Adversarial Policy Training against Deep Reinforcement Learning

Chen Gong

12 May 2021

1 Summary

1.1 Key Idea

In this paper, authors extends a reinforcement learning algorithm (Proximal Policy Optimization, PPO) to guide the training of the adversarial agent in the two-agent competitive game setting. The main contributions are concluded as follows,

1. The attack mechanism is the first work that can effectively exploit the weakness of victim agents without manipulation of the environment, explicit knowledge of the opponent policy network and state-transition model. So this attack mechanism, is practical, and trains an adversarial agent in an effective and efficient fashion.

2. To facilitate the search of the adversarial policy network, authors adjust the weight of the action deviation based on by how much the victim agent pays attention to the action of the action of the adversarial. The weight is estimated by an explainable AI technique.

3. Authors conduct experiments using MuJoCo and roboschool Pong to evaluation their attack. Compared with the SOTA technique, the adversarial agent exhibits a much stronger capability in exploiting the weakness of victim agents. Besides, adversarial attack shows less variation in the training process.

1.2 Algorithm Design

In this section, I introduce the details in algorithm. This method extend the PPO loss function L_{PPO} a new loss term

$$L_{\text{ad}} = \text{maximize}_{\theta} \left(\left\| \hat{a}_v^{(t+1)} - a_v^{(t+1)} \right\|_1 - \left\| \hat{o}_v^{(t+1)} - o_v^{(t+1)} \right\|_1 \right), \tag{1}$$

where θ is parameters in adversarial agent π_α. $\hat{o}_v^{(t+1)}$ and $\hat{a}_v^{(t+1)}$ are the different observation and action taken by the opponent agent if, at the time step t, the adversarial agent takes an action different from the ones indicated by the trajectory rollouts. The key idea of this term can be concluded as, when maximizing the deviation of an opponent action, we need to ensure the minimal change of the environment observation.

Neither the opponent policy network π_v nor state-transition model $\mathcal{P}_v^{ss'}$, we cannot get the observation of the opponent agent $\hat{o}_v^{(t+1)}$ and the action of the opponent agent $\hat{a}_v^{(t+1)} = \pi_v(\hat{o}_v^{(t+1)})$. Therefore, author build two individual deep nature networks to approximate the opponent policy network and its state-transition model.

The state-transition model is defined as $H(o_v^{(t)}, a_v^{(t)}, a_\alpha^{(t)}; \theta_h)$, where $(o_v^{(t)}, a_v^{(t)}, a_\alpha^{(t)})$ is input and $o_v^{(t+1)}$ is output. So the $\hat{a}_v^{(t+1)}$ is the output of $H(o_v^{(t)}, a_v^{(t)}, \hat{a}_\alpha^{(t)}; \theta_h)$.

The policy opponent policy model is defined as $F(o_v^{(t)}; \theta_f)$, where $(o_v^{(t)})$ is input and $a_v^{(t+1)}$ is output. So the $\hat{a}_v^{(t+1)}$ is the output of $F(H(o_v^{(t)}, a_v^{(t)}, \hat{a}_\alpha^{(t)}; \theta_h); \theta_f)$.

The L_{ad} can be rewritten as

$$L_{ad} = \text{maximize}_\theta \left(\left\| F\left(H\left(o_v^{(t)}, a_v^{(t)}, \hat{a}_\alpha^{(t)}\right)\right) - a_v^{(t+1)} \right\|_1 - \left\| H\left(o_v^{(t)}, a_v^{(t)}, \hat{a}_\alpha^{(t)}\right) - o_v^{(t+1)} \right\|_1 \right) \tag{2}$$

Obviously, the total loss function is defined as $L_{\text{PPO}} + \lambda L_{\text{ad}}$. The hyperparameter λ indicate the importance of newly added term L_{ad}, which is calculated based on the weight that the opponent agent pays attention to the adversarial. The λ in step time t is defined as

$$I^{(t)} = \left\| F\left(o_v^{(t)}\right) - F\left(o_v^{(t)} \odot \left(\tilde{g}^{(t)} \odot M\right)\right) \right\|_\infty, \quad \lambda^{(t)} = \frac{1}{1 + I^{(t)}}, \tag{3}$$

where $\tilde{g}^{(t)} = \sum_{j=1:q} g_{ij}^{(t)}$, and $g^{(t)_{ij}} = \nabla_{(o_i^{(t)}} F(o_v^{(t)})_j$. In $o_v^{(t)}$, if the corresponding observation dimensions indicate the actions of adversarial agent, we assign 1 to the corresponding element in M, and the rest is assign to 0.

2 My Reflections

1. Analysis the Reinforcement Learning Algorithms by Software Engineering Methods.
2. Meta Reinforcement Learning
3. Multi-Agent Reinforcement Learning.