Exercise 6

Jari Mattila - 35260T
ELEC-E8125 - Reinforcement Learning

November 7, 2021

Task 1

The training performance plot for Task 1 is presented in Figure **??** for the continuous Cart-pole environment. The variance learned as a parameter of the network is applied and TD(0) updates are performed at the end of each episode.

Source files: cartpole_task1.py, agent.py

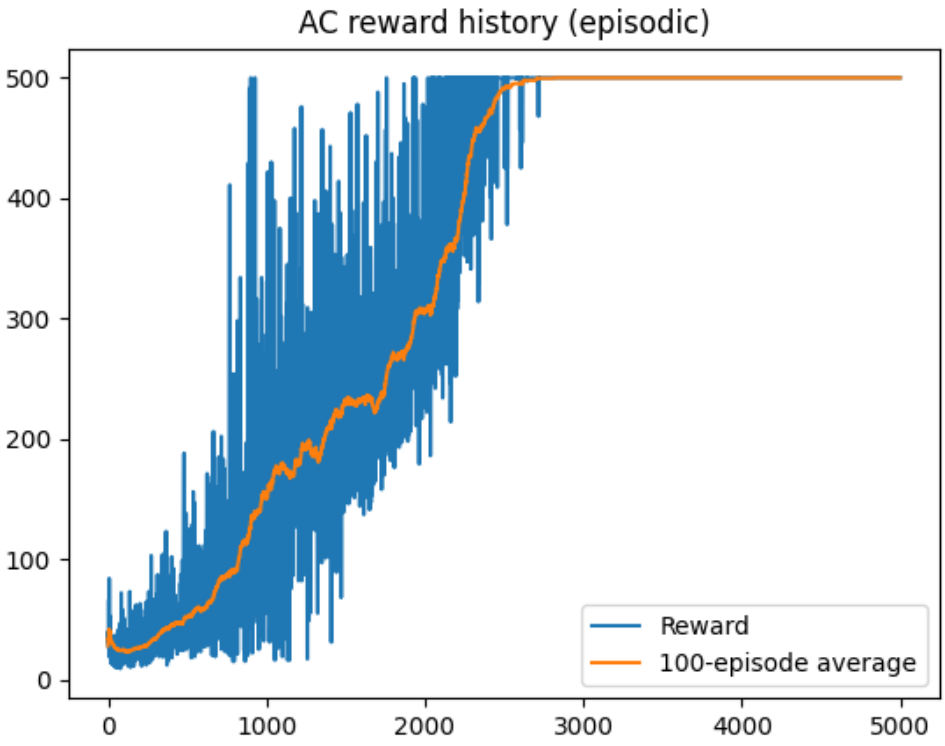

Figure 1: Training performance using an episodic version of actor-critic algorithm.

Question 1

What is the relationship between actor-critic and REINFORCE with baseline?

In the actor critic algorithm, the baseline in the REINFORCE with baseline method is replaced with the state value function that is updated along episodes by means of least squares over gathered data.

Question 2

How can the value of advantage be intuitively interpreted?

The value of advantage corresponds to the advantage of choosing a particular action a over the current policy, i.e., how much better is taking action a compared to average (see slide 12 of Lecture 6).

Task 2

The training performance plot for Task 2 is presented in Figure **??**. The actor-critic algorithm was updated to perform TD(0) updates every 50 timesteps, instead of updating the network at the end of each episode.

Source files: cartpole_task2.py, agent.py

Task 3

The training performance plot for Task 3 is presented in Figure **??**. The actor-critic algorithm was updated to use parallel data collection. The training was executed using default parameters (8 processes, 8 envs per process) that correspond to 64 parallel environments.

Source files: parallel_cartpole.py, agent.py

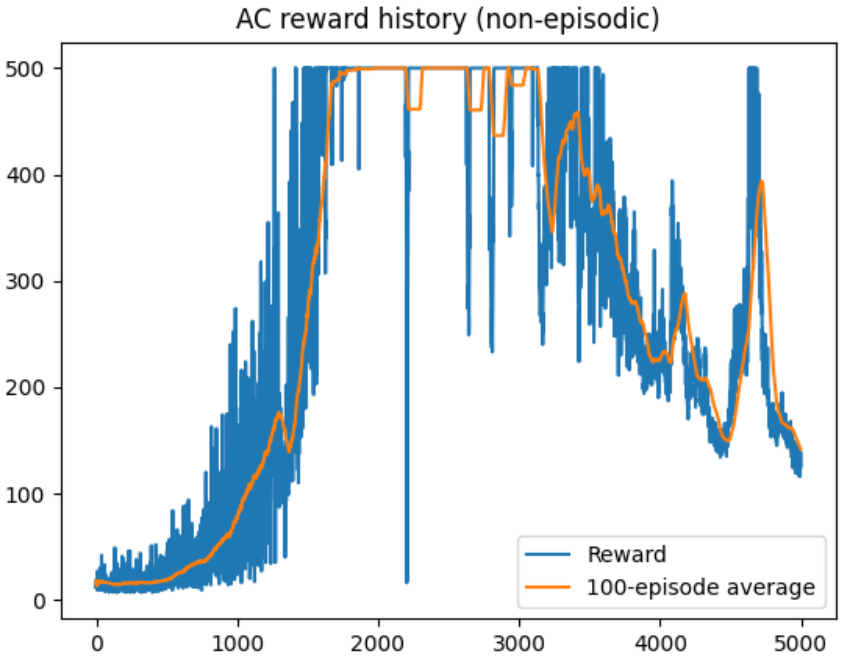

Figure 2: Training performance using a non-episodic version of actor-critic algorithm.

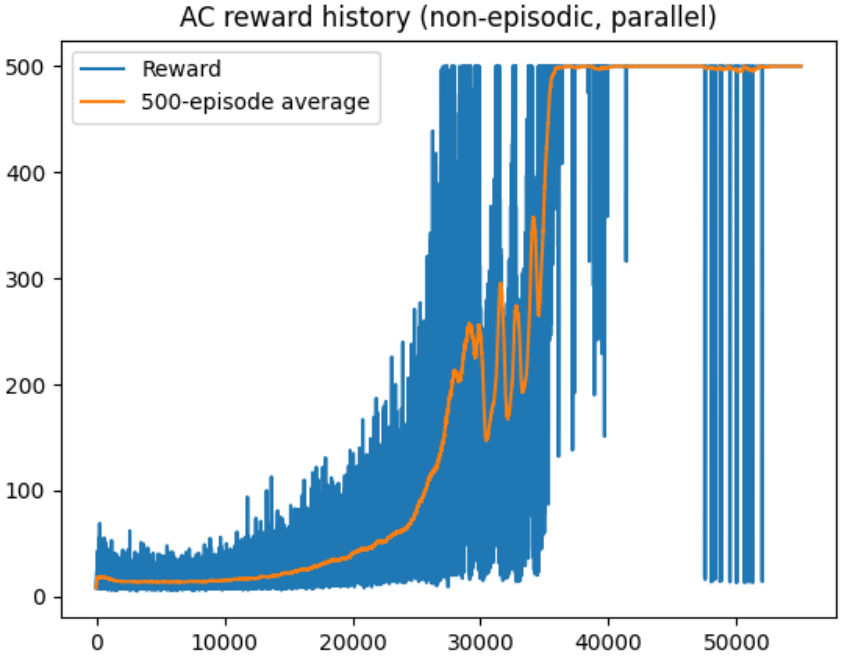

Figure 3: Training performance using a non-episodic parallel version of actor-critic algorithm.

Question 3

How is parallel data collection different from the parallelism in multiple_cartpoles.py script we've seen in Exercises 1 and 5? Can it replace multiple runs of the training algorithm for comparing RL algorithms? **Explain your answer.**

Actor Parallel Training When you use parallel computing with AC agents, each worker generates experiences from its copy of the agent and the environment. After every N steps, the worker computes gradients from the experiences and sends the computed gradients back to the client agent (the agent associated with the MATLAB® process which starts the training). The client agent updates its parameters as follows.

For asynchronous training, the client agent applies the received gradients without waiting for all workers to send gradients, and sends the updated parameters back to the worker that provided the gradients. Then, the worker continues to generate experiences from its environment using the updated parameters.

For synchronous training, the client agent waits to receive gradients from all of the workers and updates its parameters using these gradients. The client then sends updated parameters to all the workers at the same time. Then, all workers continue to generate experiences using the updated parameters.

Question 4

Figure 1 shows the training performance for all three actor-critic variants and the REINFORCE algorithm from the last lecture. In terms of initial performance, REINFORCE seems to completely outperform all tested A2C flavours on Cartpole, despite being a simpler algorithm. **Why is it so? Explain your answer.**

An explanation for the high variance could be because of the returns. For REINFORCE with baseline, I was able to use whitening rewards to normalize the rewards to reduce variance. That was because the algorithm samples the whole trajectory before applying updates, so I was able to apply normalization.

Question 5.1

How do actor-critic methods compare to REINFORCE in terms of bias and variance of the policy gradient estimation? **Explain your answer.**

The actor-critic methods may learn faster than policy gradient because variance of policy gradient estimate is reduced.

Question 5.2

How could the bias-variance tradeoff in actor-critic be controlled?

Eligibility traces should be used to bring us toward Monte Carlo methods, but not all the way there. In the future it may be possible to more finely vary the trade-o between TD and Monte Carlo methods by using variable , but at present it is not clear how this can be done reliably and usefully (Sutton page 317).

Running with the same data multiple times to avoid bad initialisation.

Question 6

What are the advantages of policy gradient and actor-critic methods compared to action-value methods such as Q-learning? **Explain your answer.**

Actor-critic approaches allow addressing continuing problems and continuous action spaces.

However, they are actually different internally. The most fundamental differences between the approaches is in how they approach action selection, both whilst learning, and as the output (the learned policy). In Q-learning, the goal is to learn a single deterministic action from a discrete set of actions by finding the maximum value. With policy gradients, and other direct policy searches, the goal is to learn a map from state to action, which can be stochastic, and works in continuous action spaces.

As a result, policy gradient methods can solve problems that value-based methods cannot:

Large and continuous action space. However, with value-based methods, this can still be approximated with discretisation - and this is not a bad choice, since the mapping function in policy gradient has to be some kind of approximator in practice.

Stochastic policies. A value-based method cannot solve an environment where the optimal policy is stochastic requiring specific probabilities, such as Scissor/Paper/Stone. That is because there are no trainable parameters in Q-learning that control probabilities of action, the problem formulation in TD learning assumes that a deterministic agent can be optimal.