \sum_{s\in\tau} D_s f_s$$

We can use Dynamic Programming to calculate the state visitation frequency for the given optimal policy $\pi(a, s)$ and the transition matrix $P_{sa}(s')$. We can use μ_t to denote the probability of visiting state s at time t.
for t = 1,2,.....,T.

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \mu_t(s') \pi(a, s') P_{sa}(s')$$

and,

$$P(s) = \sum_t \mu_t(s)$$

The algorithm solves MDP in each iteration of training and assumes known dynamics but it can scale to non-linear cost for neural networks.

Algorithm

1. Initialize θ, gather demonstration D
2. Solve for Optimal policy $\pi(a,s)$ w.r.t reward $r(\tau)$
3. Solve for state visitation frequency $p(s|\theta)$
4. Compute the gradient $\nabla_\theta L = -\frac{1}{m}(\sum_{s\in\tau_d}\frac{dr(s_d)}{d(\theta)} + \sum_{s\in\tau} p(s|\theta,T)\frac{dr(s)}{d\theta})$
5. Update θ with one gradient step using $\nabla_\theta L$
6. Repeat from step 2

Unknown Dynamics and Continuous Space (Sampling) [2]
Important Sampling

$$\sum_{\tau_s\sim q}\frac{e^{r(\tau_s)}}{q(\tau_s)}$$

To choose the distribution we can sample from, we need minimum variance of the estimator.

$$minvar : q(\tau) \propto |e^{r(\tau)}|$$

So now the objective can be defined as,

$$q = argmax_q E_q[r(\tau)] - H(q)$$

We can use mixture sampling for various optimal distribution of q

$$v(\tau) = \frac{1}{k}\sum_k q_k(\tau)$$

So now the Objective function becomes,

$$\theta^* = argmax_\theta \frac{1}{m}\sum_{\tau_d\in D}(r(\tau_d) - \log\sum_{\tau_s}\frac{e^{r(\tau_s)}}{v(\tau)})$$

*1.2. **A Connection Between Generative Adversarial Networks,Inverse Reinforcement Learning, and Energy-Based Models** (Finn et Al. 16) [3]*

Demonstrate an equivalence between a sample based algorithm for maximum entropy IRL [1] and a GAN [4] in which the generator density can be evaluated and is provided as an additional input to the discriminator.

For a fixed generator with a density $q(\tau)$, and actual data distribution $p(\tau)$ the optimal discriminator is as follows,

$$D^*(\tau) = \frac{p(\tau)}{p(\tau) + q(\tau)}$$

When the generator density $q(\tau)$ can be evaluated, the discriminator can be modified to only can estimate the value of $p(\tau)$. In this case the discriminator modifies to

$$D_\theta(\tau) = \frac{\overline{p_\theta}(\tau)}{\overline{p_\theta}(\tau) + q(\tau)}$$

To make a connection to MaxEnt IRL we can replace the estimated data density with the Boltzmann distribution. Now we the discriminator's output is,

$$D_\theta(\tau) = \frac{\frac{1}{Z}e^{-c_\theta(\tau)}}{\frac{1}{Z}e^{-c_\theta(\tau)} + q(\tau)}$$

The Discriminator Loss is

$$L_{discriminator}(D_\theta) = E_{\tau\sim p}[-\log D_\theta(\tau)] + E_{\tau\sim q}[-\log 1 - D_\theta(\tau)]$$

$$L_{discriminator}(D_\theta) = E_{\tau\sim p}[-\log \frac{\frac{1}{Z}e^{-c_\theta(\tau)}}{\frac{1}{Z}e^{-c_\theta(\tau)} + q(\tau)}] + E_{\tau\sim q}[-\log \frac{q(\tau)}{\frac{1}{Z}e^{-c_\theta(\tau)} + q(\tau)}]$$

We can write the objective of Maximum entropy IRL as,

$$L_{cost}(D_\theta) = E_{\tau\sim p}[c_{\theta(\tau)}] + \log(E_{\tau\sim\frac{1}{p}+\frac{1}{q}}[\frac{e^{-c_\theta(\tau)}}{\frac{1}{2}\overline{p}_(\tau) + \frac{1}{2}q(\tau)}])$$

We will substitute $\overline{p}(\tau) = p_\theta(\tau) = \frac{1}{Z}e^{-c_\theta(\tau)}$, as we are using the current model to estimate the importance weights.

$$L_{cost}(D_\theta) = E_{\tau\sim p}[c_{\theta(\tau)}] + \log(E_{\tau\sim\mu}[\frac{e^{-c_\theta(\tau)}}{\frac{1}{2Z}e^{-c_\theta(\tau)} + \frac{1}{2}q(\tau)}])$$

The value of Z which minimizes the discriminator's loss is an importance-sampling estimate for the partition function, so the derivative of the discriminator's loss w.r.t θ is equal to derivative of MaxIRL objective.

When θ and Z are optimized, $\frac{1}{Z}e^{-c_\theta(\tau)}$ is an estimate for the density of $p(\tau)$. Let $\mu = \frac{p}{2} + \frac{q}{2}$ be the mixture distribution over trajectory. So we can write

$$\overline{\mu}_\tau = \frac{1}{2Z}e^{-C_\theta(\tau)} + \frac{1}{2}q(\tau)$$

So Now the discriminators loss becomes,

$$L_{discriminator}(D_\theta) = E_{\tau\sim p}[-\log \frac{\frac{1}{Z}e^{-c_\theta(\tau)}}{2\overline{\mu}_\tau}] + E_{\tau\sim q}[-\log \frac{q(\tau)}{2\overline{\mu}_\tau}]$$

$$L_{discriminator}(D_\theta) = \log Z + E_{\tau\sim p}[c_\theta(\tau)] + 2E_{\tau\sim p}[\log 2\overline{\mu}(\tau)] - E_{\tau\sim q}[\log q(\tau)]$$

To Find the minimum value of Z, we will take the derivative of Loss w.r.t Z will be 0.

$$\frac{dL_{discriminator}(D_\theta)}{dZ} = \frac{1}{Z} + 2E_{\tau\sim\overline{\mu}}[\frac{1}{2\overline{\mu}(\tau)}\frac{2d\overline{\mu}_(\tau)}{dZ}] = 0$$

$$\frac{dL_{discriminator}(D_\theta)}{dZ} = \frac{1}{Z} + 2E_{\tau\sim\overline{\mu}}[\frac{1}{\overline{\mu}(\tau)}\frac{-1}{Z^2}e^{-c_\theta}(\tau)] = 0$$

$$\frac{1}{Z} = 2E_{\tau\sim\overline{\mu}}[\frac{\frac{1}{Z^2}e^{-c_\theta}(\tau)}{\overline{\mu}(\tau)}]$$

$$Z = 2E_{\tau\sim\overline{\mu}}[\frac{e^{-c_\theta}(\tau)}{\overline{\mu}(\tau)}]$$

Now if we differentiate the equation w.r.t to θ,

$$\frac{dL_{discriminator}(D_\theta)}{d\theta} = E_{\tau\sim p}[\frac{dc_\theta(\tau)}{d\theta}] - E_{\tau\sim\mu}[\frac{\frac{1}{Z}e^{-c_\theta(\tau)}\frac{dc_\theta(\tau)}{d\theta}}{\overline{\mu}_\tau}]$$

Now if We differentiate the MaxIRL objective,

$$L_{cost}(D_\theta) = E_{\tau\sim p}[c_{\theta(\tau)}] + \log(E_{\tau\sim\mu}[\frac{e^{-c_\theta(\tau)}}{\frac{1}{2Z}e^{-c_\theta(\tau)} + \frac{1}{2}q(\tau)}])$$

$$\frac{dL_{cost}(D_\theta)}{d\theta} = E_{\tau\sim p}[\frac{dc_{\theta(\tau)}}{d(\theta)}] + \frac{d}{d\theta}\log(E_{\tau\sim\mu}[\frac{e^{-c_\theta(\tau)}}{\overline{\mu}(\tau)}])$$

$$\frac{dL_{cost}(D_\theta)}{d\theta} = E_{\tau\sim p}[\frac{dc_{\theta(\tau)}}{d(\theta)}] + \frac{E_{\tau\sim\mu}[\frac{-e^{-c_\theta(\tau)}}{\overline{\mu}(\tau)}\frac{dc_{\theta(\tau)}}{d(\theta)}]}{E_{\tau\sim\mu}[\frac{e^{-c_\theta(\tau)}}{\overline{\mu}(\tau)}]}$$

Now $Z = E_{\tau\sim\mu}[\frac{e^{-c_\theta(\tau)}}{\overline{\mu}(\tau)}]$

$$\frac{dL_{cost}(D_\theta)}{d\theta} = E_{\tau\sim p}[\frac{dc_{\theta(\tau)}}{d(\theta)}] - E_{\tau\sim\mu}[\frac{\frac{1}{Z}e^{-c_\theta(\tau)}}{\overline{\mu}(\tau)}\frac{dc_{\theta(\tau)}}{d(\theta)}]$$

Which is exactly equal to the discriminator loss we derived above.
Now if we go to see the generators loss,

$$L_{generator}(q) = E_{\tau\sim q}[\log(1 - D(\tau)) - \log(D(\tau))]$$

$$L_{generator}(q) = E_{\tau\sim q}[\log(\frac{q(\tau)}{\overline{\mu}(\tau)}) - \log(\frac{\frac{1}{Z}e^{-c_\theta(\tau)}}{\overline{\mu}(\tau)})]$$

$$L_{generator}(q) = E_{\tau\sim q}[\log q(\tau)] + \log Z + E_{\tau\sim q}[c_\theta(\tau)]$$

We know,

$$L_{sampler}(q) = E_{\tau\sim q}[\log q(\tau)] + E_{\tau\sim q}[c_\theta(\tau)]$$

$$L_{generator}(q) = \log Z + L_{sampler}(q)$$

The term log Z is kept as a fixed parameter of the discriminator while optimizing the generator.

1.3. ***Generative Adversarial Imitation Learning*** *(Jo & Ermon) [5]*

Derives a model-free imitation learning algorithm which uses generative adversarial training to fit distributions of states and actions defining expert behavior. Uses maximum casual entropy IRL [1] which fits a cost function C with the optimization problem

Harnesses generative adversarial training to fit the distribution of states and actions defining expert behavior.

Max. Entropy IRL

for a cost function $c \in C$ that assign low cost to expert policy and high cost to other policies. It finds the expert policy via a certain reinforcement learning procedure which maps a cost function to high-entropy policies that minimizes the expected cumulative cost.

$$RL(c) = argmin_{\pi} - H(\pi) + E_{\pi}[c(s,a)]$$

The IRL can easily overfit when provided a finite dataset when it has a large environment with its capabilities $C = R^{S \times A}$. So we incorporate a convex cost function regularizer $\psi : R^{S \times A} \rightarrow \overline{R}$

Now the IRL cost function with the cost regularized by ψ becomes,

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

Let $\overline{c} \in IRL_{\psi}(\pi_E)$ $RL(\overline{c})$ is the policy given by running reinforcement learning on the output of IRL.So a policy occupancy measure can be given as $p_{\pi}(s,a) = \pi(a,s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$. The occupancy measure can be interpreted as the unnormalized distributions of state-action pairs that an agent encounters while navigating the environment.

In reality, the expert trajectory will be provided only as finite set of samples, so in large environments, most of the expert's occupancy measure will be small, and exact occupancy measure matching will force the learned policy to rarely visit these unseen state-action pair simply to lack of data.

The Regularizer

The problems with regularizers are :

- Constant regularizer leads to an imitation learning that exactly matches occupancy measures but intractable in large environments.
- The indicator regularizers [6] leads to an algorithm incapable of exactly matching without careful tuning, but tractable in large environments.

The proposed cost regularizer combines the both objective:

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

where,

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

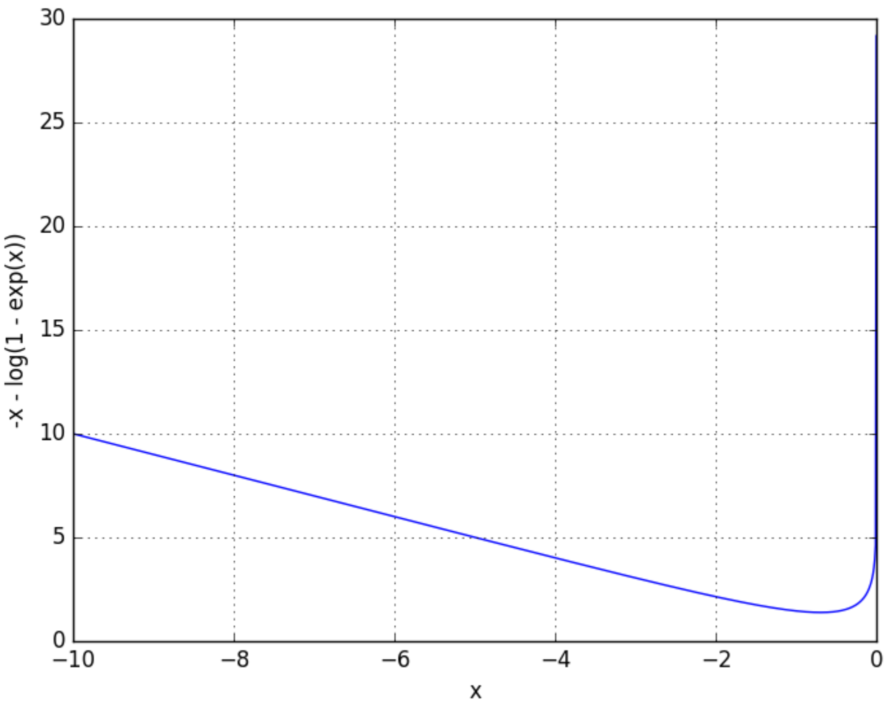

Figure 1: Plot of g(x)

The regularizer places low penalty on cost functions c that assigns an amount of negative cost to expert state-action pairs. If c however assigns large cost (close to zero) to the expert then ψ_{GA} will heavily penalize c.

$$\psi_{GA}(p_\pi - p_{\pi_E}) = sup_{D \in (0,1)^{s \times A}} E_\pi[\log(D(s,a))] + E_{\pi_E}[\log(1 - D(s,a))]$$

derivation

we have to convert surrogate loss function ϕ for binary classification state-action pair drawn from the occupancy measure p_π and p_{π_E}, into cost regularizers ψ, for which $\psi^*(p_\pi - p_{\pi_E})$ is the minimum expected risk $R_\phi(p_\pi - p_{\pi_E})$ for ϕ

$$R_\phi(\pi, \pi_E) = \sum_{s,a} \min_{\gamma \in R} p_\pi(s,a)\phi(\gamma) + p_{\pi_E}(s,a)\phi(-\gamma)$$

So it can generate any imitation learning algorithm that minimizes an f-divergence between occupancy measures, as long as that f-divergence is induced by a strictly decreasing convex surrogate ϕ

We are assuming ϕ to be convex. Let T be the range of -ϕ.

$$g(x) = \begin{cases} -x - \phi(-\phi^{-1}(-x)) & \text{if} \quad x \in T \\ +\infty & \text{otherwise} \end{cases}$$

and

$$\psi_\phi(c) = \begin{cases} \sum_{s,a} p_{\pi_E}(s,a) g_\phi(c(s,a)) & \text{if} \quad c(s,a) \in T \text{for all s,a} \\ +\infty & \text{otherwise} \end{cases}$$

and

$$RL(IRL_{\psi_\phi}(\pi_E)) = arg \min_\pi -H(\pi) - R_\phi(p_\pi, p_{\pi_E})$$

Now

$$\psi^*(p_\pi, p_{\pi_E}) = \max_{cinC} \sum_{s,a} (p_\pi(s,a) - p_{\pi_E}(s,a))c(s,a) - \sum_{s,a} p_\pi(s,a) g_\phi(c(s,a))$$

$$\psi^*(p_\pi, p_{\pi_E}) = \max_{cinC} \sum_{s,a} (p_\pi(s,a) - p_{\pi_E}(s,a))c(s,a) - \sum_{s,a} p_\pi(s,a)[-c + \phi(-\phi^{-1}(-c))]$$

$$\psi^*(p_\pi, p_{\pi_E}) = \max_{cinC} \sum_{s,a} (p_\pi(s,a))c(s,a) - \sum_{s,a} p_\pi(s,a)(\phi(-\phi^{-1}(-c)))$$

We put $c(s,a) \to -\phi(\gamma)$

$$\psi^*(p_\pi, p_{\pi_E}) = \max_{cinC} \sum_{s,a} (p_\pi(s,a))(-\phi(\gamma)) - \sum_{s,a} p_\pi(s,a)(\phi(-\phi^{-1}(\phi(\gamma))))$$

$$\psi^*(p_\pi, p_{\pi_E}) = \max_{cinC} \sum_{s,a} (p_\pi(s,a))(-\phi(\gamma)) - \sum_{s,a} p_\pi(s,a)(\phi(-\gamma))$$

$$\psi^*(p_\pi, p_{\pi_E}) = -R_\phi(p_\pi, p_{\pi_E})$$

Here we obtained a corollary, a cost function regularizer for the logistic loss, whose optimal expected risk is, up to a constant, the Jensen-Shannon divergence.

Using the logistic loss $\phi(x) = \log(1 + e^{-x})$ So Now,

$$\psi^*(p_\pi, p_{\pi_E}) = \sum_{s,a} \max_\gamma (p_\pi(s,a))(-\log(1+e^{-\gamma})) - \sum_{s,a} p_\pi(s,a)(\log(1+e^{\gamma}))$$

$$\psi^*(p_\pi, p_{\pi_E}) = \sum_{s,a} \max_\gamma (p_\pi(s,a))(\log(\frac{1}{1+e^{-\gamma}})) + \sum_{s,a} p_\pi(s,a)(\log(\frac{1}{1+e^{\gamma}}))$$

$$\psi^*(p_\pi, p_{\pi_E}) = \sum_{s,a} \max_\gamma (p_\pi(s,a))(\log(\frac{1}{1+e^{-\gamma}})) + \sum_{s,a} p_\pi(s,a)(\log(1 - \frac{1}{1+e^{-\gamma}}))$$

Where the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ has range of (0,1)

$$\psi^*(p_\pi, p_{\pi_E}) = \sum_{s,a} \max_\gamma (p_\pi(s,a))(\log(\sigma(\gamma)) + \sum_{s,a} p_\pi(s,a)(\log(1-\sigma(\gamma)))$$

$$\psi^*(p_\pi, p_{\pi_E}) = \sum_{s,a} \max_{d\in(0,1)} (p_\pi(s,a))(\log(d)) + \sum_{s,a} p_\pi(s,a)(\log(1-d))$$

$$\psi^*(p_\pi, p_{\pi_E}) = \max_{D\in(0,1)^{(s,a)}} \sum_{s,a} (p_\pi(s,a))(\log(D(s,a)) + p_\pi(s,a)(\log(1-D(s,a)))$$

which is the desired equation.

The observation from the equation are:

- The equation is proportional to the optimal negative log loss of the binary classifier problem of distinguishing between state-action pair of π and π_E.

- Optimal loss is up to a constant shift and scaling the Jensen-Shannon divergence $D_{JS}(p_\pi, p_{\pi_E}) = D_{KL}(p_\pi || (p_\pi + p_{\pi_E})/2) + D_{KL}(p_{\pi_E} || (p_\pi + p_{\pi_E})/2)$

Now treating the casual entropy H as a policy regularizer controlled by $\lambda >= 0$ we get the objective

$$minimize_\pi \psi^*_{GA}(p_\pi - p_{\pi_E}) - \lambda H(\pi) = D_{JS}(p_\pi, p_{\pi_E}) - \lambda H(\pi)$$

which finds a policy whose occupancy measure minimizes Jensen-Shannon divergence to the expert's policy.

Algorithm The equation draws a connection between imitation learning and generative adversarial network. The learner's occupancy measure p_π is analogous to the data distribution generated by G, and the expert's occupancy measure p_{π_E} is analogous to true data distribution.

GAIL

```
1. Input : Expert trajectories $\tau_E \sim \pi_E$, initial policy and discriminators policy $\theta_0, w_0$
2. for i in range ($no.iterations$) do
3.     Sample trajectories $\tau_i \sim \pi_{\theta_i}$
4.     Update the discriminator from $wi$ to $wi + 1$ with the gradient
           $E_{\tau_i}[\nabla_w \log(D_w(s,a))] + E_{\tau_E}[\nabla_w \log(1 - D_w(s,a))]$
5.     Take a policy step from $\theta_i$ to $\theta_{i+1}$ using the TRPO rule with the cost function $\log(D_{w_{i+1}}(s,a))$ Specifically take a KL-constrained natural gradient step with,
           $E_{\tau_i}[\nabla_\theta \log(\pi_\theta(a|s)Q(s,a)] - \lambda \nabla_\theta H(\pi_\theta)$
           where Q(s',a') $= E_{\tau_i}[\log(D_w(s,a))|s_0 = s', a_0 = a']$
6. End for
```

Role of TRPO[7] : Prevents the policy from changing too much due to noise in the policy gradients.

2. Experiments

The experiments were carried out in Gridworld (discrete) and Cartpole (continous) environment.

2.1. ***Gridworld***

The environment has discrete states and discrete actions. The gridsize is variable, but was fixed to 20 due to limit of computational capacity.

The result of recovered reward function by Maximum Entropy Inverse reinforcement learning can be seen in figure 2. Here, we can see the Maximum entropy IRL recovered the exact cost function as the expert, but it is a problem when we scale to large environment, when the agent enter an unseen state before it will take improper actions. MEIRL fits the exact reward function but we need to experiment more for larger environments and continuous states.

The result of recovered reward function by Generative Adversarial Imitation learning can be seen in figure 3. Here, we can see the GAIL didn't recover the exact cost function as the expert, so it will be also scale to large environment. So, when the agent enter an unseen state, it will not take improper actions. GAIL doesn't fits the exact reward function but we need to experiment more for larger environments and continuous states. Also the value near the start states are unexplainable. We assume that the network might have learned a symmetric function which gave high value to the start and goal position. It may also be that the GAN would be ineffective to sample discrete states.

Distributed reward Gridworld

The result of recovered reward function by Generative Adversarial Imitation learning for continuous reward can be seen in figure 4. The value near the top-left states are unexplainable. We assume its due to unexplored states of the expert agent. The expert's policy takes the boundary path to the left and to the top, which was captured by GAIL with regularization. Although it was not able to assign high value to the goal positions.It may also be that the GAN would be ineffective to sample discrete states. We need to experiment more with GAIL for distributed reward with a robust optimal expert policy.

2.2. ***Cartpole***

The Cartpole environment (figure: 5) was used to test Generative Adversarial Imitation Learning on continuous state space. In the environment

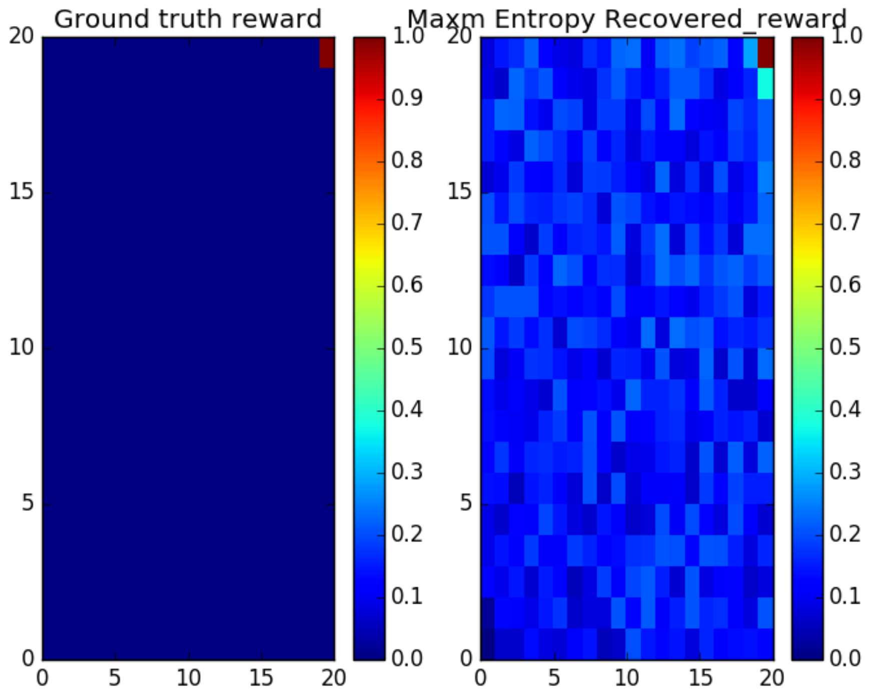

Figure 2: Maximum Entropy Inverse Reinforcement Learning (Gridworld) : a) The actual reward function b) Maximum Entropy Inverse Reinforcement Learning recovered reward

a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center.

For the experiment, a expert policy was trained on Cartpole environment using Proximal Policy Optimization(PPO)[8], once the expert was trained, the state and action of the expert were captured. For training GAIL, a agent policy (generator) was defined similar to the trained policy. The discriminator was trained using positive samples from the expert demonstrations and the negative sample from the generator state action pair. The generator was trained using Proximal Policy Optimization from the reward he received from

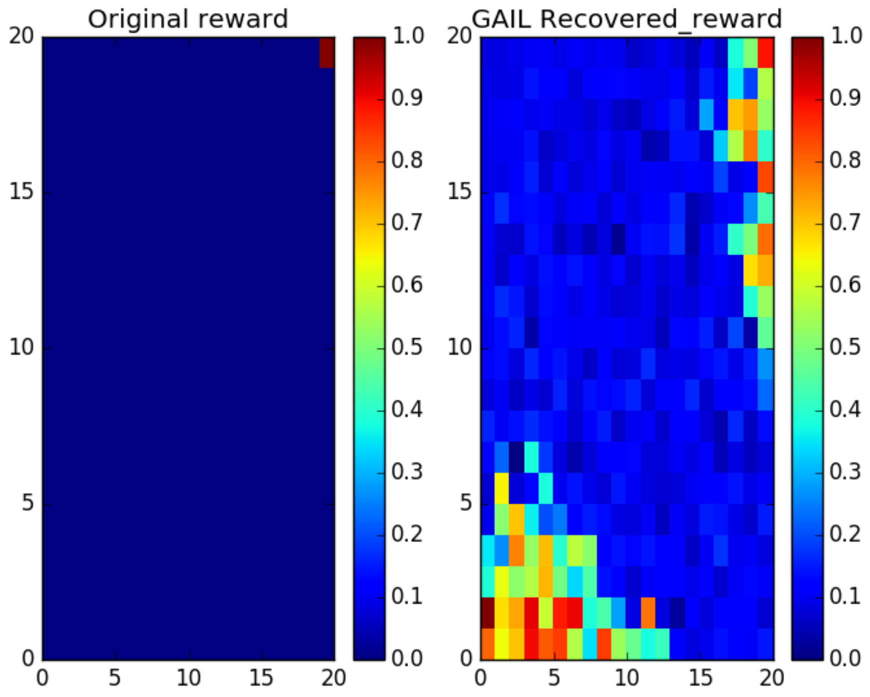

Figure 3: Generative Adversarial Imitation Learning(Gridworld) : a) The actual reward function b) Generative Adversarial Imitation Learning recovered reward

the discriminator, as how close it policy was to the expert's policy. The role of Proximal Policy Optimization is just to avoid policy changing too much due to noise in policy gradients. GAIL was able to recover the expert policy in the continuous state space (figure 6, 7).

3. References

[1] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning., in: AAAI, volume 8, Chicago, IL, USA, pp. 1433–1438.

[2] C. Finn, S. Levine, P. Abbeel, Guided cost learning: Deep inverse optimal control via policy optimization, in: International Conference on Machine Learning, pp. 49–58.

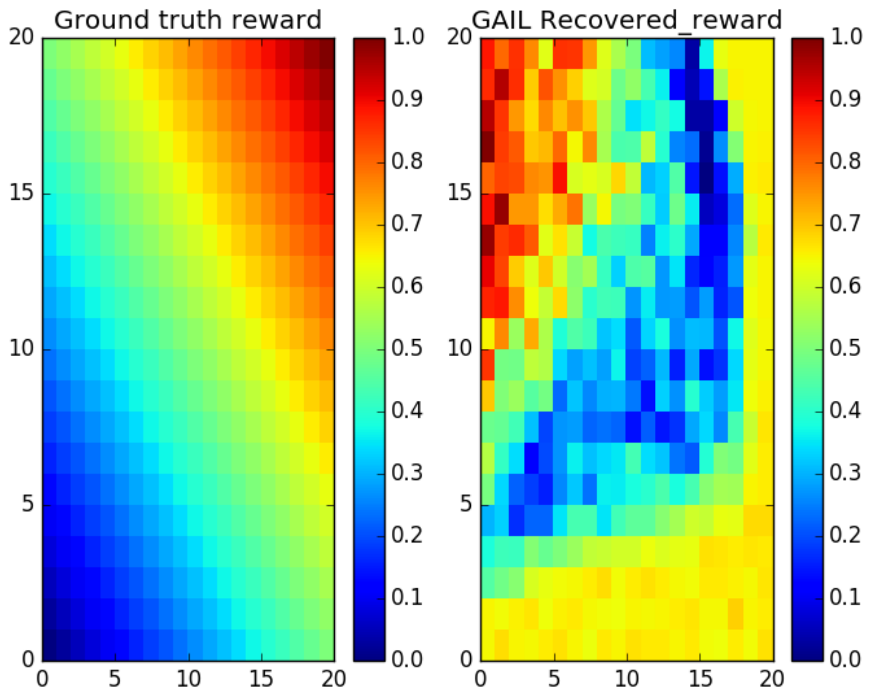

Figure 4: Generative Adversarial Imitation Learning(Distributed reward Gridworld) : a) The actual reward function b) Generative Adversarial Imitation Learning recovered reward

[3] C. Finn, P. Christiano, P. Abbeel, S. Levine, A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models, arXiv preprint arXiv:1611.03852 (2016).

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, pp. 2672–2680.

[5] J. Ho, S. Ermon, Generative adversarial imitation learning, in: Advances in Neural Information Processing Systems, pp. 4565–4573.

[6] P. Abbeel, A. Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the twenty-first international conference on Machine learning, ACM, p. 1.

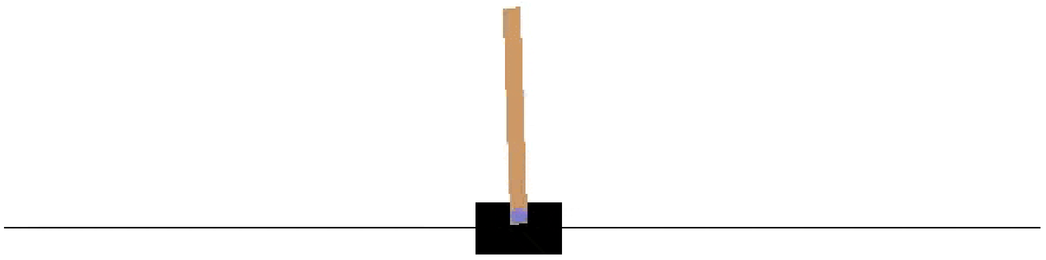

Figure 5: Cartpole environment (OpenAI gym)

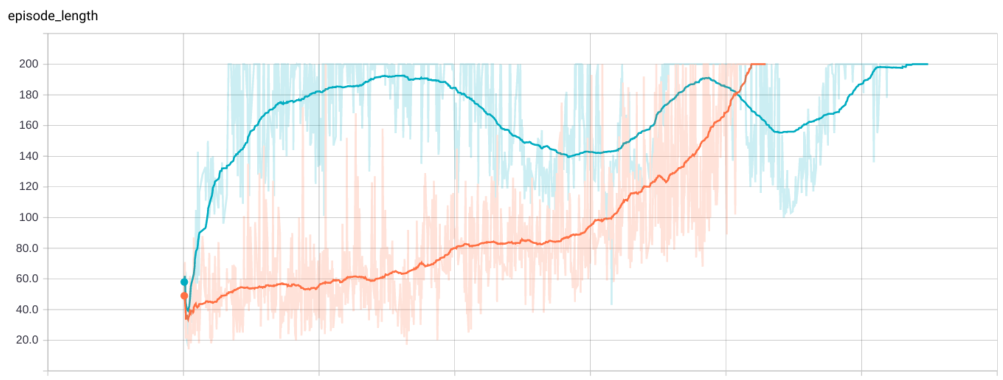

Figure 6: Generative Adversarial Imitation Learning (Cartpole) :Episode lenght (red: Agent policy, blue: GAIL)

[7] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning, pp. 1889–1897.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

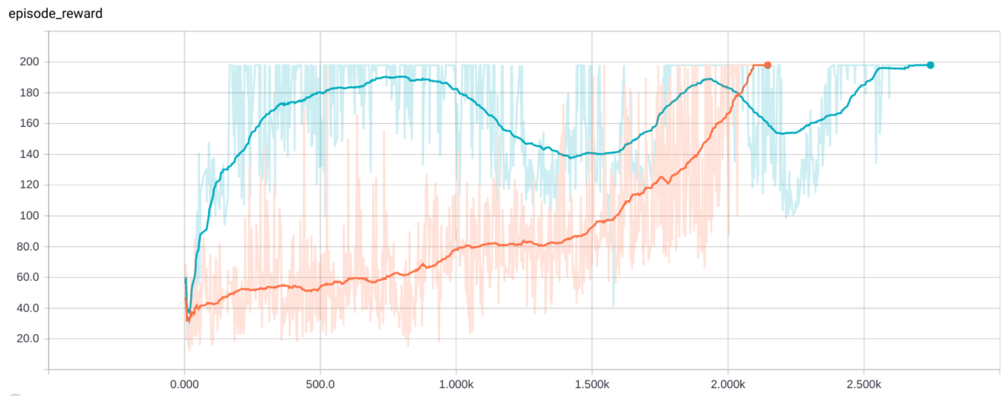

Figure 7: Generative Adversarial Imitation Learning (Cartpole) :Episode reward (red: Agent policy, blue: GAIL)