Reinforcement Learning
Exercise 5

October 3, 2022

In this exercise, we will implement the REINFORCE policy gradient algorithm for InvertedPendulum environment. The InvertedPendulum is similar to the previously used CartPole environment, except it has a continuous action space: the action can now have values in a range of $[-3, 3]$. The action value represents a numerical force applied to the cart, with magnitude representing the amount of force and sign representing the direction.

Policy Gradient

Policy gradient with and without baseline

Task 1 — 25 points Implement the REINFORCE policy gradient algorithm to balance the InvertedPendulum. Use `agent.py` as a starting point and complete the unfinished implementation (marked with `TODO`s). Also finish the `train()` function in `train.py`, similarly to how it was done in Exercise 1.

Use constant standard deviation $\sigma = 1$ (i.e. $\log(\sigma) = 0$) for the output action distribution throughout the training. Implement

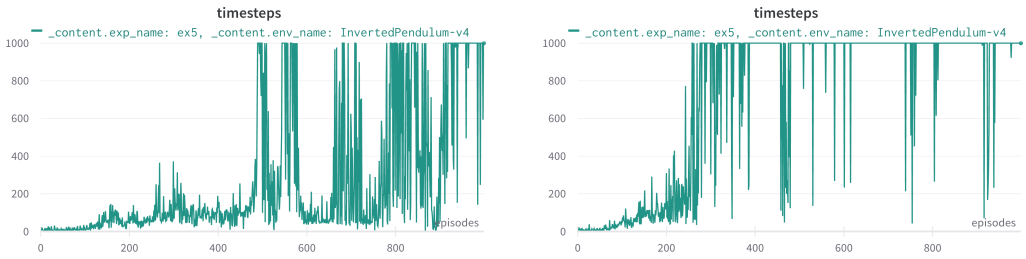
(a) basic REINFORCE without baseline **(15 points)**,

(b) REINFORCE with a constant baseline $b = 20$ **(5 points)**,

(c) REINFORCE with discounted rewards normalized to zero mean and unit variance **(5 points)**,

Attach the training performance plots for each case in your report.

Hint: The `agent.py` file contains a basic neural network structure. We include reasonable hyperparameters in the `cfg` folder.



(a) REINFORCE without baseline.

(c) REINFORCE with discounted rewards normalized to zero mean and unit variance.

Figure 1: The training performance plots for Task 1 (a) and (c) might look like these graphs.

Hint: Your policy should output a probability distribution over actions. A good (and easy) choice would be to use a normal distribution (`from torch.distributions import Normal`). Log-probabilities can be calculated using the `log_prob` function of the distribution. We strongly recommend you to read the official PyTorch documentation to learn how to use the distributions and related functions.

Question 1.1 — 15 points How would you choose a good value for the baseline? Why is the training more stable when using a baseline? **Justify your answer.**

Choosing the value of variance

Task 2 — 10 points Implement the policy's variance as a learnable parameter of the network and update it during training. Set the initial value σ_0^2 to 1. Use REINFORCE with normalized discounted returns for this task. **Attach the training performance plot in your report.**

Hint: To make your learned variance automatically updated by the optimizer, declare your variable inside the `__init__` function of the model using `torch.nn.Parameter(some_tensor)`.

Question 2.1 — 5 points What are the strong and weak sides of using either a) constant variance during training, or b) learning the variance during training? **Please explain.**

Question 2.2 — 5 points In case of learned variance, what's the impact of initialization on the training performance? **Please explain**.

PG and experience replay

Question 3 — 15 points Why the method implemented in this exercise could not be **directly** used with experience replay? Which steps of the algorithm would be problematic to perform with experience replay? How the problematic steps could be resolved? **Explain your answer.**

Real-world control problems

Question 4.1 — 5 points What could go wrong when a model with an **unbounded** continuous action space and a reward function like the one used here (+1 for survival) were to be used with a physical system?

Question 4.2 — 10 points How could the problems appearing in Question 4.1 be mitigated without putting a hard limit on the actions? **Explain your answer.**

Discrete action spaces

Question 5 — 10 points Can policy gradient methods be used with discrete action spaces? Why/why not? Which steps of the algorithm would be problematic to perform, if any? **Explain your answer.**

Submission

The deadline to submit the solutions through MyCourses is on Monday, 24.10 at 23:55.
Your submission should consist of

1. **Answers to all questions** asked in the text.
2. The **training performance plots** for each of the tasks (Task 1 a, b, and c, and Task 2).

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 join the exercise sessions.

Good luck!