RUTGERS UNIVERSITY
School of Engineering
Department of Electrical & Computer Engineering

ECE 472 – Robotics & Computer Vision– Fall 2022

Project 2 - Reinforcement Learning

Name (last, first) : Mehmet Ali Soner

netID : mas996

RUID: 196000499

Date: November 22, 2022

1 Problem 1

Cart Pole code assignment, code + explanation

Important: For google colab and displaying videos, use code below:

```
# install dependencies needed for recording videos
!apt-get install -y xvfb x11-utils
!pip install pyvirtualdisplay==0.2.*


from pyvirtualdisplay import Display
display = Display(visible=False, size=(1400, 900))
_ = display.start()

from gym.wrappers.monitoring.video_recorder import VideoRecorder
before_training = "before_training.mp4"

video = VideoRecorder(env, before_training)
# returns an initial observation
env.reset()
for i in range(200):
  env.render()
  video.capture_frame()
  # env.action_space.sample() produces either 0 (left) or 1 (right).
  observation, reward, done, info = env.step(env.action_space.sample())
  # Not printing this time
  #print("step", i, observation, reward, done, info)

video.close()
env.close()


from base64 import b64encode
def render_mp4(videopath: str) -> str:
  """
  Gets a string containing a b4-encoded version of the MP4 video
  at the specified path.
  """
  mp4 = open(videopath, 'rb').read()
  base64_encoded_mp4 = b64encode(mp4).decode()
  return f'<video width=400 controls><source src="data:video/mp4;' \
         f'base64,{base64_encoded_mp4}" type="video/mp4"></video>'


from IPython.display import HTML
html = render_mp4(before_training)
HTML(html)
```

2 Problem 2

Explain DQN algorithm in paragraphs, include definitions of state, action,environment, reward.

Notes: Agents and environment: agent(s) interacts with environment, which can be a simulation, such as the cart pole example. Each step, the agent observes the environment (state of environment) and takes action and receives a reward based on that. Agents learn from repeated trials; these are called episodes. RL framework trains a policy for the agent to follow. Policy shows which actions to take one after the other in order to maximize reward.

In RL, want to train the agent to make better decision or act better with each episode. Policy == neural network.

```
# returns an initial observation
env.reset()
```

```
for i in range(20):

  # env.action_space.sample() produces either 0 (left) or 1 (right).
  observation, reward, done, info = env.step(env.action_space.sample())

  print("step", i, observation, reward, done, info)

env.close()
```

step: how many times through environment. observation = [x cart, y cart, pole angle, pole ang. velocity]. reward = 1 for each step. done = when episode is done (maybe pole tipped over too much)

3 Problem 3

Performance metrics and plots

4 Problem 4

Three other problems for RL; state, action, environment and reward for each