CMPE 58Y - Robot Learning
Homework 1: Tabular Q-Learning

TA: G. Tuba Ç. Girgin
tubacibuk1@gmail.com

March 15, 2022

1 Introduction

In this homework, you will implement Q-learning for the cart pole task [1] using OpenAI Gym. `gym` module provides an abstraction for Markov decision processes, enabling the rapid development and test of reinforcement learning algorithms. You can find the installation instructions at https://github.com/openai/gym. Here is a short example:

```
import gym

env = gym.make("CartPole-v1")
env.reset()
for _ in range(1000):
    # you don't have to render. it's just for visualization.
    env.render()
    # take a random action
    observation, reward, done, info = env.step(env.action_space.sample())
env.close()
```

In this script, we take random actions and the environment returns `observation`, `reward`, `done`, and `info`.

- `observation`: next state of the environment.
- `reward`: the reward which the agent receives after executing the action (here, it is a random one) at the current environment state (previous iteration's observation value).
- `done`: a boolean indicating whether the trajectory has arrived at a terminal state or not.
- `info`: other debug stuff, depends on the environment.

The convergence of the algorithm will depend on your hyperparameter settings. These hyperparameters are:

- The quantization method of the continuous state values. If you make a very fine-grained discrete state representation, you will have lots of values to fill in the table. If you make it very coarse-grained, then the Q table might not have enough complexity to solve the task.
- Learning rate (step size), α. Too much, you diverge. Too small, you will need a lot of steps (and possibly a lot of exploration). You can choose a value between 0.01 and 0.5.
- Discounting factor, γ. 0: no horizon, will learn faster, but short-sighted. 1: full horizon, will learn slower, though the agent can see long horizons. You can choose a value between 0.99999 and 0.9.
- ϵ factor for exploration. 0: full exploitation, 1: full exploration.

- ϵ decay rate. We do not want the exploration to end very fast, but at some point, we have to let go of exploration. You can choose a value between 0.99999 and 0.9.

You can use the following method and attributes to understand your environment:

```
env.observation_space.sample()
env.observation_space.low
env.observation_space.high
```

Plot the reward over episodes and submit your code (a jupyter notebook is also fine) to `tubacibuk1@gmail.com`. Feel free to ask any questions. Cheating will be penalized by -200 points.

Deadline: 22.03.2022 23:59 GMT+3
Extended deadline: 29.03.2022 23:59 GMT+3 (-20 points)

References

[1] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983.