6.884 Spring 2020 Computational Sensorimotor Learning HW 1

DUE DATE: 03/10/2020 (by 2.30pm ET)

1 Problem 1

We have discussed in lectures about Proximal Policy Optimization (PPO). To further your understanding of PPO, in this assignment we want to you to implement the algorithm in code. However, you are free to take inspiration and use large chunks of code from existing implementation **as long as you write the policy gradient and clipping** parts on your own. In case you are using code from other repositories, do cite the source.

We will test your implementation of PPO by training a policy to reach a desired goal location (i.e. within 0.075m of the goal location). We will use a simulated UR5e arm for this purpose. In this environment, we will use low-dimensional state representation composes of the location of the tip of the arm. The robot's action space is change in position in the x-y plane.

In addition to the policy gradient algorithm, training a RL agent, requires a reward function. You need to design a reward function that can suitably solve the task. If you end up experimenting with the reward function, we would love to see how the choice of reward function changes either the convergence properties or the speed of learning.

Deliverables:

- Source code with a working implementation of PPO (30 pts).
- A mathematical description of the reward function (15 pts).
- Training plot showing rewards as a function of time. Report the average performance over 3 random seeds (15 pts).
- Video showing evaluation of policy (5 pts).

2 Problem 2

In reinforcement learning, we often assume that reward is provided by an oracle. However, it's not always straightforward to design rewards that result in the desired behaviors. To better understand this issue, we will train the UR5e robot to reach around the wall. For this part of the assignment, we will use the environment `reacher_wall`, which is described in the file `reacher_wall.py`.

Complete the following steps:

1. Learn a reaching policy for `reacher_wall` environment using the reward function you designed in Problem 1. Does the agent succeed? If not, why does it fail? Do you have any suggestions on how to overcome these challenges?
2. Design a reward function that outperforms the agent trained in part 1 in terms of the success rate. What are the pros/cons of this reward function in comparison to the one used in Part 1?

Deliverables:

- Answer to part 1 (10 pts).
- Training plot showing reward as a function of time for part 1. Report the average performance over 3 random seeds (15 pts).
- Video showing evaluation of policy for part 1 (5 pts).
- Answer to part 2 which includes a mathematical description of the reward function (15 pts).
- Training plot showing rewards as a function of time for part 2. Report the average performance over 3 random seeds (15 pts).
- Video showing evaluation of policy for part 2 (5 pts).

3 Problem 3

Now, let's consider a more interesting task of pushing a cylinder to a goal location. You are provided with a `pusher` environment in the file `pusher.py`. You need to implement the reward function for the `pusher` environment that can suitably solve the task of pushing the cylinder to the goal location (i.e. within 0.05m of the goal location).

Deliverables:

- A mathematical description of the reward function (15 pts).
- Training plot showing rewards as a function of time. Report the average performance over 3 random seeds (15 pts).
- Video showing evaluation of policy (5 pts).

4 Submission Instructions

The deliverables mentioned above should be submitted on gradescope. Please also submit your code (along with the instructions on how to run it) in a zip file on gradescope.