Deep reinforcement learning - Assignment 2

Gil Lifshits - 322597717 **Amiel Giloni - 311453997**

December 25, 2021

1 Monte-Carlo Policy Gradient (REINFORCE)

1.1

The advantage is defined as the difference between the Q-function and the value-function, and reflects the additional rewards that could be obtained from taking an action in a certain state. Due to the high variance caused by the sampled return, the basic REINFORCE algorithm is likely to converge slowly. Therefore, it's better to use the advantage function instead of the return itself, in order to reduce the gradient update size and to mitigate the high variant problem.

1.2

The prove of the following equation is presented in the following subsection:

$$E_{\pi_\theta}[\nabla log \pi_\theta(a_t|s_t)b(s_t)] = 0 \tag{1}$$

Using the expectation definition on equation 1 yield the following result:

$$E_{\pi_\theta}[\nabla log \pi_\theta(a_t|s_t)b(s_t)] = \int \nabla log \pi_\theta(a_t|s_t)b(s_t)\pi_\theta(a_t|s_t)d(a_t|s_t) \tag{2}$$

With the identity in equation 3, expression 2 can written as follow:

$$\nabla log \pi_\theta(a_t|s_t) = \frac{\nabla \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \tag{3}$$

$$E_{\pi_\theta}[\nabla log \pi_\theta(a_t|s_t)b(s_t)] = \int \nabla \pi_\theta(a_t|s_t)b(s_t)d(a_t|s_t) \tag{4}$$

With the prerequisite condition, which state that the baseline function - b(s) does not depend on the action, equation 4 can be written as follow:

$$\int \nabla \pi_\theta(a_t|s_t)b(s_t)d(a_t|s_t) = b(s_t)\nabla \int \pi_\theta(a_t|s_t)d(a_t|s_t) = b(s_t)\nabla 1 = 0 \tag{5}$$

hence,

$$E_{\pi_\theta}[\nabla log \pi_\theta(a_t|s_t) b(s_t)] = 0 \tag{6}$$

1.3 Reinforcement with value-function approximation baseline

The NN structure of the policy network is show in the following table:

Layer	Layer type	Layer size	Activation
Input	dense	4	
Hidden	dense	12	ReLU
Output	dense	2	linear

The NN structure of the value-function network is show in the following table:

Layer	Layer type	Layer size	Activation
Input	dense	4	
Hidden	dense	64	ReLU
Output	dense	1	linear

The hyper-parameters that were used are presented in the following table:

Learning rate - Value network	Learning rate - Policy network	Discount factor
0.0002	0.0004	0.99

In order to compare between the base REINFORCE algorithm and baseline REINFORCE algorithm, we tested each algorithm four times and plotted the episode score and the mean episode score over 100 episodes in the following figures:

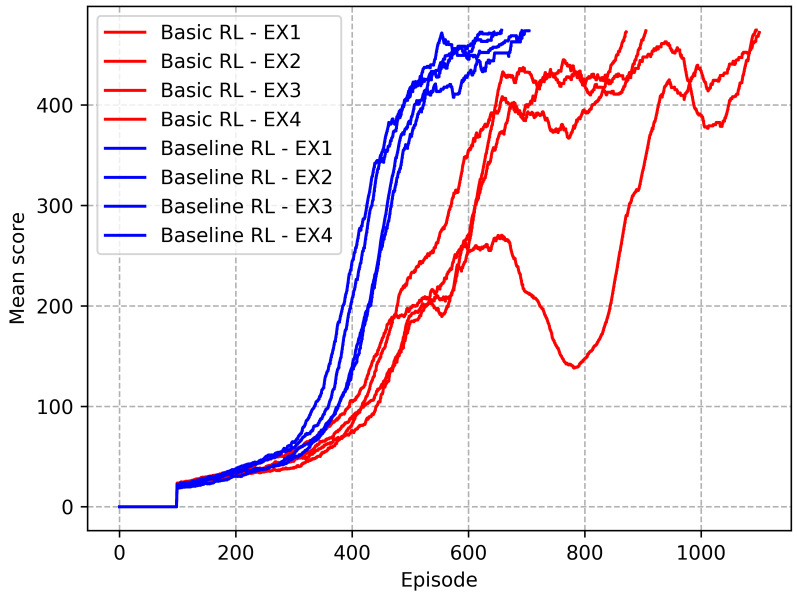

Figure 1: Mean episode score over 100 consecutive episode

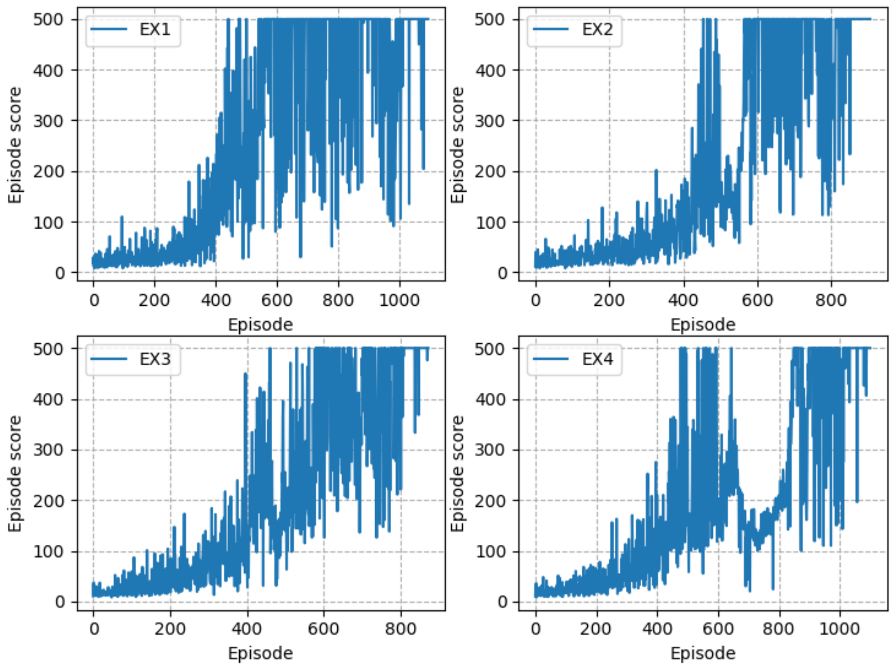

Figure 2: Reward per episode - Base REINFORCE algorithm

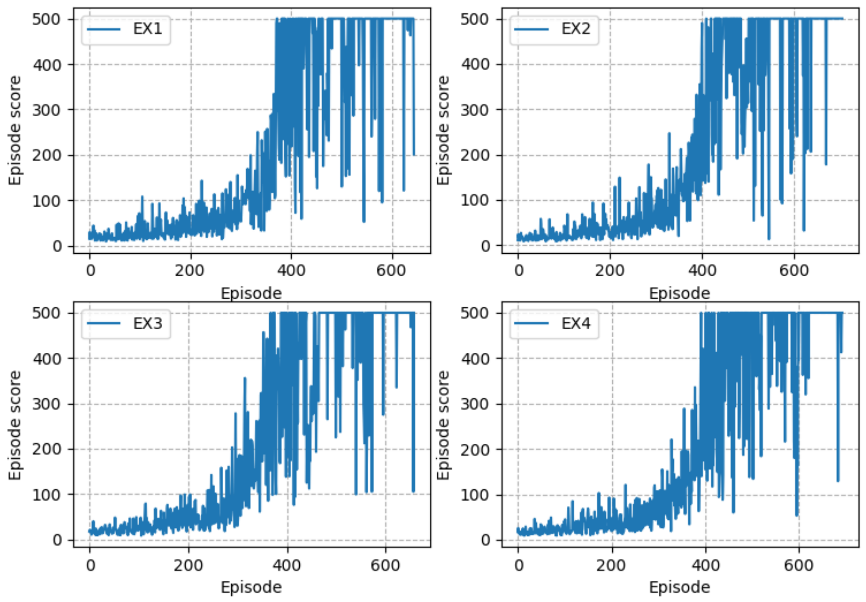

Figure 3: Reward per episode - Baseline REINFORCE algorithm

From figure 1, it can be observed that the baseline REINFORCE algorithm with outperformed the basic REINFORCE algorithm. The basic REINFORCE algorithm converges on average after 993 episodes (with a standard deviation of 105 episodes), and the baseline REINFORCE algorithm converges after 674 episodes (with a standard deviation of 25 episodes). These results accrued due to the reduction of the gradient step size, which reduces the effect of the high variance in the sampling process. However, the average converging time of the basic REINFORCE is 9 minutes and the average converging time of the baseline REINFORCE algorithm is almost 17 minutes, i.e., almost twice as much. These results occur due to the implication of the baseline, which requires to estimate the advantage-function with another NN and therefore, increase the computation time and the convergence time respectively. From figure 2 and figure 3 it can be seen that the baseline RL is more stable compared to the basic RL, which emphasizes the positive effect of reducing the gradient step size on the sampling process variance and the converging time.

1.4 Instruction for running the scripts

Please run the `baseline_policy_gradients.py` script in order to observed the baseline REINFORCE algorithm performing on the CartPole-v1 environment.

2 Advantage Actor-Critic

2.1

Given that the TD error is defined as:

$$\delta_{\pi_\theta} = r + \gamma v_{\pi_\theta}(s') - v_{\pi_\theta}(s) \tag{7}$$

Than we can derive the advantage function using the expectation of the TD error:

$$\begin{aligned} E_{\pi_\theta}[\delta_{\pi_\theta}|s,a] &= E_{\pi_\theta}[R_{t+1} + \gamma v_{\pi_\theta}(S_{t+1})|s,a] - v_{\pi_\theta}(s) \\ &= Q_{\pi_\theta}(s,a) - v_{\pi_\theta}(s) \\ &= A_{\pi_\theta}(s,a) \end{aligned} \tag{8}$$

Therefore, using the TD error of the value function for the update of the policy network parameters is the same as using the advantage estimate.

2.2

The policy-function - π is the actor function and the value-function - v is the critic function.

The **critic** estimates the value function. The critic used to inform the actor how good was the action and how it should adjust. This could be the action-value (Q value) or state-value (V value).

The **actor** is deciding which action should be taken according to the policy distribution. The actor is updated according to the direction suggested by the critic (such as with the policy gradients).

2.3 Implementation of Actor-Critic algorithm

The NN structure of the policy network is the same as the previous section, and the NN structure of the value-function network is show in the following table:

Layer	Layer type	Layer size	Activation
Input	dense	4	
Hidden	dense	64	ELU
Hidden	dense	16	ELU
Output	dense	1	linear

The hyper-parameters that were used are presented in the following table:

Learning rate - Value network	Learning rate - Policy network	Discount factor
0.0007	0.0004	0.99

In order to compare between the baseline REINFORCE algorithm and the Actor-Critic algorithm, we tested the Actor-Critic algorithm four times and plotted the episode scored and the mean episode score over 100 episodes in the following figures:

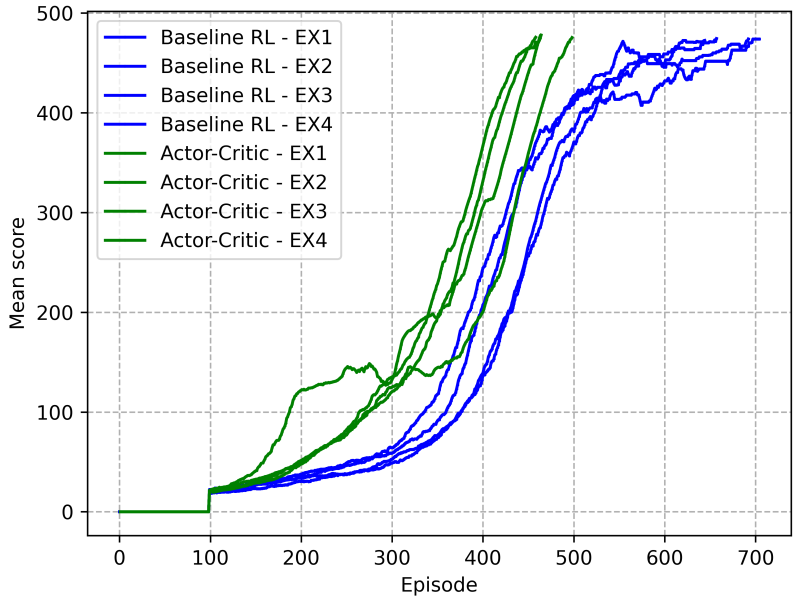

Figure 4: Mean episode score over 100 consecutive episode

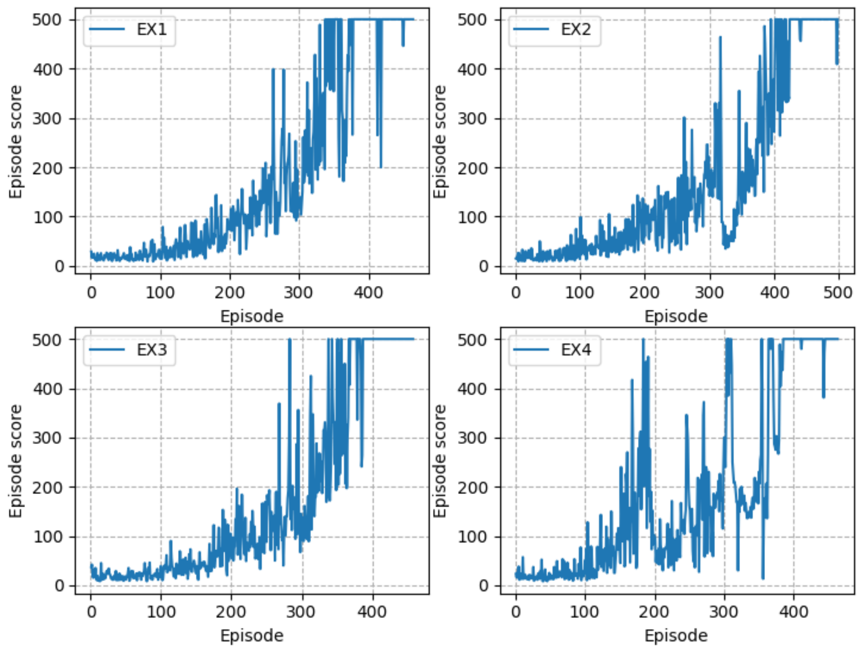

Figure 5: Reward per episode - Actor-Critic algorithm

From figure 4, it can be observed that the Actor-Critic outperformed the baseline REINFORCE algorithm. The Actor-Critic converges on average after 471 episodes (with a standard deviation of 16 episodes), and (as seen before) the baseline REINFORCE algorithm converges after 674 episodes (with a standard deviation of 25 episodes). Meaning, the Actor-Critic converges faster compared to the baseline REINFORCE algorithm, and with a lower standard deviation indicating the high stability of the model. The better stability of the Actor-Critic algorithm relative to the baseline REINFORCE algorithm can also be seen by comparing figure 5 and figure 3, which shows the episode score over the training process of the Actor-Critic and the baseline REINFORCE algorithm respectively. In addition, the time took for the Actor-Critic to converge was less than the time took the baseline REINFORCE (13 minutes versus 17 minutes). This indicates that not only the Actor-Critic reaches higher rewards faster, it does not come with a trade off of run-time to reach convergence. The improvement in converging time can be expected, due to the fact that the Actor-Critic updates the policy and value-function estimation at the end of each step, unlike the baseline RL that can only perform the update when reaching the end of the episode.

2.4 Instruction for running the scripts

Please run the `actor_critic.py` script in order to observed the Actor-Critic algorithm performing on the CartPole-v1 environment.

3 Python packages versions

- Gym - 0.21.0
- TensorFlow - 2.5
- TensorBoard - 2.5
- NumPy - 1.21.4