Assignment 2 - Policy Gradient Methods

Efrat Ravid 311149942
Or Oxenberg 312460132

Section 1

***Question -** What does the value of the advantage estimate reflect? Why is it better to follow the gradient computed with the advantage estimate instead of just the return itself?*

Answer - Following the gradient computed with the advantage estimate instead of the return value itself is better as it reduces the variance in the gradients, leading to faster and more stable convergence. The value of the advantage estimate reflects how much better the current action would be compared to the return based on an “average” action.

***Bonus Question -** The reduction of the baseline does not introduce bias to the expectation of the gradient since:*

$$\mathrm{E}_{\pi_\theta}[\nabla log\pi_\theta(a_t|s_t)b(s_t)] = 0$$

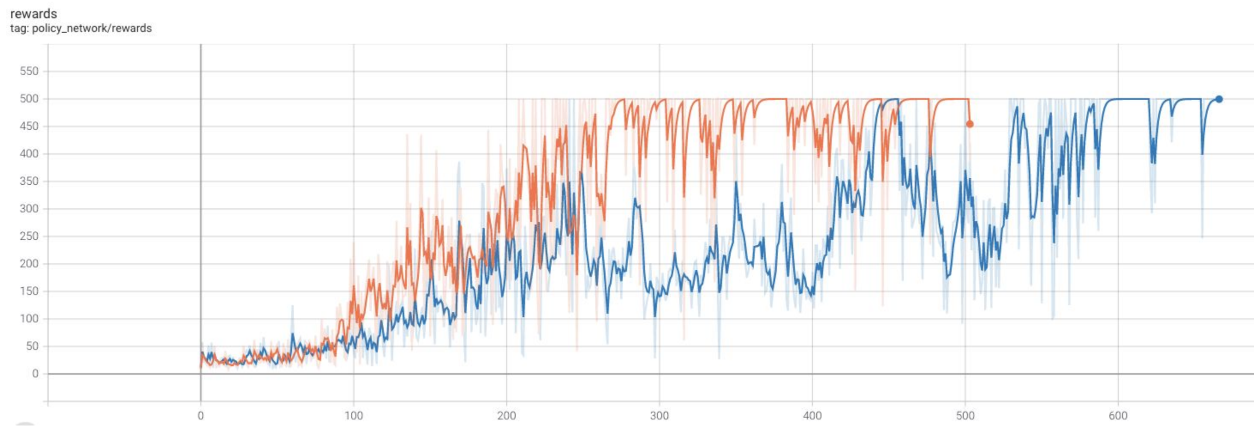
What is the prerequisite condition for that equation to be true?

Answer - The prerequisite condition for the equation to be true is that the baseline term $b(s_t)$ is independent on a and does not vary with a.

$E[\ \nabla_\theta \pi_\pi(s,a)B(s)] \ = \ \sum_s p(s) \sum_a \ \nabla_\theta \pi_\pi(s,a)B(s) \ = \sum_s p(s) \nabla_\theta B(s) \sum_a \ \pi_\pi(s,a) \ = \ 0$ we can make this transition only if the B is a function of state s but not of action a. After that the sum of the actions will be 1 (as it is a probability function) and the derivative of 1 is zero so:
$E[\ \nabla_\theta \pi_\pi(s,a)B(s)] \ = \ 0$

Algorithm Results

The following graph shows the sum of rewards per episode. The blue graph is when the agent used REINFORCE without baseline and the orange graph is when the agent learned with the baseline. Both agents were trained until achieving an average reward of over 475 in 100 consecutive episodes.



As expected, using a baseline reduces the variance in the gradient resulting in the agent learning faster. The agent who learned with the baseline converged ~25% faster.

How to run the script

You may find this section's code in the policy_gradients.py file. The script has both a main() function which you can call or an entry point for running. The best hyperparameters found by a grid search are hardcoded. To run REINFORCE with baseline, make sure the USE_BASELINE flag is set to true. If it is set to false, the agent will learn based on REINFORCE without baseline.

Section 2

Question - *Why is using the TD-error of the value function for the update of the policy network parameters practically the same as using the advantage estimate? Prove.*

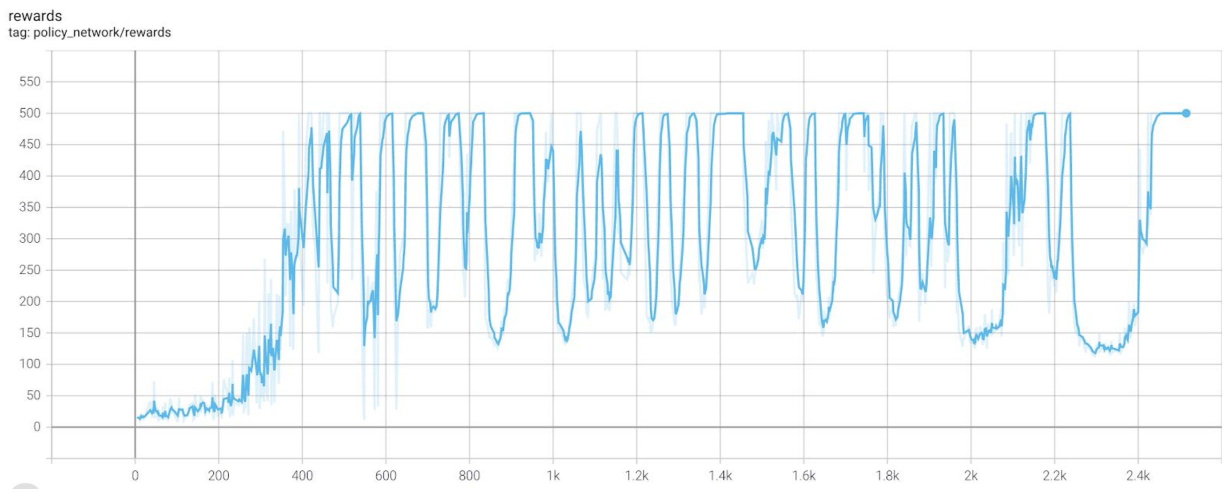
Answer - We can look at the gradient for the advantage function, this function calculate by: $Q(s,a) - V(s)$ this function tells us the advantage for taking a specific action.
We can say that the Q here can be define by : $Q(s,a) = r(s,a) + \gamma V*(\delta(s,a))$ where the v* is the best value for this state by optimal policy.
If we take the Q equation to equation 1 we get the TD error.

Question - *Explain which function is the actor and which is the critic of the model and what is the role of each one.*

Answer - The policy function is the actor and the value function is the critic. The actor takes in the current environment state and determines the best action to take from there - he acts. The critic evaluates the actors performance by returning a score that represents how apt the action is for the state.

Algorithm Results

The following graph shows the sum of rewards per episode for a One-Step Advantage Actor Critic agent. After a hyperparameter tuning process, the agent was trained until achieving an average reward of over 475 in 100 consecutive episodes.



The A2C agent converges slower than the REINFORCE agents both with baseline and without. We believe this is a result of this specific task, meaning the Cartpole environment is less suitable for an online algorithm because it benefits from looking ahead at all the rewards of an episode - the goal is to balance the pole as long as you can. When you evaluate the value function at each step you can't see the benefit of the number of steps in the episodes, that will lead to “noise” like effect in your learning and create large variance reward results.

How to run the script

You may find this section’s code in the actor_critic.py file. The script has both a main() function which you can call or an entry point for running. The best hyperparameters found by a grid search appear in the main function. This script depends on the classes defined in the policy_gradients.py script and assumes they are in the same root directory.