Learning to Play Modified version of Breakout Atari Game Using Actor-Critic with Eligibility Trace

Amin Majdi

Fall 2020

1 Introducing

The aim of this work is to use reinforcement learning (RL) methods for developing an intelligent agent that is able to play a simple game similar to the Atari breakout game and maximize the reward.

Game Description: The game idea comes from the breakout game. In this game there is one button in the top of the game window, there is one moving ball and there is one paddle in the bottom of the window that should be used to hit the ball to the button in the top border of the game window to gain a reward. Each time ball bounce to any of the borders of the game window, it returns, except the bottom border. If ball bounce to the bottom border, game is over. Therefore, the episode will end in either conditions that the agent loses (ball bounce to the bottom border) or win (ball bounce to the button).

States and Actions: In this game actions are move left, move right and stay. States are different for simple and advanced version of the game. In simple version, states are defined based on location of ball, angle of ball movement, and location of the paddle. In advanced version of the game location and angle of the second ball should also be added to the states.

Game Implementation: For Implementation of this game I used pygame module [1].

2 Method

I used Actor-Critic with Eligibility Trace for learning agent to successfully play this game. In this approach both policy and value function are learned by the use of approximation methods. Value function is estimated as a linear function of weight vector and constructed features with Fourier Basis method [2] using equation (1).

$$\hat{V}(s, w) = w^\top x(s) = \sum_{i=1}^{d} w_i x_i(s) \tag{1}$$

For policy approximation exponential soft-max distribution is used, equation (2):

$$\pi(a|s,\theta) = \frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}} \tag{2}$$

and action preference in equation (2) is parametrizad using equation (3):

$$h(s,a,\theta) = \theta^\top x(s,a) \tag{3}$$

In both of the linear methods used for approximation of h and $\hat{V}$, multi-dimensional order-n Fourier cosine basis method [2] is used, which construct features with equation (4).

$$x_i(s) = cos(\pi s^\top c^i) \tag{4}$$

Reward Function: Each time ball hits the paddle, the agent receives a positive reward equal to 1000. If the paddle miss the ball and ball hits the bottom border, agent receives -1000 reward, and if agent hit the ball to the reward button, it will receive a reward equal to 10000.

3 Results

I started with a simplified version of the magic game, in which magic button is removed and there is only one ball in the game. In this version of the game ball always initiates from the same state (i.e., same location and same angle). Results for AC algorithm learning to play this version of the game with $\alpha^\theta = 0.0001$, $\alpha^w = 0.0001$, $\lambda^\theta = 0.65$ and $\lambda^w = 0.65$ for order seven (n=7) Fourier Basis with $\gamma = 0.99$ is shown in Figure 1. As you can see in this figure, the agent learns to hit the ball with the paddle and repeat that until the ball hits the reward button. As it is a multi-goal game, its learning has different steps. It first learns to hit the ball once with the paddle, then it learns to hit the ball with paddle more than one time until the ball hits the reward button and gain the highest reward which is 15000 in this case.

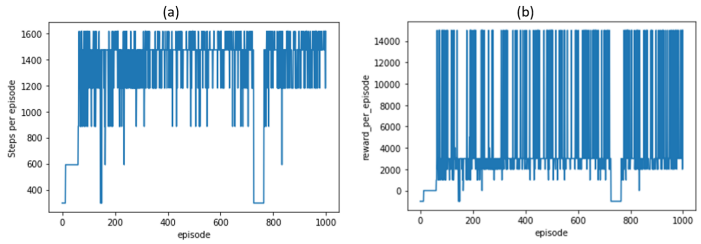

Figure 1: results for 1000 episodes of AC algorithm learning first simplified version of the game with one initial angle for ball, (a) step per episodes, (b) reward per episodes

I also made a second version of the game in which there are three possible initial states for the ball each of which have different angels. Figure 2, shows the results for AC algorithm

learning to play the second version of the game with similar parameters as those which were used for learning previous version of the game. As you can see in this figure the agent learns gradually to hit the ball multiple times.

In the third version of my game also there are three different initial states for the ball similar to the second version. However, in this version ball returns with three different angles when it hits the paddle. If the ball hits the paddle in the middle, it will return with same angle, but if the ball hits the paddle in each of the paddle ends of the paddle, return angle changes by $\pi/8$. My agent also learned to a good extent to follow the ball in this version of the game with $\alpha^{\theta} = 0.0001$, $\alpha^{w} = 0.0001$, $\lambda^{\theta} = 0.65$ and $\lambda^{w} = 0.65$ for order five (n=7) Fourier Basis with $\gamma = 0.99$.

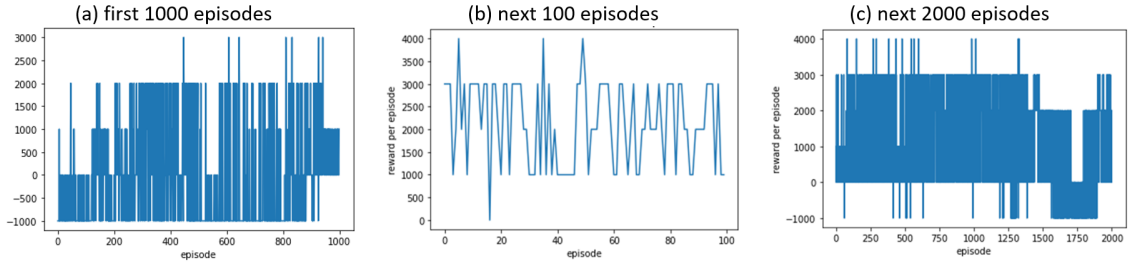

Figure 2: rewards gained by AC algorithm in the second simplified version of the game with three initial angles for the ball, (a) in first 1000 episodes, (b) in next 100 episodes,and (c) in next 2000 episodes

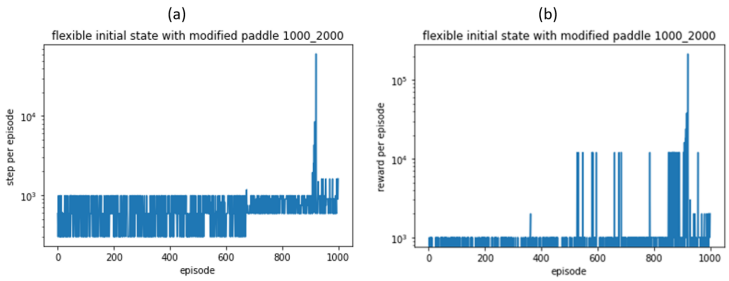

Figure 3: (a)time steps and (b)gained rewards

References

[1] P. Shinners, "This is how you cite a website in latex," https://www.pygame.org/docs/tut/PygameIntro.html, May 2000.

[2] G. Konidaris, S. Osentoski, and P. Thomas, "Value function approximation in reinforcement learning using the fourier basis," 2011.