CS529– Applied Artificial Intelligence

Instructor : Shashi Shekhar Jha (shashi@iitrpr.ac.in)

Lab Assignment - 3

Due on 22/11/2019 2400 Hrs

Total: 100 Points

Submissions Instructions:

The submission is through google classroom in a single zip file. In case you face any trouble with the submission, you can comment in the google classroom with your queries.

Your submission must be your original work. **Do not indulge in any kind of plagiarism or copying.** Abide by the honour and integrity code to do your assignment.

Any late submissions after the due date will attract penalties.

You submission must include:

- A legible PDF document with all your answers to the assignment problems/ questions.
- A folder named as 'code' containing the scripts for the assignment along with the other necessary files to run your code.
- A README.txt file explaining how to execute your code.

Naming Convention:

Name the ZIP file submission as follows:

YourName_rollnumber.zip E.g. if your roll number is 2016csx1234 and Abcd,

then you should name the zip file as: Abcd_2016csx1234.zip

Programming Language

As was the case in earlier Lab. assignments, you will be using **Python** as the chosen programming language. All data wrangling operations should be done using ***Numpy*** and ***Pandas***.

In case you are new to python, you can use the following resource to get you started on Python https://wiki.python.org/moin/BeginnersGuide/Programmers

Open AI Gym

https://gym.openai.com/docs/

For this assignment, we will be making use of the Open AI Gym. Gym is a toolkit for developing and comparing reinforcement learning algorithms. It provides an array of environments with the specifications of state and action spaces wherein different reinforcement learning algorithms can be tested and benchmarked with.

You can check the complete registry of environments present in Gym using the following command:

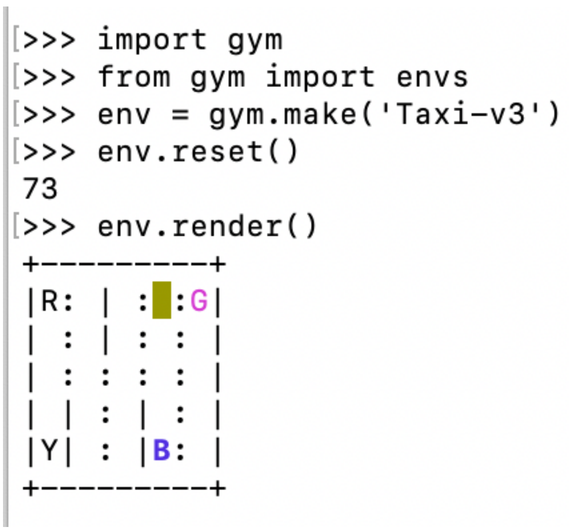
```
from gym import envs
print(envs.registry.all())
```

For the initial implementation of algorithms, we will be focusing on an environment called "Taxi-v3" which is a taxi game environment. The details about taxi environment is as follows:

Representations:

- | → WALL (Can't pass through, will remain in the same position if tries to move through wall)
- Yellow → Taxi Current Location
- Blue → Pick up Location
- Purple → Drop-off Location
- Green → Taxi turn green once passenger board
- Letters → Locations

Here is an example of Taxi environment in gym:



```
[>>> import gym
[>>> from gym import envs
[>>> env = gym.make('Taxi-v3')
[>>> env.reset()
73
[>>> env.render()
+---------+
|R: | :█:G|
| : | : : |
| : : : : |
| | : | : |
|Y| : |B: |
+---------+
```

You can use the following commands for various information regarding the taxi environment:

```
env.observation_space.n   #Total no. of states
env.action_space.n   #Total no. of actions
```

Actions (6 in total)

- 0: move south
- 1: move north
- 2: move east
- 3: move west
- 4: pickup passenger
- 5: dropoff passenger

You can execute an action in the environment using the `step()` function:

```
env.step(3)
```

At each timestep, the agent chooses an action, and the environment returns an observation and a reward.

There are 4 elements returned by the step function:

- **Observation (object)**: the state the environment is in or an environment-specific object representing your observation of the environment.
- **Reward (float)**: Reward achieved by the previous action.

    - +20: Last step when a successfully picked up passenger is dropped off at their desired location
    - -1: for each step in order for the agent to try and find the quickest solution possible
    - -10: every time an incorrectly pick up or drop off happens
- **Done (boolean)**: whether it's time to reset the environment again. Most (but not all) tasks are divided up into well-defined episodes, and done being True indicates the episode has terminated.
- **Info (dict)**: Can be ignored, diagnostic information useful for debugging. Official evaluations of your agent are not allowed to use this for learning.

Implementation:

You need to implement this assignment into the file - "assignment_3.py" available along side this assignment.

Questions:

Q.1. Implement a random policy for the Taxi environment in "assignment_3.py". Run the random policy for 1000 episodes. Report the average number of steps taken to drop-off a passenger using the random policy. [10 points]

Q.2. Implement the DP based policy iteration and value iteration methods in "assignment_3.py". How many iterations does it take for policy and value iterations to converge to optimal policy? Do both the method results in the same optimal policy? Comment on the computational efficiency of the policy and value iterations. Also, show the effect of discount factor (γ) assuming different values on the convergence policy and value iteration. Draw a plot of the distribution of number of steps taken to complete the mission using 1000 episodes of the optimal policy found in value and policy iterations. [30 points]

Q.3. Implement the Q-learning algorithm in "assignment_3.py". Do a parameter study of learning rate (α<1) with different values until convergence? What is your criteria of convergence here? Plot a graph showing the improvement in policy with each episode ? Think about the quantity to put in the y-axis. Hint: check the book. [20 points]

Q.4. Example 10.1 in the book talks about the mountain car task. Implement the mentioned value function approximation based Q-learning algorithm for the mountain car and cart-pole environments of **gym**. Take care to put the exact versions in the env() function. This implementation needs to be done in a separate python script. Regenerate all the graphs in Fig 10.1 (in book) from your implementation. [40 points]

____________________-******__________________******__________________________