MULTI-AGENT GENERATIVE ADVERSARIAL SELF IMITATION LEARNING

Anurag Dutt
Department of Applied Mathematics and Statistics
Stony Brook University
Stony Brook, NY 11794
anurag.dutt@stonybrook.edu

Pratyush Ranjan
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794
pranjan@cs.stonybrook.edu

May 18, 2020

ABSTRACT

This project implements and experiments a Generative Adversarial Self-Imitation Learning(GASIL) approach for the multiagent predator-prey environment. The GASIL approach stores and imitates the "good" or high reward trajectories seen in the past, and imitates them in the GA framework. GASIL could be complemented with the Q-Learning or Policy gradient in the control part of the reinforcement learning. We implemented the GASIL in multi-agent environment and our results show that the GASIL outperforms the basline DDPG model.

Contents

List of Tables

List of Figures

Keywords Reinforcement Learning · Generative Adversarial · Imitation Learning

1 Introduction

Reinforcement Learning is a mechanism of optimizing the expected reward for a given state. Coordination mechanism between the independent agents being trained on their separate GAN networks. Joint action learners and Independent learners. Joint Action Learning Algorithm - every agent's reward is considered independently.??? Multi-agent DDPG consists of a centralized critic updating policy of all agents involved. Our experiments runs on a decentralized or the independent learners for each agent. Self Imitation Learning - local observation (current action and rewards) comparison and the good trajectories from the past.

The positive buffer is designed subcurriculum experience replay that helps pickout the prefereable experiences that the agent has experienced, and hence we combine the two. We also show that GASIL is robust to stochastic dynamics to some extent in practice.

2 Relevant Concepts

2.1 Self Imitation Learning

In an environment with the very sparse reward, it's difficult to learn the whole task at once. It is natural to master some basic skills for solving easier tasks firstly. e.g., In Montezuma's Revenge (an Atari game), the agent needs to pick up the key and then open the door. Directly learning opening the door is hard due to the poor exploration, but it is easier to master picking up the key at first. Based on this idea, self-imitation learning (SIL) is a very recent approach proposed to solve the sparse reward problem by learning to imitate the agent's own past good experiences. In brief, SIL stores previous experiences in a replay buffer and learns to imitate the experiences when the return is greater than the agent's expectation. Experimental results show that this bootstrapping approach (learn to imitate the agent's own past good decisions) is highly promising on hard exploration tasks A proper level of exploitation of past good experiences during learning can lead to a deeper exploration (moving to the deeper region) of the learning environment.

2.2 Generative Adversarial Imitation learning

Imitation learning is also known as learning from demonstrations or apprenticeship learning, whose goal is to learn how to perform a task directly from expert demonstrations, without any access to the reward signal r(s, a). Recent main lines of researches within imitation learning are behavioural cloning (BC). which performs supervised learning from observations to actions when given a number of expert demonstrations; inverse reinforcement learning (IRL). , where a reward function is estimated that explains the demonstrations as (near) optimal behavior; and generative adversarial imitation learning (GAIL) , which is inspired by the generative adversarial networks (GAN). Let T_E denote the trajectories generated by the behind expert policy π_E , each of which consists of a sequence of state-action pairs. . In the GAIL framework, an agent mimics the behavior of the expert policy π_E by matching the generated state-action distribution ρ_{π_θ}(s, a) with the expert's distribution ρ_{π_θ}(s, a). The state-action visitation distribution (occupancy measure [17]) of a policy π_θ is defined as:

$$\rho_{\pi_\theta}(s,a) = \pi_\theta \sum_{t=0}^{\infty} \gamma^t p(s_t = s|\pi_\theta) \tag{1}$$

where $p(s_t = s|\pi_\theta)$ is the probability of being in state s at time t when starting at state s_0 ρ_0 and following policy π_θ The optimum is achieved when the distance between these two distributions is minimized as measured by Jensen-Shannon divergence. The formal GAIL objective is denoted as:

Unlike GANs, the original GAIL requires interactions with the environment/simulator to generate state-action pairs, and thus the objective is not differentiable end-to-end with respect to the policy parameter θ. Hence, optimization of the policy requires RL techniques based on Monte-Carlo estimation of policy. The optimization over the GAIL objective is performed by alternating between K gradient step to increase with respect to the discriminator parameters phi, and a Trust Region Policy Optimization (TRPO) step to decrease (with respect to the policy parameters θ (using log(D_ϕ (s, a)) as the reward function).

2.3 Generative Adversarial Self-Imitation learning

The idea behind GASIL is to treat good trajectories collected by the agent as trajectories that the agent should imitate as described in Algorithm 1. More specifically, GASIL performs the following two updates for each iteration.

Algorithm 1 Generative Adversarial Self-Imitation Learning

Initialize policy parameter θ
Initialize discriminator parameter ϕ
Initialize good trajectory buffer $\mathcal{B} \leftarrow \emptyset$
for each iteration **do**
Sample policy trajectories $\tau_\pi \sim \pi_\theta$
Update good trajectory buffer $\mathcal{B}$ using τ_π
Sample good trajectories $\tau_E \sim \mathcal{B}$
Update the discriminator parameter ϕ via gradient ascent with:

$$\nabla_\phi \mathcal{L}_{\text{GASIL}} = \mathbb{E}_{\tau_\pi} \left[\nabla_\phi \log D_\phi(s,a)\right] + \mathbb{E}_{\tau_E} \left[\nabla_\phi \log(1 - D_\phi(s,a))\right]$$

Update the policy parameter θ via gradient descent with:

$$\nabla_\theta \mathcal{L}_{\text{GASIL}} = \mathbb{E}_{\tau_\pi} \left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a)\right] - \lambda \nabla_\theta \mathcal{H}(\pi_\theta),$$
$$\text{where } Q(s,a) = \mathbb{E}_{\tau_\pi} \left[\log D_\phi(s,a) | s_0 = s, a_0 = a\right]$$

end for

- **Updating good trajectory error** (β)

  GASIL maintains a good trajectory buffer $\beta = \tau_i$ that contains a few trajectories (i) that obtained high rewards in the past. Each trajectory consists of a sequence of states and actions: s_0, a_0, s_1, a_1, ..., s_T . We define 'good trajectories' as any trajectories whose the discounted sum of rewards are higher than that of the policy. Though there can be many different ways to obtain such trajectories, we propose to store top-K episodes according to the return $R = \sum_{t=0}^{m} \gamma^t r_t$

- **Updating discriminator** (D_ϕ) **and policy** (π_θ)

  The agent learns to imitate good trajectories contained in the good trajectory buffer β using generative adversarial imitation learning.

3 Motivation

- Why GASIL over GAIL?

- One limitation of GAIL is that it requires a significant number of interactions with the learning environment in order to imitate an expert policy. To address the sample inefficiency of GASIL, we use off-policy DDPG algorithm and perform off-policy training of the GAIL discriminator performed in such way: for each agent i, instead of sampling trajectories from the current policy directly, we sample transitions from the replay buffer R collected while performing off-policy training

- Another problem of GAIL is that either r_i (s, a) = log(1D(s, a)) or r_i (s, a) = log(D(s, a)) (which is often used as the reward function in GAIL approaches) has reward biases that can either implicitly impose prior knowledge about the true reward, or alternatively, prevent the policy from imitating the optimal expert. We use a more stable reward function:

$$r_i(s,a) = log(D(s,a)) - log(1 - D(s,a)) \tag{2}$$

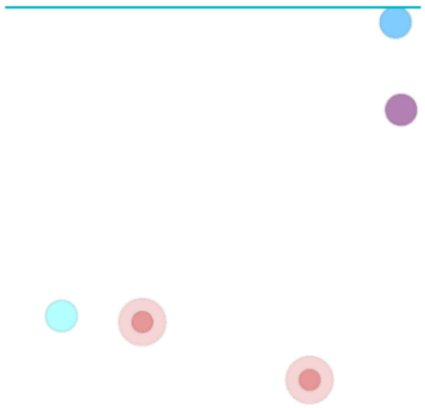

Figure 1: A snapshot of the predator prey environment. The red dots are the predator. Purple, dark blue and light blue are respectively the high, medium and low reward preys.

Table 1: GASIL Hyperparameters for Predator-Prey

Hyperparameters	Value
Architecture	FC(64)-FC(64)
Policy Learning rate	0.01
DDPG Learning rate	0.001
Discriminator Learning rate	0.001
Number of episodes	7M
Minibatch size	1024
Discount factor(γ)	0.99
Steps in each episode	[60, 120]
Max Episode Length	120
Discriminator minibatch size	1024
Number of discriminator updates per batch	10
Size of good trajectory buffer (steps)	[128, 512]
Use of prioritization (α)	0.6

4 Experiment and Approach

4.1 Predator-Prey environment

The environment is the classic predator-prey game of Wolfpack where two wolves collaborate with each other to catch a prey. We have a square map filled with many copies of predator and prey agents. Agents have a limited energy supply and die after running out of energy (the episode ends after a fixed number of steps), falling off the map, or being eaten by another agent. The world offers a state to each agent which include a few internal variables, such as relative energy compared to maximum (0 implies imminent death), as well as vision in the form of an array of gray-scale and distance. Agents may choose to move forward (and optionally left or right), stay still, attack. Predators are modestly rewarded for attacking the wrong prey, strongly rewarded for attacking the high-priority prey and harshly punished for dying in any way or being unable to catch any prey. There are N slower cooperating rescue agents which cooperatively chase and rescue one of the M faster wounded animals in a randomly generated. The target for each rescue agent is learning to rescue the same wounded animal **independently** without knowing each other's policy environment.

All rescue agents and wounded animals can observe the relative positions of others. Besides, each rescue agent can observe the relative velocities of wounded animals (can't observe the other rescuers' velocities). Actions are accelerations in 4 directions (controlled by 2 actions actually: north or south, east or west). The acceleration of a direction is controlled by the force applied. To sum up, the action space is continuous with a valid range of [-1, 1].

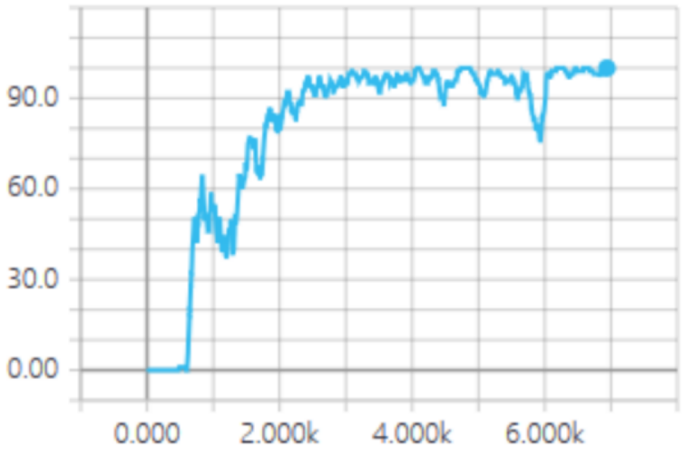

Figure 2: Reward vs. Episodes

All rescue agents and wounded animals can observe the relative positions of others. Besides, each rescue agent can observe the relative velocities of wounded animals (can't observe the other rescuers' velocities). Actions are accelerations in 4 directions (controlled by 2 actions actually: north or south, east or west).

4.2 Predator-Prey Architecture

In this paper, all of our policies, critics and discriminators are parameterized by a two-layer ReLU MLP (Multilayer Perceptron) followed by a fully connected layer activated by tanh functions for DDPG's policy nets. and sigmoid functions for all discriminators. For the hyperparameters used and their optimization, please see table.1

5 Results and Conclusion

The reward curve is the average of previous 100 episodes and is interpreted as a percentage. If the predators are able to capture the purple prey 50 time out of the past 100 episodes, then the reward (win rate) is 50%. As we can see from the Figure 2:

1. Upto 500K episodes, the predators are not able to catch the prey. The reward average is less than <5% for the stage. This is the inital training stage when the GASIL is trying to learn the optimal policy.
2. After about 600K episodes, the predator performance significantly improves. The predators are able to catch prey 60% of the time
3. After about 1.5 million episodes, the predator is mostly able to catch prey 90% of time. The model has learnt to select optimal policy/action in a given state.
4. After 2 million episodes ,the predators show a consistent high hunt rate > 90%.

The GASIL model in the multiagent environment with independent learned predator agents outperforms the DDPG prey model and is able to manage a win rate of over 90%.

6 Future Work and Development

The current model could be further improved by applying following techniques. These techniques are part of a future work and development of GASIL.

1. **Model Based GASIL** - an added improvement where the environment model is learnt while training the GAN. The model is then used to simulate the new policies.
2. **Multi-Modal GASIL** - A GASIL implementation with multiple policy distributions. The distributions are then learned separately as a "skill" and applied for a higher rewards.

References

[1] Yijie Guo , Junhyuk Oh, Satinder Singh, and Honglak Lee. Generative Adversarial Self-Imitation Learning. In *ICLR, 2014.*

[2] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.

[3] Jonathan Ho and Stefano Ermon Generative Adversarial Imitation Learning

[4] Jiaming Song, Hongyu Ren, Dorsa Sadigh ans Stefano Ermon Multi-Agent Generative Adversarial Imitation Learning

[5] Self-Imitation-Learning with A2C - https://github.com/TianhongDai/self-imitation-learning-pytorch

[6] Multi-Agent Generative Adversarial Imitation Learning - https://github.com/ermongroup/multiagent-gail