Generative Adversarial Imitation Learning

Jatin Arora (NetID: `jatin2`)

1 Introduction

This project report intends to provide an understanding of Generative Adversarial Imitation Learning[1]. We first develop the theoretical foundations of the concept and then present an empirical study.

As the name suggests, the theme here is imitation learning in which the goal is to make an agent learn a policy that closely matches expert behavior. In the current setting, we are given some trajectories sampled from the expert but we cannot interact with the expert or get any other reinforcement signals. Standard technique is a two step process. First, recover a cost(reward) function that captures expert behavior from expert trajectories. Then, design a Markov decision process (MDP) and use the recovered cost function for learning an optimal policy on it. However, in this work, the authors demonstrate the effectiveness of a model-free approach that directly interacts with expert data and learns a policy, skipping the intermediate step of recovering the cost function.

1.1 Notation

We define a MDP as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma, p_0)$ where,

$\mathcal{S}$ is the finite state space.

$\mathcal{A}$ is the finite action space.

$P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is transition function. $P(s'|s, a)$ is probability of reaching state $s' \in \mathcal{S}$ when at $s \in \mathcal{S}$ and taking action $a \in \mathcal{A}$.

$R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is reward function. Alternatively, define cost function $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Maximizing reward corresponds to minimizing cost. $c(s, a)$ is cost incurred by taking action a in state s.

γ is the discount factor.

$p_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution.

Stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ captures agent behavior. $\pi(s, a)$ is probability of taking action a in state s. Π is the family of all stochastic policies. We work in γ-discounted infinite horizon setting where,

$$\mathbb{E}_\pi[c(s,a)] \triangleq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t c(s_t,a_t)] \tag{1}$$

such that $s_0 \sim p_0$, $a_t \sim \pi(.|s_t)$, $s_{t+1} \sim P(.|s_t,a_t)$ for all $t \geq 0$.

2 Theoretical Study

2.1 Inverse reinforcement learning

In reinforcement learning (RL) setting, we are given a cost function and our task is to learn an optimal policy. In inverse reinforcement learning (IRL), we use expert demonstrations to infer a cost function as well as a corresponding policy consistent with expert behavior. IRL typically involves a RL step ultimately to recover the policy from the inferred cost function. As mentioned in [2] and [3], it is worth pointing out that recovering the exact cost function from trajectories is an ill-posed problem and many IRL approaches suffer from some or the other of the following limitations:

- Many cost functions including all zeros can make the given trajectories optimal. Additional assumptions about expert behavior are essential to make the problem well-defined.
- If demonstration data is sub-optimal, the learnt stochastic policy may assign zero probability to it.
- For certain policy distributions, the most likely policy may not be the one that maximizes the objective.
- May require solving a non-convex optimization problem which may be intractable.

Maximum causal entropy (MCE) IRL[3] approach is devoid of the above limitations and aims to maximize the causal likelihood of expert demonstrations assuming a certain stochastic policy distribution. Given that we only have a finite amount of expert demonstrations, it is an indirect way of handling unknown rewards/costs. The learnt policy is one that could "work well" with the widest variety of cost functions. Let π_E be the expert policy using which the expert trajectories are generated, then MCE IRL finds the cost function $c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ which satisfies:

$$\text{IRL}(\pi_E) = \underset{c\,\in\,\mathbb{R}^{\mathcal{S}\times\mathcal{A}}}{\arg\max} - \psi(c) + \left(\min_{\pi\in\Pi} -H(\pi) + \mathbb{E}_\pi[c(s,a)]\right) - \mathbb{E}_{\pi_E}[c(s,a)] \tag{2}$$

Here, $\psi(c)$ is the regularization term and H is the γ-discounted causal entropy of policy π defined as,

$$H(\pi) \triangleq \mathbb{E}_\pi[-\log\, \pi(a|s)] \tag{3}$$

In practice, π_E is not known and the corresponding expectation term is estimated from the sampled expert trajectories. IRL finds cost functions that assign a low cost to expert policy and high costs to all other policies thereby clearly segregating the expert behavior from others. Let $\tilde{c} \in \mathrm{IRL}_\psi(\pi_E)$. Then, we recover the expert policy by computing $\mathrm{RL}(\tilde{c})$ which gives us high entropy near-expert policies which minimize the cost function $\tilde{c}$,

$$\mathrm{RL}(\tilde{c}) = \underset{\pi\in\Pi}{\arg\min} - H(\pi) + \mathbb{E}_\pi[\tilde{c}(s,a)] \tag{4}$$

2.2 Removal of IRL step in imitation learning

In the imitation learning scenario with given expert trajectories, the IRL procedure (estimation of cost function) acts as an intermediate step and we wish to remove it. Combining equations (2) and (4), the paper proves,

$$\mathrm{RL} \circ \mathrm{IRL}(\pi_E) = \underset{\pi\in\Pi}{\arg\min} - H(\pi) + \psi^*(\rho_\pi - \rho_{\pi_E}) \tag{5}$$

where, ψ^* is the convex conjugate of cost-regularizer ψ and ρ_π is the occupancy measure for policy π. Define $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. Then, for $\psi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \overline{\mathbb{R}}$, convex conjugate $\psi^* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \overline{\mathbb{R}}$ is given by,

$$\psi^*(x) = \sup_{y\,\in\,\mathbb{R}^{\mathcal{S}\times\mathcal{A}}} x^T y - \psi(y) \tag{6}$$

The occupancy measure $\rho_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ gives the infinite horizon state-action distribution when following policy π and is defined as,

$$\rho_\pi(s,a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t\, P(s_t = s|\pi) \tag{7}$$

So,

$$\mathbb{E}_\pi[c(s,a)] = \sum_{s,a} \rho_\pi(s,a)\, c(s,a) \tag{8}$$

From equation (5), we see that our aim is to find a policy $\pi \in \Pi$ whose occupancy measure is close to expert using ψ^* as the distance measure. Moreover, different cost regularizers ψ give rise to different imitation learning algorithms.

2.3 Using constant function as cost regularizer

If ψ is a constant function and $\tilde{\pi} \in \mathrm{RL} \circ \mathrm{IRL}_\psi(\pi_E)$ then it can be shown that $\rho_{\tilde{\pi}} = \rho_{\pi_E}$. This means the algorithm tries to identify a policy that exactly matches the expert occupancy measure. However, in practical situations, we only know a finite number of expert trajectories, so expert occupancy measure is zero for most state-actions. Exact match would force our learned policy to never visit these state-action pairs. Also, the exact match has to be ensured for each state-action pair in this formulation making it intractable for large environments.

2.4 Connection with apprenticeship learning

In the general form, equation (5) can be written as,

$$\mathrm{RL} \circ \mathrm{IRL}(\pi_E) = \underset{\pi \in \Pi}{\arg\min} - H(\pi) + d_\psi(\rho_\pi, \rho_{\pi_E}) \tag{9}$$

We get equation (5) by setting $d_\psi(\rho_\pi, \rho_{\pi_E}) \triangleq \psi^*(\rho_\pi - \rho_{\pi_E})$. Apprenticeship algorithms work with parameterized policies in large state spaces and find a policy better than expert by optimizing the objective,

$$\underset{\pi}{\text{minimize}} \ \max_{c \in \mathcal{C}} \mathbb{E}_\pi[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \tag{10}$$

where $\mathcal{C} \subset \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is the set of cost functions formed by linear combinations [4, 5] of basis functions $f_1, \cdots, f_d$. Using indicator function $\delta_\mathcal{C} : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \bar{\mathbb{R}}$ defined as,

$$\delta_\mathcal{C} = \begin{cases} 0 & c \in \mathcal{C} \\ +\infty & \text{otherwise} \end{cases} \tag{11}$$

we have,

$$\max_{c \in \mathcal{C}} \mathbb{E}_\pi[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \tag{12}$$

$$= \max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} -\delta_\mathcal{C}(c) + \mathbb{E}_\pi[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \tag{13}$$

$$= \max_{c \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}} -\delta_\mathcal{C}(c) + \sum_{s,a} (\rho_\pi(s,a) - \rho_{\pi_E}(s,a)) \ c(s,a) \tag{14}$$

$$= \delta^*_\mathcal{C}(\rho_\pi - \rho_{\pi_E}) \tag{15}$$

So, from equation (10), now we can write the entropy-regularized objective as,

$$\underset{\pi}{\text{minimize}} \ - H(\pi) + \delta^*_\mathcal{C}(\rho_\pi - \rho_{\pi_E}) \tag{16}$$

From equation (16), apprenticeship learning is a special case of our imitation learning objective (9). The advantage of apprenticeship learning is that it is tractable in large environments and works with function approximations. However, a limitation is that the cost functions class $\mathcal{C}$ is very restrictive. It is just a linear combination of basis functions and so may not always be able to well represent the true expert cost function. In such situations, the algorithm may not be able to imitate the expert well.

2.5 Policy gradient for apprenticeship learning

As shown by [6], apprenticeship learning objective can be optimized using policy gradient approach. For a parameterized policy π_θ, let,

$$c^* = \arg\max_{c \in \mathcal{C}} \mathbb{E}_{\pi_\theta}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] \tag{17}$$

Then gradient of apprenticeship learning objective can be written as,

$$\begin{aligned} &\nabla_\theta \max_{c \in \mathcal{C}} \mathbb{E}_{\pi_\theta}[c(s,a)] - \mathbb{E}_{\pi_E}[c(s,a)] && (18)\\ &= \nabla_\theta \, \mathbb{E}_{\pi_\theta}[c^*(s,a)] && (19)\\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \, \log \pi_\theta(a|s) \, Q_{c^*}(s,a)] && (20)\end{aligned}$$

Here,

$$Q_{c^*}(s,a) = \mathbb{E}_{\pi_\theta}[c^*(s,a) \mid s_0 = s, a_0 = a] \tag{21}$$

This is similar to the actor-critic setup with additional c^* handing and can be solved using the iterative algorithm proposed by [6],

1. Sample trajectories using current policy estimate π_{θ_i} and fit cost function c_i^* as per equation (17).
2. Estimate gradient using above formulation with existing sampled trajectories and form new policy estimate $\pi_{\theta_{i+1}}$ by taking TRPO[7] or PPO[8] step.

Due to the similarity of apprenticeship learning objective and our imitation learning setup in equation (9), we will be applying a very similar iterative algorithm for imitation learning.

2.6 Connection with generative adversarial networks

In equation (9), next we use the cost regularizer defined as,

$$\psi_{GA}(c) \triangleq \begin{cases} \mathbb{E}_{\pi_E}[g(c(s,a))] & c < 0 \\ +\infty & \text{otherwise} \end{cases} \tag{22}$$

where function $g : \mathbb{R} \to \bar{\mathbb{R}}$ is defined as,

$$g(x) \triangleq \begin{cases} -x - \log(1 - e^x) & x < 0 \\ +\infty & \text{otherwise} \end{cases} \tag{23}$$

Using this ψ_{GA}, [1] shows that,

$$\psi^*_{GA}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_\pi[\log D(s,a)] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))] \tag{24}$$

This represents the negative log loss of distinguishing state-actions pairs generated by π and π_E and can also be represented by Jensen-Shannon divergence,

$$D_{JS}(\rho_\pi, \rho_{\pi_E}) \triangleq D_{KL}(\rho_\pi \,\|\, (\rho_\pi + \rho_{\pi_E})/2) + D_{KL}(\rho_{\pi_E} \,\|\, (\rho_\pi + \rho_{\pi_E})/2) \tag{25}$$

Next, treating the entropy term as a policy regularizer controlled by $\lambda \geq 0$, we get the overall imitation learning formulation,

$$\mathrm{RL} \circ \mathrm{IRL}(\pi_E) = \underset{\pi \in \Pi}{\arg\min} - \lambda H(\pi) + D_{JS}(\rho_\pi, \rho_{\pi_E}) \tag{26}$$

This will return a policy whose occupancy measure minimizes the Jensen-Shannon divergence with the expert's. This is our final formulation.

The loss objective we get in equation (24) draws the connection of imitation learning with generative adversarial networks where the learner's occupancy measure ρ_π is analogous to distribution generated by a generator model G. The task of the discriminator model D is to distinguish ρ_π from expert's occupancy measure ρ_{π_E} (true distribution). When discriminator can no longer distinguish we can say that our imitation learning has been successful and we have generated a policy π that captures the expert behavior.

2.7 Algorithm

Parameterize the policy π to be learnt as π_θ with policy parameters θ. Use function approximation for discriminator function D in equation (24) as $D_w : \mathcal{S} \times \mathcal{A} \rightarrow (0, 1)$ with weights w.

```
Algorithm 1: GAIL
  Input: Expert trajectories $\tau_E \sim \pi_E$, initial parameters $\theta_0$, $w_0$
  for $i = 0, 1 \cdots$ do
      Sample trajectories $\tau_i \sim \pi_{\theta_i}$
      Update discriminator weights from $w_i$ to $w_{i+1}$ through gradient,

          $\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s,a))]$    (27)

      Update policy parameters $\theta$ by taking TRPO/PPO step on cost
      function $\log(D_{w_{i+1}}(s,a))$ using equation (20)
  end
```

2.8 Trust region policy optimization

Proposed by [7], Trust region policy optimization (TRPO) develops on top of the vanilla actor-critic policy gradient setup. Firstly, to reduce variance instead of using Q-values directly, we work with the advantage function. For robustness and efficiency, trajectories are not sampled at each step of policy gradient. Instead, importance sampling is used. However, to make sure that

$\pi_{\theta_{i+1}}$ does not deviate too far from π_{θ_i}, the algorithm bounds the KL-divergence between the two policies. Let the bound be δ. The advantage function is $A_\pi(s,a) \triangleq Q_\pi(s,a) - V_\pi(s)$. For any two iterations with policy parameters, θ_{old} and θ, the algorithm works with the general objective,

$$\underset{\theta}{\text{minimize}}\ \mathbb{E}_{\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\, A_{\pi_{\theta_{old}}}(s,a)\right] \tag{28}$$

$$\text{subject to } \mathbb{E}_{\pi_{\theta_{old}}}[D_{KL}(\pi_{\theta_{old}}(.|s) \,\|\, \pi_\theta(.|s))] < \delta \tag{29}$$

Here, we minimize because advantage is defined in terms of cost function c instead of rewards. So, overall goal is to minimize costs. This can also be written as a KL-penalized objective, in terms of another hyperparameter β as,

$$\underset{\theta}{\text{minimize}}\ \mathbb{E}_{\pi_{\theta_{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\, A_{\pi_{\theta_{old}}}(s,a)\right] - \beta\, \mathbb{E}_{\pi_{\theta_{old}}}[D_{KL}(\pi_{\theta_{old}}(.|s) \,\|\, \pi_\theta(.|s))] \tag{30}$$

It turns out that optimizing this objective involves the calculation of a second-order derivative and its inverse, which is a very expensive operation. Practical TRPO algorithm does try to handle this by making some approximations however still the method is less sample efficient.

2.9 Proximal Policy Optimization

Introduced by [8], this approach builds on top of the TRPO objective but adds soft constraints as approximation to allow first-order gradient descent methods to work well. This makes the PPO algorithm sample efficient without compromising on performance.

Adaptive KL Penalty. This involves dynamically adjusting β parameter in (30). If the KL-divergence term is smaller than a fixed target value, we expand the trust region by increasing β. If it exceeds another fixed target value, we shrink the trust region by reducing β.

Clipped Objective. The importance sampling ratio in the TRPO objective gives a measure of difference between the old and new policy. To discourage large and abrupt policy changes, this approach clips the importance sampling ratio used in the expectation term. We define the ratio as,

$$r_\theta(s,a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \tag{31}$$

The overall objective then becomes,

$$\underset{\theta}{\text{minimize}}\ \mathbb{E}_{\pi_{\theta_{old}}}\left[\max\left(r_\theta\, A_{\pi_{\theta_{old}}}, \text{clip}(r_\theta, 1+\epsilon, 1-\epsilon) A_{\pi_{\theta_{old}}}\right)\right] \tag{32}$$

The notation here is opposite from the original paper. Note that we are working with costs here (which are negative values). We want to minimize the costs. The advantage function and Q-values in our formulation are defined in terms of costs. This is opposite to rewards (which are positive values) and we want to maximize the rewards. So, in our clipped objective, we have a max operator to take the less negative term (the one which has the smaller absolute magnitude and hence the least abrupt change).

3 Empirical Study

Taking inspiration from existing open-source implementation of GAIL in Tensorflow[1] and PPO in PyTorch[9], we open-source a PyTorch-based implementation of the GAIL algorithm. The **source code** can be found on GitHub[2].

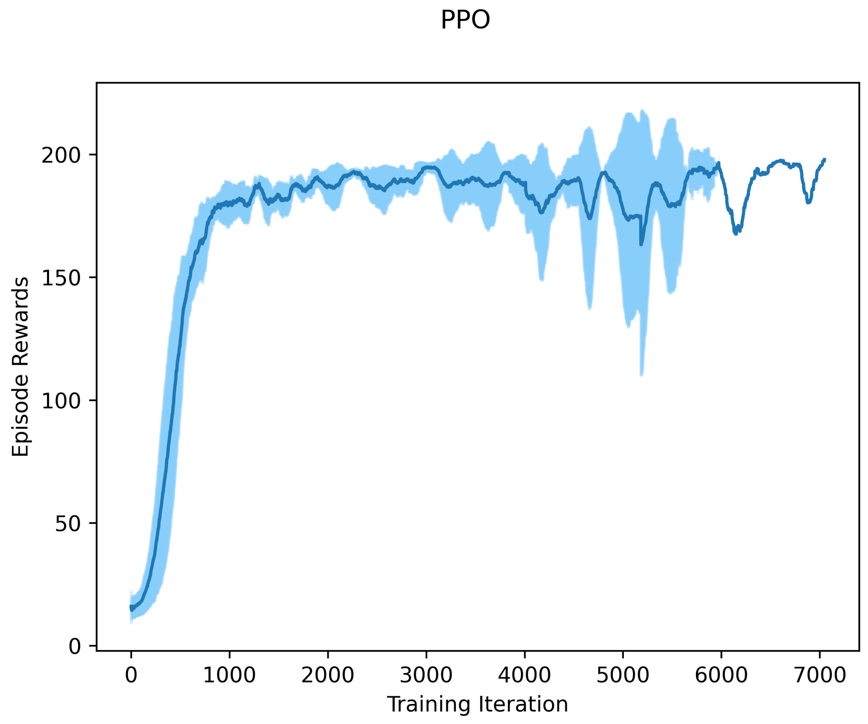

Figure 1: `CartPole-v0` PPO Results

[1]https://github.com/nav74neet/gail_gym
[2]https://github.com/jatinarora2702/gail-pytorch

3.1 CartPole environment

We work with the `CartPole-v0` environment of OpenAI Gym library. We first run PPO algorithm to get an expert policy on the environment. We sample 25 trajectories using this expert policy. Using these policies we next train our imitation learning (GAIL) framework using the PPO algorithm internally.

The `CartPole-v0` environment is considered to be solved when we get an average reward of 195.0 over 100 consecutive trials. We set this as our termination condition for both PPO and GAIL setups. During the training phase, we keep recording the average reward over past 100 episodes. With random initialization of model and environment parameters, we run GAIL and PPO algorithms 5 times each and plot the mean and error at 95% confidence interval. For the GAIL algorithm, the expert sampled trajectories are kept the same across all 5 runs. Figures 1 and 2 give the plots for PPO and GAIL algorithms respectively. The bold line represents the mean value and the light region around the line represents the error.

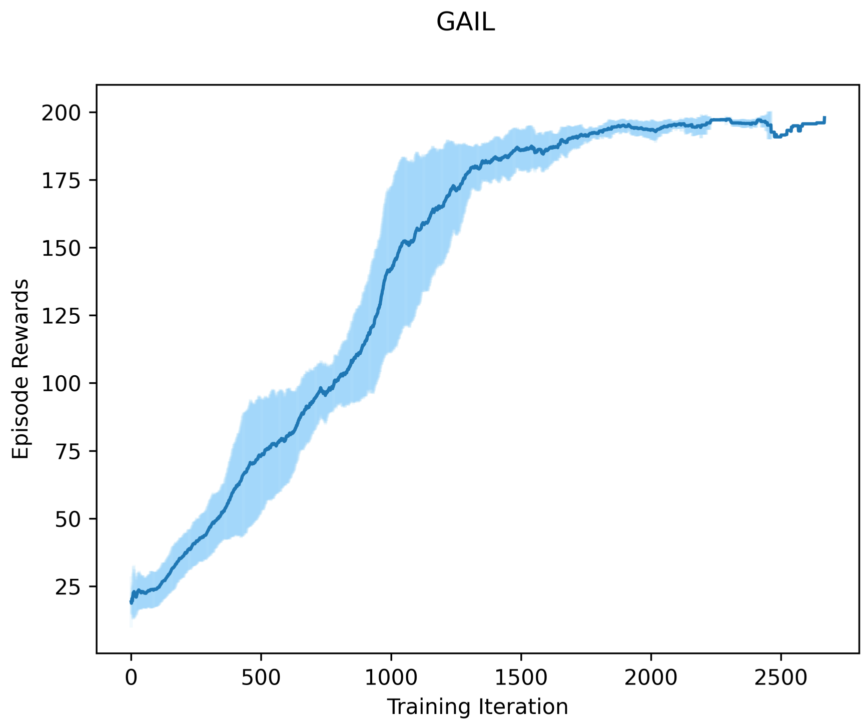

Figure 2: `CartPole-v0` GAIL Results

3.2 Observations and Inferences

Both the algorithms reach close to 200 average rewards in the first 1000 episodes (training steps). However, the PPO algorithm is found to be sensitive to learning rates and hyperparameters. The error bars show that for our model settings, even in the later stages of the training, the algorithm sometimes reaches local optima and starts giving low rewards giving rise to high error regions.

The GAIL algorithm is found to converge well with the expert behavior and the error region shrinks during the later stages of training signifying the stability of the algorithm.

4 Conclusion

In this work, we studied an important model-free imitation learning algorithm GAIL[1]. We presented a theoretical as well as empirical study on the algorithm. In the theoretical study, we presented the imitation learning problem as IRL followed by RL and then removed the redundant IRL step. We looked at various cost-regularizer settings and their connections with other related works. We then designed a cost-regularizer that makes the overall objective similar to generative adversarial networks thus giving us a generative adversarial imitation learning algorithm. We then briefly discussed policy gradient methods like trust region policy optimization (TRPO) and proximal policy optimization (PPO). Next, we implemented the algorithm in PyTorch and ran it on a simple `CartPole-v0` environment to prove its empirical effectiveness.

Acknowledgement

This work has been done as a course project by the author for CS498: Reinforcement Learning course taught at University of Illinois Urbana-Champaign. The author would like to thank our instructor Professor Nan Jiang along with the course teaching assistants for their guidance and support throughout the course.

References

[1] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

[2] Michael Bloem and Nicholas Bambos. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *53rd IEEE Conference on Decision and Control*, pages 4911–4916. IEEE, 2014.

[3] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[4] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[5] Umar Syed, Michael Bowling, and Robert E Schapire. Apprenticeship learning using linear programming. In *Proceedings of the 25th international conference on Machine learning*, pages 1032–1039, 2008.

[6] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pages 2760–2769. PMLR, 2016.

[7] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[9] Nikhil Barhate. Minimal pytorch implementation of proximal policy optimization. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.