Reinforcement Learning
Exercise 6

October 18, 2021

In this exercise we will implement a reinforcement learning algorithm based on the actor-critic approach.

1 Actor critic

1.1 Actor critic with episodic updates

The simplest step forward from the REINFORCE implementation from Exercise 5 is to learn the value function and replace the discounted rewards with advantages.

Task 1 — 20 points Revisit the policy gradient solution for the continuous Cartpole from Exercise 5 with learned sigma and implement the actor-critic algorithm. In the initial setup, perform TD(0) updates at the end of each episode.

Hint: Check out the PyTorch tutorial to see how to calculate the $A_\theta \nabla_\theta \log \pi_\theta(a_i|s_i)$ term using the `detach()` function.

Question 1 — 10 points What is the relationship between actor-critic and REINFORCE with baseline?

Question 2 — 5 points How can the value of advantage be intuitively interpreted?

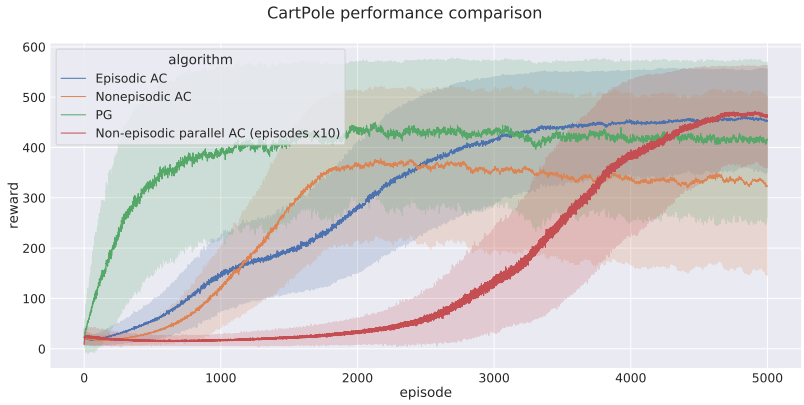

Figure 1: Training performance for different algorithms

1.2 Actor critic with non-episodic updates

Notice that, this time, we rely on TD(0) advantage estimates, rather than calculating Monte Carlo returns. This means that we can apply this method also to non-episodic scenarios. In this case, we can perform policy updates every some number of timesteps, rather than at the end of each episode.

Task 2 — 10 points Update the actor-critic code to perform TD(0) updates every 50 timesteps, **instead of** updating your network at the end of each episode. Make sure to handle the end of each episode correctly (as in the previous exercises, the value of the terminal state is 0).

1.3 Actor critic with parallel data collection

So far, we haven't been using the available computation resources particularly efficiently; we were only using a single CPU core to collect data from the environment. Modern CPUs, however, have multiple cores, which facilitates performing data collection in a parallel way. With the non-episodic setup you've just implemented, it's easy to synchronize data collection across multiple parallel processes. This formulation should result in more stable training, as data coming from separate environments will be less correlated than consecutive samples from one environment.

Task 3 — 5 points Update your code to use parallel data collection. Start up with the `parallel_cartpole.py` script. This code makes use of the parallel wrapper for the environment, which can be found in `parallel_env.py`. This code instantiates `processes` worker processes, with `envs_per_process` threads each. After adapting your code, run it with at least 20 parallel environments and report the results. You can use Aalto computational resources (Maari, Paniikki) if your computer runs sluggish.

Question 3 — 10 points 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.

Question 4 — 10 points 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.

Question 5.1 — 10 points How do actor-critic methods compare to REINFORCE in terms of bias and variance of the policy gradient estimation? Explain your answer.

Question 5.2 — 10 points How could the bias-variance tradeoff in actor-critic be controlled?

Question 6 — 10 points What are the advantages of policy gradient and actor-critic methods compared to action-value methods such as Q-learning? Explain your answer.

2 Submission

The deadline to submit the solutions through MyCourses is on Monday, 08.11 at 12:00. Example solutions will be presented during exercise sessions the same week (08–10.11).

1. **Answers to all questions** posed in the text.
2. The **training performance plots** for each of the tasks (Tasks 1, 2 and 3).

In addition to the report, you must submit as separate files, in the same folder as the report:

1. Python code used to solve **all task exercises**.

Please remember that not submitting a PDF report following the **Latex template** provided by us will lead to subtraction of points.

For more formatting guidelines and general tips please refer to the submission instructions file on mycourses.

If you need help or clarification solving the exercises, you are welcome to come to the exercise sessions in weeks 42/43/44 (on Mondays, Tuesdays and Wednesdays).

Good luck!