Navigation_Project_Solution

November 24, 2023

1 Navigation

In this notebook, you will learn how to use the Unity ML-Agents environment for the first project of the Deep Reinforcement Learning Nanodegree.

1.1 Import Libraries

In [1]:
```
# Step 1. Update the PATH env var.
import os
os.environ['PATH'] = f"{os.environ['PATH']}:/root/.local/bin"
os.environ['PATH'] = f"{os.environ['PATH']}:/opt/conda/lib/python3.6/site-packages"
# Step 2. Restart the Kernel.
# If you skip this step, your notebook may not be able to import the packages well.
```

In [1]:
```
# Install the pinned version of packages, similar to below or use requirements.txt

!python -m pip install 'numpy==1.19.5' 'prompt-toolkit<2.0.0,>=1.0.15' 'jupyter-client>=

# Check the version of any specific package
!python -m pip freeze | grep numpy

!pip -q install ./python
```

```
Collecting numpy==1.19.5
  Downloading https://files.pythonhosted.org/packages/45/b2/6c7545bb7a38754d63048c7696804a0d9473
    100% || 13.4MB 1.2MB/s eta 0:00:01
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.15 in /opt/conda/lib/python3.6/site-pa
Collecting jupyter-client>=7.0.0
  Downloading https://files.pythonhosted.org/packages/56/a7/f4d3790ce7bb925d3ffe299244501a264f23
    100% || 133kB 21.9MB/s ta 0:00:01
Collecting jsonschema>=3.0.1
  Downloading https://files.pythonhosted.org/packages/e0/d9/05587ac378b9fd2c352c6f024f1324016836
    100% || 71kB 18.0MB/s ta 0:00:01
Collecting widgetsnbextension==3.0.0
  Downloading https://files.pythonhosted.org/packages/f3/7b/8591debe2bb7907a70d2aecee05baac918ab
```

```
    100% || 2.5MB 10.2MB/s ta 0:00:01
Requirement already satisfied: six>=1.9.0 in /opt/conda/lib/python3.6/site-packages (from prompt
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.6/site-packages (from prompt-to
Collecting jupyter-core>=4.6.0 (from jupyter-client>=7.0.0)
  Downloading https://files.pythonhosted.org/packages/60/7d/bee50351fe3ff6979e949b9c4c00c556a7a9
    100% || 92kB 19.2MB/s ta 0:00:01
Collecting nest-asyncio>=1.5 (from jupyter-client>=7.0.0)
  Downloading https://files.pythonhosted.org/packages/ab/d3/48c01d1944e0ee49fdc005bf518a68b0582d
Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/lib/python3.6/site-packages (f
Requirement already satisfied: pyzmq>=13 in /opt/conda/lib/python3.6/site-packages (from jupyter
Requirement already satisfied: tornado>=4.1 in /opt/conda/lib/python3.6/site-packages (from jupy
Requirement already satisfied: entrypoints in /opt/conda/lib/python3.6/site-packages (from jupyt
Requirement already satisfied: traitlets in /opt/conda/lib/python3.6/site-packages (from jupyter
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /opt/conda/lib/p
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.6/site-packages (from jso
Collecting importlib-metadata; python_version < "3.8" (from jsonschema>=3.0.1)
  Downloading https://files.pythonhosted.org/packages/a0/a1/b153a0a4caf7a7e3f15c2cd56c7702e2cf3d
Requirement already satisfied: notebook>=4.4.1 in /opt/conda/lib/python3.6/site-packages (from w
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: decorator in /opt/conda/lib/python3.6/site-packages (from traitle
Collecting zipp>=0.5 (from importlib-metadata; python_version < "3.8"->jsonschema>=3.0.1)
  Downloading https://files.pythonhosted.org/packages/bd/df/d4a4974a3e3957fd1c1fa3082366d7fff6e4
Collecting typing-extensions>=3.6.4; python_version < "3.8" (from importlib-metadata; python_ver
  Downloading https://files.pythonhosted.org/packages/45/6b/44f7f8f1e110027cf88956b59f2fad776cca
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.6/site-packages (from notebook>=
Requirement already satisfied: nbformat in /opt/conda/lib/python3.6/site-packages (from notebook
Requirement already satisfied: nbconvert in /opt/conda/lib/python3.6/site-packages (from noteboo
Requirement already satisfied: ipykernel in /opt/conda/lib/python3.6/site-packages (from noteboo
Requirement already satisfied: Send2Trash in /opt/conda/lib/python3.6/site-packages (from notebo
Requirement already satisfied: terminado>=0.8.1 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: prometheus_client in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: mistune>=0.8.1 in /opt/conda/lib/python3.6/site-packages (from nb
Requirement already satisfied: pygments in /opt/conda/lib/python3.6/site-packages (from nbconver
Requirement already satisfied: bleach in /opt/conda/lib/python3.6/site-packages (from nbconvert-
Requirement already satisfied: pandocfilters>=1.4.1 in /opt/conda/lib/python3.6/site-packages (f
Requirement already satisfied: testpath in /opt/conda/lib/python3.6/site-packages (from nbconver
Requirement already satisfied: defusedxml in /opt/conda/lib/python3.6/site-packages (from nbconv
Requirement already satisfied: ipython>=4.0.0 in /opt/conda/lib/python3.6/site-packages (from ip
Requirement already satisfied: html5lib!=0.9999,!=0.99999,<0.99999999,>=0.999 in /opt/conda/lib/
Requirement already satisfied: backcall in /opt/conda/lib/python3.6/site-packages (from ipython>
Requirement already satisfied: simplegeneric>0.8 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.6/site-packages (from ipyth
Requirement already satisfied: pexpect; sys_platform != "win32" in /opt/conda/lib/python3.6/site
Requirement already satisfied: jedi>=0.10 in /opt/conda/lib/python3.6/site-packages (from ipytho
Requirement already satisfied: setuptools>=18.5 in /opt/conda/lib/python3.6/site-packages (from
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.6/site-packages (from p
Installing collected packages: numpy, jupyter-core, nest-asyncio, jupyter-client, zipp, typing-e
```

```
  Found existing installation: numpy 1.12.1
    Uninstalling numpy-1.12.1:
      Successfully uninstalled numpy-1.12.1
  Found existing installation: jupyter-core 4.4.0
    Uninstalling jupyter-core-4.4.0:
      Successfully uninstalled jupyter-core-4.4.0
  Found existing installation: jupyter-client 5.2.4
    Uninstalling jupyter-client-5.2.4:
      Successfully uninstalled jupyter-client-5.2.4
  Found existing installation: jsonschema 2.6.0
    Uninstalling jsonschema-2.6.0:
      Successfully uninstalled jsonschema-2.6.0
  Found existing installation: widgetsnbextension 3.1.0
    Uninstalling widgetsnbextension-3.1.0:
      Successfully uninstalled widgetsnbextension-3.1.0
Successfully installed importlib-metadata-4.8.3 jsonschema-4.0.0 jupyter-client-7.1.2 jupyter-co
numpy==1.19.5
ipython 6.5.0 has requirement prompt-toolkit<2.0.0,>=1.0.15, but you'll have prompt-toolkit 3.0.
```

```
In [2]: from unityagents import UnityEnvironment
        import numpy as np
        import torch
        import matplotlib.pyplot as plt
        from dqn_agent import Agent
        from collections import deque
```

1.1.1 1. Start the Environment

```
In [3]: #env = UnityEnvironment(file_name="/mnt/store_data/Value-based-methods/p1_navigation/Ban
        env = UnityEnvironment(file_name="/data/Banana_Linux_NoVis/Banana.x86_64")
```

```
INFO:unityagents:
'Academy' started successfully!
Unity Academy name: Academy
        Number of Brains: 1
        Number of External Brains : 1
        Lesson number : 0
        Reset Parameters :

Unity brain name: BananaBrain
        Number of Visual Observations (per agent): 0
        Vector Observation space type: continuous
        Vector Observation space size (per agent): 37
        Number of stacked Vector Observation: 1
        Vector Action space type: discrete
        Vector Action space size (per agent): 4
        Vector Action descriptions: , , ,
```

Environments contain ***brains*** which are responsible for deciding the actions of their associated agents. Here we check for the first brain available, and set it as the default brain we will be controlling from Python.

```
In [4]: # get the default brain
        brain_name = env.brain_names[0]
        brain = env.brains[brain_name]
```

1.1.2 2. Examine the State and Action Spaces

The simulation contains a single agent that navigates a large environment. At each time step, it has four actions at its disposal: - `0` - walk forward - `1` - walk backward - `2` - turn left - `3` - turn right

The state space has `37` dimensions and contains the agent's velocity, along with ray-based perception of objects around agent's forward direction. A reward of `+1` is provided for collecting a yellow banana, and a reward of `-1` is provided for collecting a blue banana.

Run the code cell below to print some information about the environment.

```
In [5]: # reset the environment
        env_info = env.reset(train_mode=True)[brain_name]

        # number of agents in the environment
        print('Number of agents:', len(env_info.agents))

        # number of actions
        action_size = brain.vector_action_space_size
        print('Number of actions:', action_size)

        # examine the state space
        state = env_info.vector_observations[0]
        print('States look like:', state)
        state_size = len(state)
        print('States have length:', state_size)
```

```
Number of agents: 1
Number of actions: 4
States look like: [1.         0.         0.         0.         0.84408134 0.
 0.         1.         0.         0.0748472  0.         1.
 0.         0.         0.25755    1.         0.         0.
 0.         0.74177343 0.         1.         0.         0.
 0.25854847 0.         0.         1.         0.         0.09355672
 0.         1.         0.         0.         0.31969345 0.
 0.        ]
States have length: 37
```

1.1.3 3. Train the Agent

```
In [6]: # Initilize the agent network with the suitable parameters.
        seed=42
        agent = Agent(state_size=state_size, action_size=action_size, seed=seed)
```

```
In [7]: def dqn(agent, env, brain_name, n_episodes=2000, max_t=1000, eps_start=1.0, eps_end=0.01
            """Deep Q-Learning.

            Trains a Deep Q-Network (DQN) to solve a given environment using the epsilon-greedy

            Params:
            - agent: The DQN agent responsible for interacting with the environment.
            - env: The Unity environment.
            - brain_name (str): The name of the brain associated with the environment.
            - n_episodes (int): maximum number of training episodes.
            - max_t (int): maximum number of timesteps per episode.
            - eps_start (float): starting value of epsilon for epsilon-greedy action selection.
            - eps_end (float): minimum value of epsilon.
            - eps_decay (float): multiplicative factor (per episode) for decreasing epsilon.

            Returns:
            - scores (list): list containing scores from each episode.
            """

            scores = []                        # list containing scores from each episode
            scores_window = deque(maxlen=100)  # last 100 scores
            epsilon = eps_start                    # initialize epsilon

            for i_episode in range(1, n_episodes + 1):

                # Modified for the environment, as shown in the example
                env_info = env.reset(train_mode=True)[brain_name]  # reset the environment
                state = env_info.vector_observations[0]            # get the current state
                score = 0                                          # initialize the score

                for _ in range(max_t):
                    action = agent.act(state, epsilon)                 # select an action
                    env_info = env.step(action)[brain_name]        # send the action to the envir

                    next_state = env_info.vector_observations[0]   # get the next state
                    reward = env_info.rewards[0]                    # get the reward
                    done = env_info.local_done[0]                  # see if the episode has finis

                    agent.step(state, action, reward, next_state, done)
                    state = next_state
                    score += reward

                    if done:
                        break

                scores_window.append(score)       # save the most recent score
                scores.append(score)              # save the most recent score
                epsilon = max(eps_end, eps_decay * epsilon)  # decrease epsilon
```

```
            print('\rEpisode {}\tAverage Score: {:.2f}'.format(i_episode, np.mean(scores_win
            if i_episode % 100 == 0:
                print('\rEpisode {}\tAverage Score: {:.2f}'.format(i_episode, np.mean(scores

            if np.mean(scores_window) >= 13.0:
                print('\nEnvironment solved in {:d} episodes!\tAverage Score: {:.2f}'.format
                torch.save(agent.qnetwork_local.state_dict(), 'checkpoint.pth')
                break

        # Close the environment after finishing the training.
        env.close()
        print("Scores: {}".format(score))

        return scores
```

In [8]:
```
# Example of using the DQN function to train an agent on the given environment
# and retrieve the scores achieved during training.

# The 'agent', 'env', and 'brain_name' are previously defined.

# Train the agent using the DQN function with specified parameters.
scores = dqn(agent, env, brain_name, n_episodes=2000, max_t=1000, eps_start=1.0, eps_end
```

```
Episode 100	Average Score: 1.01
Episode 200	Average Score: 4.26
Episode 300	Average Score: 7.58
Episode 400	Average Score: 10.88
Episode 477	Average Score: 13.01
Environment solved in 377 episodes!	Average Score: 13.01
Scores: 17.0
```

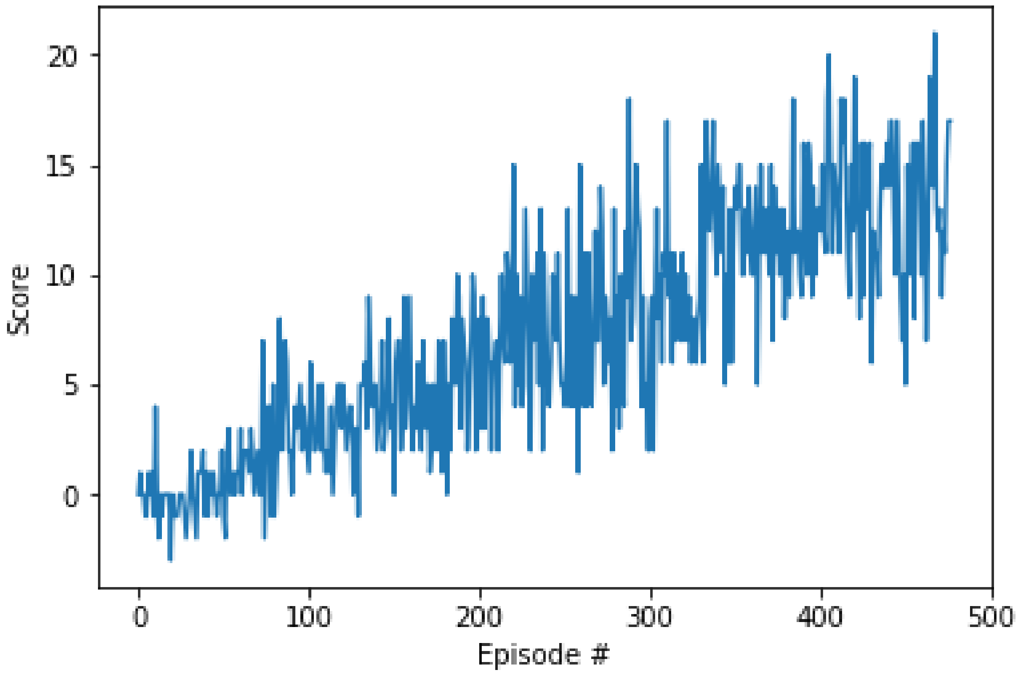
In [9]:
```
# Create a figure and axis for plotting
fig = plt.figure()
ax = fig.add_subplot(111)

# Plot the scores over episodes
plt.plot(np.arange(len(scores)), scores)

# Set labels for the axes
plt.ylabel('Score')
plt.xlabel('Episode #')

# Save the plot as an image file
plt.savefig('dqn_scores.png', bbox_inches='tight')

# Show the plot
plt.show()
```



In [ ]: