Actor Critic - Final - Team 5

Ankit Shaw
M.S. Computer Science and Engineering
ashaw7@buffalo.edu

Rounak Biswas
M.S. Computer Science and Engineering
rounakbi@buffalo.edu

Abstract

This document is the report submission for Assignment 3 of CSE546 Reinforcement Learning course at University at Buffalo, The State University of New York. The document talks about Actor Critic algorithm implementation and solve three environment using the algorithm. We implement two Actor Critic based algorithms One-Step actor critic and Proximal Policy Optimization. PPO performed much better than the One-Step actor critic algorithm with advantage function.

1 Actor Critic Algorithm - Proximal Policy Optimization(PPO)

1.1 Actor-Critic

Actor-critic algorithms are a class of algorithms in Reinforcement Learning , which is a policy based RL algorithm. In Actor-Critic we maintain two sets of parameters.

1. Critic Network that Updates action-value function parameters
2. Actor Network that Updates policy parameters, in direction suggested by critic

The actor is the policy which conducts actions in an environment. While the critic computes value functions to help assist the actor in learning. These are usually the state value, state-action value, or advantage value.

1.2 PPO Algorithm

Proximal Policy Optimization is a Reinforcement Learning algorithm that comes under the Actor-Critic class. It is a policy based RL algorithm, that means in this algorithm we directly learn the optimal policy. The goal in Proximal Policy Optimization (PPO) is to reduce frquency of changes made to the policy network so that we increase the training stability of the policy. Smaller policy changes during training are more likely to converge to an ideal solution.

In PPO, we calculate the ratio between the current and previous policies in order to determine how much the present policy has changed in comparison to the earlier one. Additionally, we clip this ratio within the range [1 - epsilon, 1 + epsilon][1,1+]. Following shows the objective function of PPO.

$$J^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r(\theta)\hat{A}_{\theta_{\text{old}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{old}}}(s,a)\right)\right]$$

Figure 1: PPO: Objective Function

1.3 Difference between the actor-critic and value based approximation algorithms

Value based Approximation Algorithms -

1. The approach for the algorithm is to approximate the value function and determining the policy based on it.
2. Policy is extracted directly from the value function.
3. Policy is not directly learned and optimised.
4. Convergence is reached relatively slower compared to Actor Critic.
5. Not so effective in continuous space.
6. Approximating Value function is efficient with low variance compared to evaluating policy in Actor-Critic.

Actor Critic -

1. The approach for this algorithm is Policy evaluation and Policy improvement which are iteratively done in order to reach the convergence.
2. . The policy gradient methods target at modeling and optimizing the policy directly.
3. We directly learn and optimise the policy.
4. Convergence is reached relatively faster.
5. Effective in high dimensional and continuous space.
6. Effective in learning stochastic policy

2 Environment descriptions

In this section we describe the different environments we have attempted to solve using PPO.

2.1 OpenAI Cartpole

This environment is provided in the OpenAI gym library. It constitutes of a cart and a pole which is balancing on the cart. The goal is to keep the pole balanced on the cart for as long as possible.

2.1.1 States

A state in this environment is provided by a numpy array of shape `(4,)`. The state space is continuous in this environment. This constitutes the following properties in the given order.

1. Cart position:- Value in range `(-4.8 to 4.8)`.
2. Cart velocity:- Value in range `(-Inf to Inf)`.
3. Pole angle:- Value in range `(-0.418 rad to 0.418 rad)`.
4. Pole angular velocity:- Value in range `(-Inf to Inf)`.

2.1.2 Actions

The environment supports 2 actions: Left (0) and Right (1). We can see that it is a discrete action space.

2.1.3 Rewards

The agent is given a reward of `+1` for every step that is executed in the environment, even for the terminal state. An episode is terminated when either the pole becomes unbalanced or the reward exceeds `+475`. The rewards are not very spare in the environment, since at each step your continue to get `+1`.

2.1.4 Goal

The environment is assumed to be solve if the reward exceeds `+475`.

2.2 Lunar Lander

Lunar Lander is a classic environment in machine learning. In the environment our aim is to land the rocket on the ground. The agent in this environment needs to find optimal policy to land the rocket using the main and side engines to control the descent to prevent crashing.

2.2.1 States

The state is an array of size 8. Following are the observations:

1. X-coordinate of the lander.
2. Y-coordinate of the lander.
3. Linear velocity of lander in X-coordinate
4. Linear velocity of lander in Y-coordinate
5. Angle of the lander
6. Boolean to indicate if leg 1 of the lander is on ground
7. Boolean to indicate if leg 2 of the lander is on ground

2.2.2 Actions

There are 4 discrete deterministic actions:-

1. Do nothing
2. Fire left orientation engine.(1)
3. Fire main engine.(2)
4. Fire right orientation engine (3).

2.2.3 Rewards

The goal of the agent is to land the lander. If lander moves away from the landing pad it losses reward. The environment is said to be solved if agent lands the lander with more than 200 points as reward. Following are the rewards for different actions.

1. lander crash : -100
2. Each leg with contact on ground : +10
3. Fire main engine : -0.3/frame
4. Fire side engines: -0.03/frame

2.3 Mario Grid Environment

For our current experiment we used the Mario Grid World that we build for the Assignment 1. The figure 2 shows the grid world we have created for the initial testing.

2.4 Definition

2.4.1 States

In our Grid Environment defined the size of the environment is (7 x 5). There are 35 states in total.

$$\text{Set of States: } S = \{S0, S1, S2, \ldots S35\}$$

Each individual cell in the grid is an unique states. These states can have different reward associated to them as well.

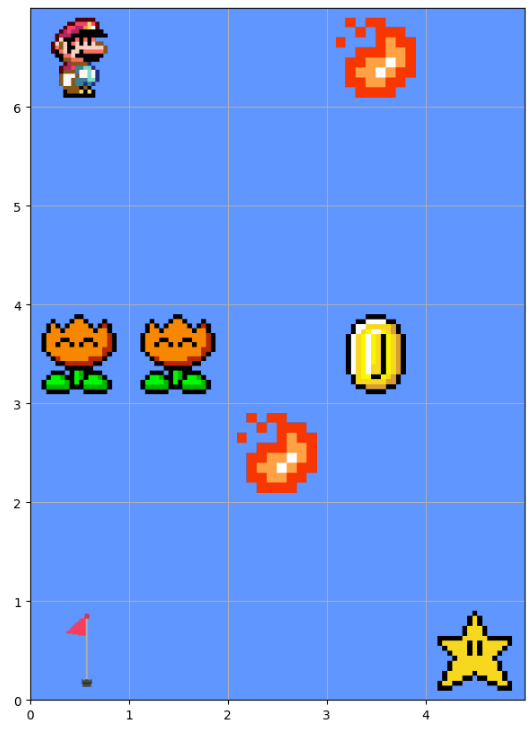


Figure 2: Grid World

2.4.2 Actions

An agent can take four actions in the environment. The actions could be Left, Right, Up, Down. These are denoted by 'L', 'R', 'U', 'D' respectively.

Set of Actions: S = {'L', 'R', 'U', 'D'}

2.4.3 Reward

We have five rewards in our grid environment. Two penalty rewards are -3,2 and there are three positive rewards 2, 4, 10. The maximum reward that an be achieved in the environment is 16.

Set of Rewards: R = {-3, -2, 2, 4, 10}

3 Part 1

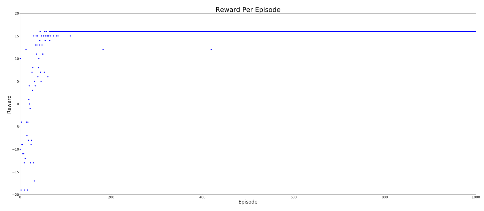
3.1 Mario Grid Environment

In this section we discuss the results of the PPO algorithm training on our Mario Grid Environment.

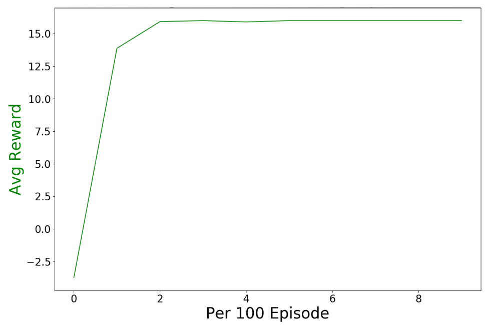
3.1.1 Training agent on the environment

Fig. 3 shows the reward per episode plot for training the agent on the MArio grid environment. The input to the network is a state of the environment as one-hot encode vector. This represents the position of the agent. Output of the actor network gives the probability of the four actions. While the critic network is gives the state value.

In the figure 3a we can observe that the agent was able to converge very soon and was able to get maximum reward of 16 points in the environment. For the first 100 episodes the agent didn't perform well which is obvious given the network parameters were not optimised yet. But after 100 episode



(a) Mario Grid: Reward per episode



(b) Mario Grid: Reward per 100 episode

Figure 3: Train results for Mario Grid World

the agent starts getting maximum reward. The agent was trained for 1000 episodes with 30 steps per episode.

The figure 3b shows the average reward per 100 episodes. This graph validates our observation that model started to converge in about 100 episodes.

3.1.2 Testing the trained agent on environment

Fig. 4 shows the reward per episode plot when agent only takes the learned greedy actions.

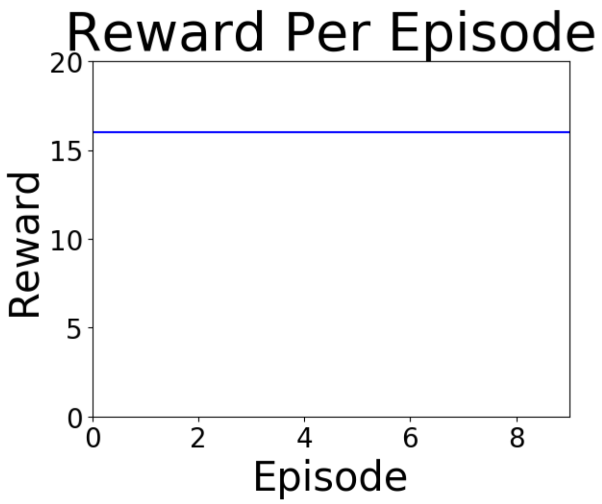


Figure 4: Mario Grid: Reward per episode for learned policy.

Once we have trained our agent we evaluate it for 10 episodes using the learned policy where agent takes greedy actions. We can observe from the fig 4 that the agent is able to get max reward consistently for all the 10 episodes. This shows our agent has learned the optima policy.

4 Part 2

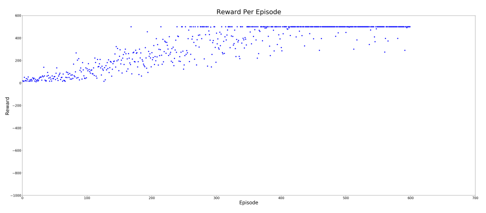
In this section we tested our PPO implementation on two complex environment and discuss the results.

4.1 OpenAI Cartpole

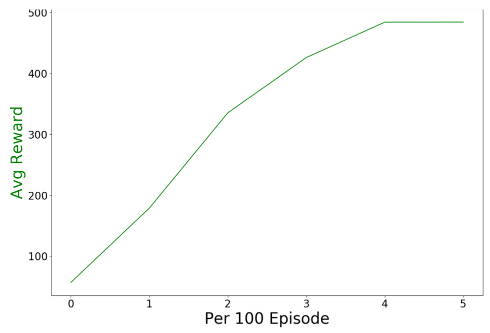
Here we will discuss the results for Cartpole

4.1.1 Training agent on the environment

Fig. 5 shows the reward per episode plot for training the agent on the Cart Pole environment. The input to the network is a state of the environment with properties like cart position, velocity, pole angle and its angular velocity. This represents the position of the agent. Output of the actor network gives the probability of the four actions. While the critic network is gives the state value.



(a) Cart Pole: Reward per episode



(b) Cart Pole: Reward per 100 episode

Figure 5: Train results for Cart Pole Environment

In the figure 10a we can observe that the agent was able to converge very soon and was able to get maximum reward of 475 points in about 650 episodes. For the first 300 episodes the agent didn't perform well which is obvious given the network parameters were not optimised yet. But after 300 episode we see the trend of convergence as agent starts to get maximum reward consistently. We stop the trained if the agent attains average reward of 470 for 200 episodes. Comparing it to Grid world in part one, this was a more challenging environment to work with hence reward graph as we see is more sparse, with variance in rewards.

The figure 7b shows the average reward per 100 episodes. This graph validates our observation that model started to converge in about 100 episodes.

4.1.2 Testing the trained agent on environment

Fig. 6 shows the reward per episode plot when agent only takes the learned greedy actions.

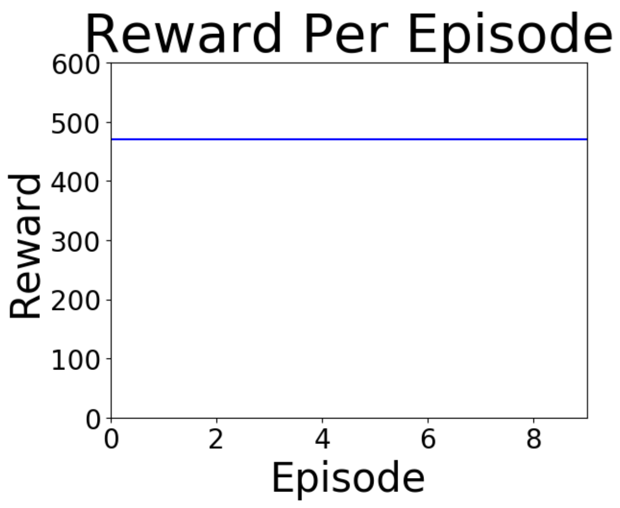
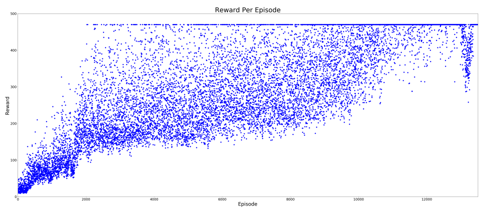


Figure 6: Cart Pole: Reward per episode for learned policy.

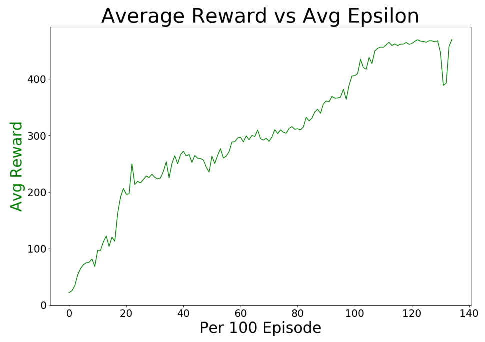
Once we have trained our agent we evaluate it for 10 episodes using the learned policy where agent takes greedy actions. We can observe from the fig 6 that the agent is able to get max reward consistently for all the 10 episodes. This shows our agent has learned the optimal policy with the PPO algorithm.

4.1.3 One-Step actor critic Vs PPO

We initially implemented the one-step actor critic algorithm. Figure 7 shows the results from the training. We obsere that it took more than 10000 episodes for the model to finally converge and stop. Compare to this, with PPO the agent took only 700 episodes to converge in the Cart Pole environment. This proves superior and stable performance of PPO. Since One-Step algorithm updates the parameters too much every time, there is less training stability compared to PPO algorithm that avoids frequent parameter updates and uses clipped surrogate objective



(a) Cart Pole: Reward per episode



(b) Cart Pole: Reward per 100 episode

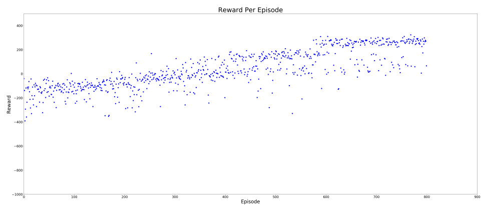
Figure 7: Train results for Cart Pole Environment on Basic Actor Critic

4.2 OpenAI Lunar Lander

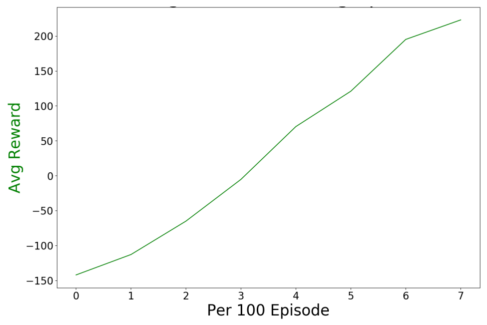
Here we will discuss the results for Lunar Lander Environment.

4.2.1 Training agent on the environment

Fig. 8 shows the reward per episode plot for training the agent on the Cart Pole environment. The input to the network is a state of the environment with lander properties. Output of the actor network gives the probability of the four actions. While the critic network is gives the state value. We trained the agent till average reward for last 200 episodes were less than 200 points.



(a) Lunar Lander: Reward per episode



(b) Lunar Lander: Reward per 100 episode

Figure 8: Train results for Lunar Lander Environment

In the figure 8a we can observe that the agent was able to converge very soon and was able to attain more than 200 points consistently after 500 episodes. Initially the rewards spare which is obvious given the network parameters were not optimised yet. But after 500 episode we see the trend of convergence as agent starts to get maximum reward consistently. We stop the training if the agent attains average reward of 200 for 200 episodes. Comparing it to Grid world and Cart pole results, this was a more challenging environment to work with hence reward graph as we see is more sparse, with variance in rewards. Also the convergence was steady.

The figure 8b shows the average reward per 100 episodes. This graph validates our observation that model started to converge in about 100 episodes.

4.2.2 Testing the trained agent on environment

Fig. 8b shows the reward per episode plot when agent only takes the learned greedy actions.

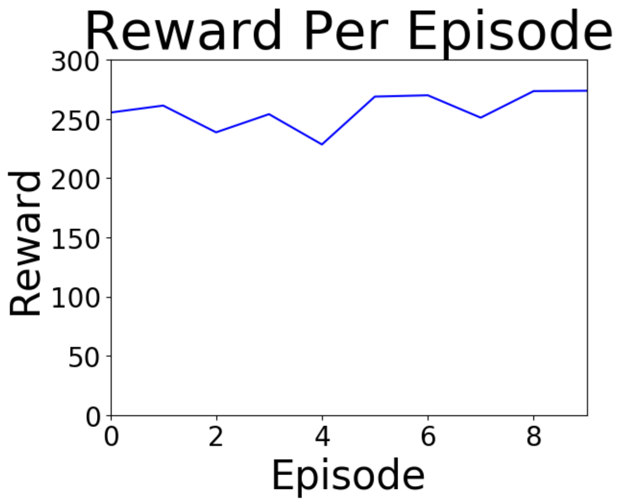


Figure 9: Lunar Lander: Reward per episode for learned policy.

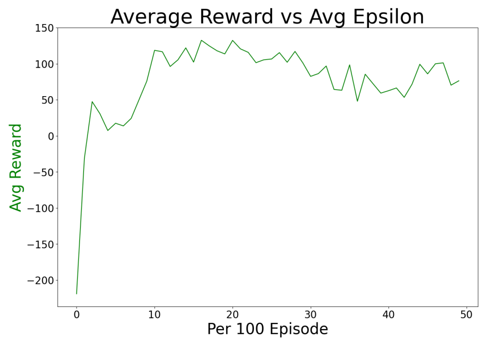
Once we have trained our agent we evaluate it for 10 episodes using the learned policy where agent takes greedy actions. We can observe from the fig 9 that the agent is able to get more than 200 points as reward for all the 10 episodes. This shows our agent has learned the optimal policy with the PPO algorithm.

4.2.3 One-Step actor critic Vs PPO

We initially implemented the one-step actor critic algorithm. Figure 9 shows the results from the training. We trained the agent multiple rounds with One step actor critic but were not able to successfully converge it. Hence we had to move to PPO algorithm. From the A2C algorithm we see that agent rewards peaked around 150 points and then it started to decrease. Figure 9 should the best result that we could achive with A2C after many rounds of training. Compare to this, with PPO the agent took only 700 episodes to converge in the Lunar Lander environment. This proves superior and stable performance of PPO. Since One-Step algorithm updates the parameters too much every time, there is less training stability compared to PPO algorithm that avoids frequent parameter updates and uses clipped surrogate objective

5 Conclusion: Comparing PPO performance for the three environments

From the rewards per episode figures for Mario Grid Environment 3, Cart Pole Environment 5 and Lunar Lander 8 we can clearly see that as the environment becomes more complex the time taken to converge increases. This also depends on the reward distribution. For example, we initially tested PPO with OpenAI's Mountain Car environment but we were not able to converge. This is possible due to the reward distribution where we get -1 reward for every step. We need good exploration in the environment for the agent to succeed. In case of Basic Actor Critic we observed the agent converges to local optimum which is a known disadvantage of Actor Critic algorithms. Due to this reason we chose Proximal Policy Optimization Algorithm which comes under advance actor-critic algorithm with many improvements over the basic actor-critic.



(a) Lunar Lander: Reward per 100 episode

Figure 10: Train results for Lunar Lander Environment on Basic Actor Critic

References

The following resources were used in making of this report.

[1] Lecture slides provided by Dr. Alina Vereshchaka, instructor CSE 546.

[2] Demo jupyter notebooks provided as class material.

[3] Matplotlib visualization base code provided by Nitin Kulkarni, TA CSE 546

[4] OpenAI gym documentation for Cartpole and LunarLander.

[5] PyTorch documentation.

[6] PyTorch discussions.
https://discuss.pytorch.org/t/utility-function-for-calculating-the-shape-of-a-conv-output/11173/4
https://discuss.pytorch.org/t/how-to-calculate-the-output-size-after-conv2d-in-pytorch/20405
https://discuss.pytorch.org/t/center-crop-in-pytorch/3214

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

Grade Split

Team Member	Code	Report
Ankit Shaw	50	50
Rounak Biswas	50	50
Total	**100**	**100**