Assignment 3: Actor-Critic

Actor-Critic based algorithms learn both the value function and the policy whereas in value based it learns the value function and in policy based there is no value function, only learns the policy. In Actor-Critic we maintain two sets of parameters:

- Critic: Updates action-values function parameter w
- Actor: Updates policy parameter θ, in direction suggested by the critic

In this project we have implemented Advantage Actor Critic (A2C) algorithm. In this algorithm the critic estimates the advantage function which can be estimated using two function approximators and two parameter vectors.

$$V_v(s) \approx V_{\mu\theta}(s)$$
$$Q_w(s,a) \approx Q_{\mu\theta}(s,a)$$
$$A(s,a) = Q_w(s,a) - V_v(s)$$

The advantage function significantly reduces variance of policy gradient.

There are various differences between actor-critic and value based approximation algorithms such as:

1. In value based we estimate the value or action-value function. Whereas in actor-critic we estimate both the value function (critic) and the policy (actor).
2. Also, in value based the policy is extracted directly from the value function whereas in actor-critic policy is learnt.
3. Value based algorithms can be used on discrete action space. Actor-critic algorithms like DDPG can be applied to continuous action space.
4. Another difference is in the exploration methodology used in both the algorithms: actor-critic uses softmax function in the actor network and value-based uses epsilon greedy method.
5. Actor-critic based algorithms require more samples to learn compared to value-based which require comparatively less.

1. Environments:

In this project we have used three environments:

- Grid World Environment
- CartPole-v1
- Acrobot-v1

1.1. Grid World

The environment used is a grid world in which the Robot must explore and reach the bag of gold while collecting intermediate gold rewards. While exploring for the gold in the world the robot must avoid the monster and avoid falling into the pit.

States

The environment is a grid of 5x5 matrix. All the states can be represented using X and Y co-ordinates

S = { [4,0], [4,1], [4,2], [4,3], [4,4],
[3,0], [3,1], [3,2], [3,3], [3,4],
[2,0], [2,1], [2,2], [2,3], [2,4],
[1,0], [1,1], [1,2], [1,3], [1,4],
[0,0], [0,1], [0,2], [0,3], [0,4]
}

Actions

There are four possible actions that the robot can take.

$$A \in \{Up, Down, Left, Right\}$$

Rewards

There are 4 positive rewards that the robot can receive. Out of 4, 3 rewards (+2, +2, +0.5) are intermediate rewards and final reward is for reaching the goal (+25) i.e. collecting the bag of coins. If the robot falls in the pit (-5) or gets eaten by the monster (-10), then the robot receives a negative reward. Additionally, for each action step taken by the robot it receives a negative reward (-0.5) and if the robot tries to go outside the grid then it receives a negative reward (-1).

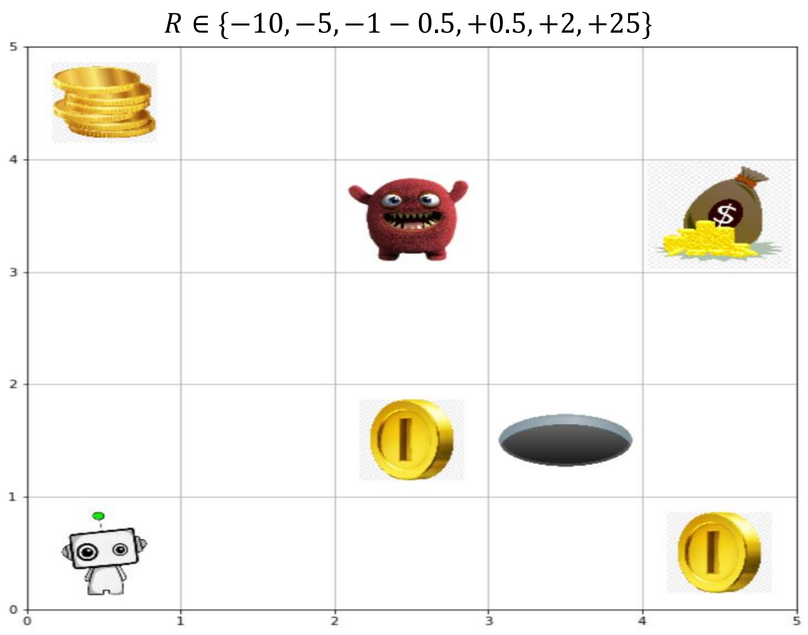


Figure 1: Visual Representation of the environment

- Agent starts at s = [0,0]
- Final goal position (bags of coins +10) is at s = [4,3]
- Intermediate Rewards 1 and 3 (+2) are at s = {[2,1], [4,0]}
- Intermediate Reward 2 (+0.5) is at s = [0,4]
- Pit (-5) is present at state s = [3,1]
- Monster (-10) is present at state s = [2,3]

1.2. CartPole-v1

In this environment, a pole is attached to a cart and the pole moves across the frictionless surface. Here, the goal is to balance the cartpole i.e., prevent it from falling by sliding the cart left or right.

Observation:

Number	Observation	Minimum	Maximum
0	Cart Position	-4.8	4.8
1	Cart Velocity	-∞	∞
2	Pole angle	-0.418 rad (-24°)	0.418 rad (24°)
3	Pole angular velocity	-∞	∞

Action space:
For cartpole-v1 environments, the cart can only take two actions to balance the pole above the cart.

$$A = \{Left, Right\}$$

Rewards:
The reward will be +1 for each step, including the termination step. The maximum reward cartpole can reach is 0.

$$R = \{+1\}$$

Starting state:
All observations are assigned a uniform random value in $[-0.05\ldots0.05]$

Episode termination:

1. Pole angle > ±12°
2. Cart position > ± 2.4 (center of the cart reaches the edge of the display)
3. Episode length > 500
4. Solved requirements:
   4.1. Average return >= 475.0 over 100 consecutive episodes.

1.3. Acrobot-v1:

In the acrobot system there are two joints and two links, where the joint between the two links is actuated. Initially the links are hanging downloads and the goal is to swing the lower part of lower link to a given height.

Observation:

The sate consists of the sin() and cos() of the two rotational joint angles and the join angular velocities:

$$[cost(theta1)\ \sin(theta1)\ \cos(theta2)\ \sin(theta2)\ thetaDot1\ thetaDot2]$$

Action Space:

The action is either applying +1, 0, or -1 torque on the joint between the two-pendulum links

$$A = \{-1, 0, +1\}$$

Rewards:

For every action the agent receives a reward of -1.

Solved Requirement:

If the average reward for 100 episodes is greater than -80 then it is considered the environment is solved and the agent has learnt.

2. Results

Here we talk about the results we received using the A2C algorithms on the above-mentioned environments

2.1. A2C on Grid World Environment:

In the below figure 2 we can see the total reward per episode graph. Initially the agent is trying to learn the value function and policy hence we can see fluctuations in the graph. But as the agent learns more its policy gets better and the agent gets the maximum possible reward it can achieve. Around 200 odd episodes we can see that the agent has learnt and only receives the maximum reward.

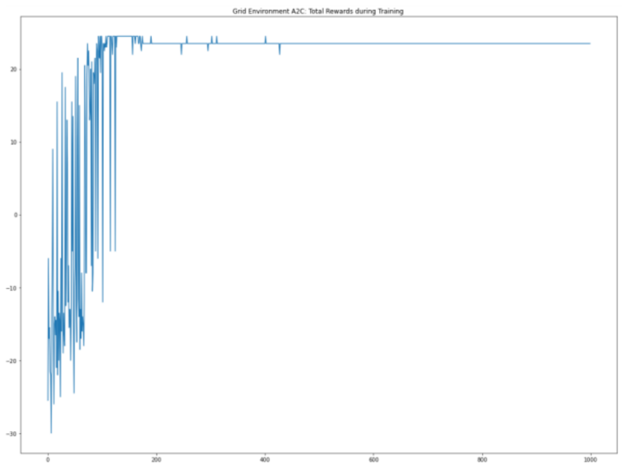


Figure 2: Training results on Grid World Environment

After training, evaluating the agent:

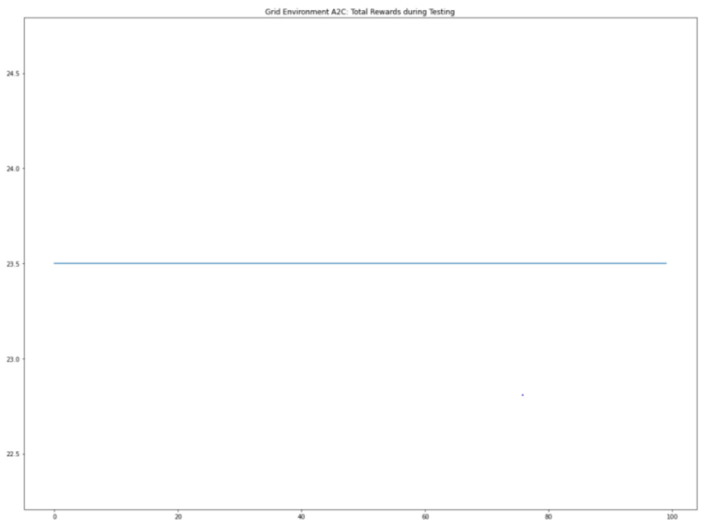


Figure 3: Test results on Grid World Environment

The above graph (Figure 3) showcases the total reward per episode for 100 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

2.2. A2C on CartPole-v1:

In the below figure 4 we can see that till 550 odd episodes the agent was unable to learn much from the environment. But as the training went on it tried to learn the policy and optimal q-values to receive higher rewards. It was able to reach the optimal reward of more than 475 for multiple times but was unable to hold that for many episodes.

We had set criteria where if the agent receives on average for last 15 episodes more than 475 then we stop the training. This criterion was met around 1800 odd episode and the training was stopped. Here, we observe that the agent has learnt and receives maximum possible reward.

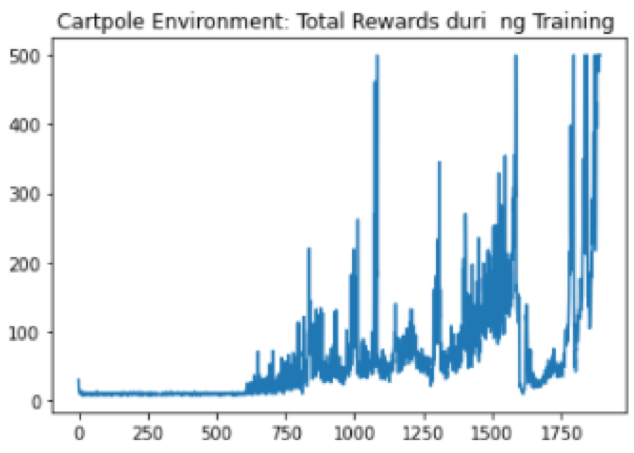


Figure 4: Training results on CartPole

After training, evaluating the agent:

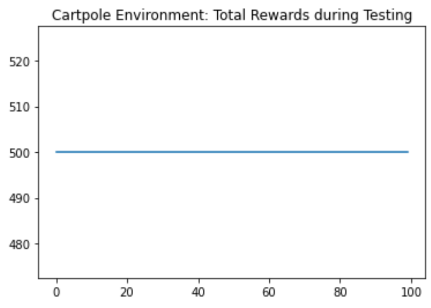


Figure 5: Testing result on CartPole

The above graph (Figure 5) showcases the total reward per episode for 100 episodes. The agent learns the optimal policy and follows the path to obtain the maximum reward.

2.3. Acrobot-v1:

In the graph below (figure 6) we can see that after 750 odd episodes the dips in the graph have decreased and it tried to converge. Being the environment to be complex compared to grid world and cartpole it is difficult to converge. We have set an exit criterion for training that if the average of the last 100 episodes is more than -80 then stop the training and hence the training got stopped early.

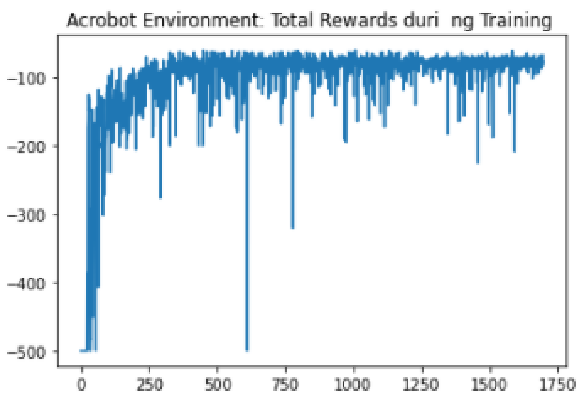
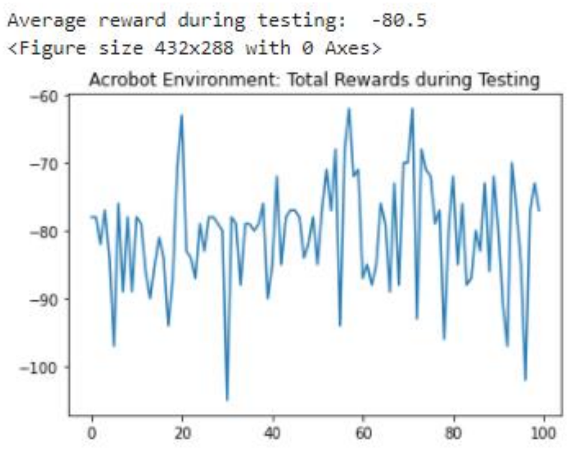


Figure 6: Training results for A2C on Acrobot-v1

After training, evaluating the agent:
During testing we can see that the average reward for 100 episodes is -80.5 which is solved. (Referred from the mentioned link)



```
Average reward during testing:  -80.5
<Figure size 432x288 with 0 Axes>
```

Figure 7: Testing results on Acrobot-v1

3. Contribution Summary

Team Member	Assignment part	Contribution (%)
Anup Atul Thakkar	Part 1,2	50%
Sagar Jitendra Thacker	Part 1,2	50%