CS 277 (W22): Control and Reinforcement Learning
Assignment 1

Due date: Tuesday, January 18, 2022 (Pacific Time)

Roy Fox
https://royf.org/crs/W22/CS277

In the following questions, a formal proof is not needed (unless specified otherwise). Instead, briefly explain informally the reasoning behind your answer.

Part 1 Relations between horizon settings (25 points)

Definition (Stationary distribution)**.** *A* stationary distribution $\bar{p}$ *is any state distribution such that, if the distribution of the state* s_t *at any time* t *is* $\bar{p}$*, then the distribution of* s_{t+1} *at time* $t+1$ *is also* $\bar{p}$*.*

Question 1 (7 points) Assume that, in a particular MDP and with a particular agent policy π, the initial distribution $\bar{p}(s_0)$ is a stationary distribution for that process. Write down an expression for the expected T-step finite-horizon return that only involves T, $\bar{p}$, π, and the reward function $r(s,a)$.

Question 2 (6 points) Write down an expression for the discounted horizon with discount γ that only involves γ, $\bar{p}$, π, and r.

Question 3 (5 points) For a given discount γ, what is the "effective finite horizon" of the discounted horizon with discount γ, i.e. the finite horizon T that would gives the same expression in both previous questions?

Question 4 (7 points) The purpose of defining the return R is to summarize a sequence of rewards into one real number that we can maximize. Suppose that instead we select just a single one of the rewards, $r_t = r(s_t, a_t)$, by drawing the variable t geometrically at random with parameter $1-\gamma$. Show that $\mathbb{E}[r_t] = c\,\mathbb{E}[R_\gamma]$, where R_γ is the discounted return with discount γ, and c doesn't depend on p, π, or r. In this question do not assume a stationary distribution.

Part 2 Value function (25 points)

Question 5 (8 points) Show by induction that, for any $t \geq t_0 \geq 0$, the conditional distribution $p(s_t|s_{t_0})$ of s_t given s_{t_0} depends only on $t - t_0$ (and not on t_0).

Question 6 (8 points) In the discounted horizon, the future return from time $t_0 \geq 0$ is defined as

$$R_{\geq t_0} = \sum_{t \geq t_0} \gamma^{t-t_0} r(s_t, a_t).$$

For any time step $t_0 \geq 0$, we define the value function to be the expected future return from time t_0, given the state s_{t_0} at that time:

$$V^{\pi}_{t_0}(s) = \mathbb{E}_{\xi \sim p_\pi}[R_{\geq t_0} | s_{t_0} = s]. \tag{1}$$

Using the claim in Question 5, show that this value function is time-invariant, i.e. it doesn't depend on t_0 after all.

Question 7 (9 points) In the T-step finite horizon, the future return from time $t_0 \geq 0$ is defined as

$$R_{\geq t_0} = \sum_{t=t_0}^{T-1} r(s_t, a_t).$$

With $V^{\pi}_{t_0}$ defined as in (1), show a very simple MDP and policy, such that in the 2-step finite horizon $V^{\pi}_0 \neq V^{\pi}_1$. In other words, give a counterexample that shows that what Question 6 claims for the discounted horizon doesn't hold in the finite horizon.

Part 3 Behavior Cloning (50 points)

In this part, you will install a Deep RL framework, RLlib, and use it to evaluate the Behavior Cloning algorithm. The OS we use in this assignment is Linux / MacOS. If you're using Windows or another OS, please use a local or remote virtual machine.

Installation

Make sure you have a recent version of Python installed, such as the latest Python 3.8; but **not Python 3.9**, as RLlib version we'll be using (1.1.0) doesn't seem to support it. It is recommended that you create a new conda environment for this assignment.

RLlib can use either of two Deep Learning libraries, TensorFlow (TF) and PyTorch. Most of the algorithms already implemented in RLlib support both these libraries. In this assignment we will use TF for CPU:

```
pip install tensorflow
```

It is also possible to install TF for GPU, and this would allow much faster execution, but is not needed in this assignment.

We will also need a simulator of the environment, and the one we will use is Box2D:

```
pip install box2d-py
```

This may require you to first install swig.

RLlib is implemented on top of the Ray distributed execution library:

```
pip install "ray[rllib]==1.1.0"
```

1. Use RLlib to train an agent to play Lunar Lander. The algorithm we will use is called PPO (we'll learn about it in a later lecture). The environment is implemented by OpenAI Gym, and is called LunarLanderContinuous-v2 (in this version, the action space is continuous). Run the algorithm for 1000 "iterations" (what iteration means in RLlib is algorithm-dependent) and create a checkpoint (a save of the agent's trained parameters) every 10 iterations.

```
rllib train
  --run PPO
  --env LunarLanderContinuous-v2
  --checkpoint-freq 10
  --stop '{"training_iteration": 1000}'
```

2. Roll out the agent that you trained to see how well it performed. The agent checkpoints are by default in a path named

   `~/ray_results/default/PPO_LunarLanderContinuous-v2_<experiment_id>`,

   where `experiment_id` is an automatically assigned identifier of the experiment you ran in the previous step, ending with the date and time. Roll out for 5000 steps, which should be about 20 episodes.

```
rllib rollout ~/ray_results/default/→
    PPO_LunarLanderContinuous-v2_<experiment_id>/→
    checkpoint_1000/checkpoint-1000
  --run PPO
  --env LunarLanderContinuous-v2
  --steps 5000
```

   You can also roll out an earlier checkpoint to see the difference in performance. RLlib will output the return of each episode, which should be well above 200 for successful episodes. If you are not seeing consistently good episodes (most should get above 200 return), rerun the training in the previous question.

3. Use the trained agent to generate demonstrations for an imitation learning agent. Roll out for 250000 steps. The demonstrations will be saved in a file named `rollouts.pkl`. The `--no-render` flag prevents rendering the episode to screen, thus saving much time.

```
rllib rollout ~/ray_results/default/→
    PPO_LunarLanderContinuous-v2_<experiment_id>/→
    checkpoint_1000/checkpoint-1000
  --run PPO
  --env LunarLanderContinuous-v2
  --steps 250000
  --out rollouts.pkl
  --no-render
```

4. Convert the demonstrations into the format used by RLlib for training from offline data. Download the utility https://royf.org/crs/W22/CS277/A1/prepare_dataset.py and run it. The results will be saved in a directory named `LunarLanderContinuous-v2`.

5. Train a Behavior Cloning agent. The agent will be trained on input from the data generated in the previous step, and evaluated in new simulations of the environment.

```
rllib train
  --run BC
  --env LunarLanderContinuous-v2
  --checkpoint-freq 1000
  --stop '{"training_iteration": 10000}'
  --config='{"input": "LunarLanderContinuous-v2", →
    "input_evaluation": ["simulation"]}'
```

6. Repeat step 2 for the trained BC agent, and report the mean and standard deviation of the returns of 5 episodes.