Exercise 6

Gengcong Yan - 1009903
ELEC-E8125 - Reinforcement Learning

November 7, 2021

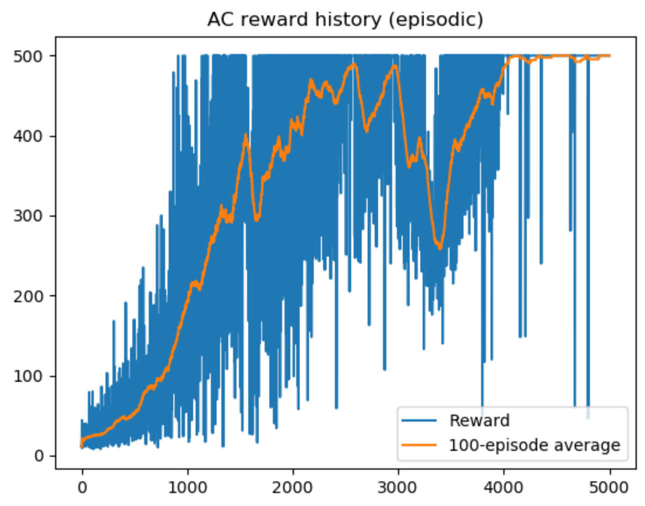

Figure 1: Reward History (episodic)

1 Task 1

```
# Partial Code in Func update_policy in agent.py

_, next_values = self.policy.forward(next_states,self.variance)
_, allvalues=self.policy.forward(states,self.variance)

next_values=next_values.reshape(-1)
allvalues=allvalues.reshape(-1)

Q_pred=rewards+self.gamma*next_values*(1-done)
c_loss=torch.sum((Q_pred-allvalues).pow(2))
advantages=Q_pred-allvalues

a_loss=torch.sum(-(advantages.detach())*action_probs)


# TODO: Compute the gradients of loss w.r.t. network parameters
# Or copy from Ex5
ac_loss=a_loss+c_loss
self.optimizer.zero_grad()
ac_loss.backward()
# TODO: Update network parameters using self.optimizer and zero
    gradients
# Or copy from Ex5
self.optimizer.step()
```

The plots in Fig.1 shows the reward history of actor-critic method under episodic update.

2 Question 1

Although the REINFORCE-with-baseline method learns both a policy and a state-value function, we do not consider it to be an actor–critic method because its state-value function is used only as a baseline, not as a critic. That is, it is not used for updating the value estimate for a state from the estimated values of subsequent states, but only as a baseline for the state whose estimate is being updated. When the state-value function is used to assess actions in this way it is called a critic, and the overall policy-gradient method is termed an actor–critic method.

3 Question 2

The value of advantage means how much better it is to take a specific action compared to the average, general action at the given state. It's the difference of q value and the average of actions which it would have taken at that state. It tells about the extra reward that could be obtained by the agent by taking that particular action.

4 Task 2

The plot in Fig.2 shows the reward history of actor-critic method with non-episodic update. The curve was not very stable and showed different variations in different attempts

```python
# partial code of non episodic update in 'cartpole.py'

#Update every 50 steps
step=0
# Run actual training
for episode_number in range(train_episodes):

    ......

    while not done:

        ......

        step+=1
        if (step)%50==0:
            # Let the agent do its magic (update the policy)
            update_counts+=1
            step=0
            agent.update_policy(episode_number)

    .......
```

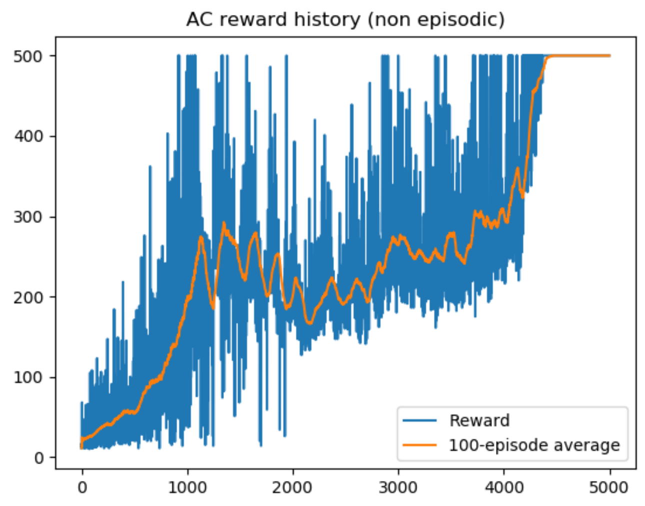

Figure 2: Reward History (non episodic)

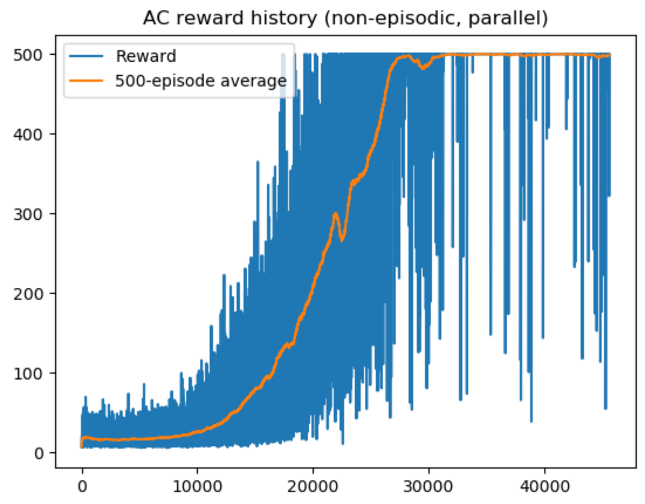

Figure 3: Reward History (non episodic parallel)

5 Task 3

The plot in Fig.3 shows the reward history of parallel actor-critic method with non-episodic update.

6 Question 3

NO.It's not the same. the multiple runs of training means running multiple agents at the same time, their information is not shared, and the variance is reduced by averaging the results separately. In contrast, the parallel synchronization algorithm contains multiple agents that share information among themselves to make the results more optimal. We update our policy every 50 steps, but we have 64 threads in parallel, then combine their results as a batch of 3200 for update, the information learned from different agents are merged together during the update. It's easy to synchronize data collection across multiple parallel processes.

7 Question 4

I think that the parameters in the value function are randomly initialized in the initial stage of actor-critic algorithm, so the reward will be poor at the beginning of the training. But REINFORCE algorithm use real values based on Monte Carlo returns, which are more accurate information in the initial stage. So REINFORCE seems to completely outperform all tested A2C in the beginning. But When the learning reaches a certain level and the parameters in the value function are learned with real meaning, the bias decreases and better results are obtained.

8 Question 5

8.1 Question 5.1

In the case of RL, variance now refers to a noisy, but on average accurate value estimate, whereas bias refers to a stable, but inaccurate value estimate.

REINFORCE is an unbiased offline algorithm, in that it uses Monte Carlo estimates of the gradient after completing an episode. Because G_t is a discounted sum of all rewards until the end of the episode, which is affected by all actions taken from state S_t. Therefore, variance of G_t is high.

Actor-Critic is a biased online algorithm that updates using predictions of future return. Monte Carlo estimates are unbiased but high variance. Actor-Critic is unbiased only if the critic gives you the exact expected return under the current policy. Otherwise, the target value will not be unbiased. if we use a neural network-based approximation of $v(s)$, actor critic will be biased. There is a lot of bias in the beginning of training where the estimate value function is far from real value function, and as the estimate improves, the bias subsides.

8.2 Question 5.2

The actor-critic relies on a value estimate rather than a Monte-Carlo roll out there is much less stochasticity in the reward signal, since our value estimate is relatively stable over time. The problem is that the signal is now biased, due to the fact that our estimate is never completely accurate. Advantage learning in actor-critic is an important solution for balancing the bias-variance trade-off. Variance can simply be subtracted out from a Monte-Carlo sample using a more stable learned value function in the critic. This value function is typically a neural network, and can be learned with different methods. The resulting advantage $A(s, a)$ is then the difference between the two estimates. This advantage estimate how much better the agent actually performed than was expected on average.

The other strategy is n-step use in updating policy. We don't want to only consider the last action, but we also don't want to consider all actions made. so we consider N steps. N-step uses the n next immediate rewards and approximates the rest with the value of the state visited n steps later. Instead of waiting for an entire episode to complete before collecting a trajectory of experience, we can break experience batches down into smaller sub-trajectories, and use a value-estimate to bootstrap when that trajectory doesn't end with the termination of the episode. N-step estimation ensures a trade-off between bias and variance.

9 Question 6

Policy gradient and actor-critic methods can obtain good performance on large and continues action space. And they can solve complicated environment where the optimal policy is stochastic, because we can not compute possibilities of every action, a deterministic agent could lead a better performance. learning stochastic policies means better performance under partially observable environments. Because we can learn arbitrary probabilities of actions, the agent is less dependent on the Markov assumption. And sometimes it could be more straightforward for function approximation to represent a policy than a value function. And the policies are parameterized with continuous values, the action probabilities change smoothly as a function of the learned parameters. Therefore, policy-based methods often have better convergence properties.

10 Feedback

1. 10h