Middle East Technical University | Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2019-2020

Homework IV: A New Hope

Due date: 31 December 2019, Tuesday, 11:55

1 Objectives

This assignment aims to assist you to expand your knowledge on Reinforcement Learning in case of Value Iteration and Q-Learning methods.

2 Problem Definition

"A long time ago, in a galaxy far, far away...

It is a period of civil war. Rebel spaceships, striking from a hidden base, have won their first victory against the evil Galactic Empire. During the battle, Rebel spies managed to steal secret plans to the Empire's ultimate weapon, the DEATH STAR, an armored space station with enough power to destroy an entire planet. Pursued by the Empire's sinister agents, Princess Leia races home aboard her starship, custodian of the stolen plans that can save her people and restore freedom to the galaxy..."

As you may already know, this is the opening text from the fourth episode of the famous *Star Wars* movie. Without giving further spoiler, *Princess Leia* needs *R2D2*'s help to get the message to *Obi-Wan Kenobi* and as being an agent, *R2D2* is a close friend of us, so we would like to help it in its divine struggle.

Figure 1: *Princess Leia* and our friend, *R2D2*

In this assignment, you will help *R2D2* by using the *force* of Reinforcement Learning. You will be given a problem domain like below and apply **Value Iteration** or **Q-Learning** on this domain according to given configurations. For such a problem, our agent, *R2D2*, should reach the goal state, *Obi-Wan Kenobi*, avoiding possible pitfalls as *stormtroopers* and from any state in which it is placed.

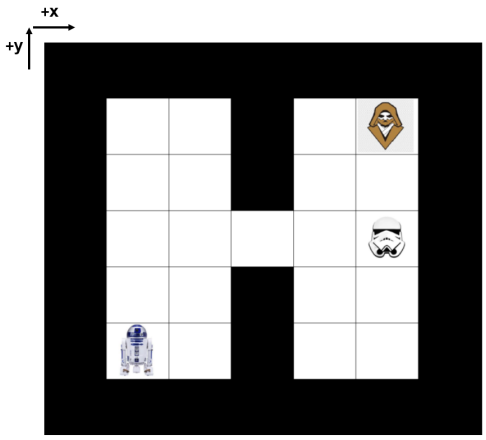

Figure 2: A possible scene from an example problem domain where *R2D2*, a *stormtrooper* and *Obi-Wan Kenobi* are in (1,1), (5,3) and (5,5) coordinates respectively.

3 Specifications

- You will write a python script which will read the problem configuration from a file and output the policy constructed by the requested method to a file.
- In **Value Iteration** you will check the maximum difference between old and updated V values within a given thresold for the termination of process as in the lecture materials.
- For **Q-Learning**;
  - The learning process will be episodic. An episode will end only when the agent reaches the goal state.
  - In each episode, the agent will start from a random state excluding the goal state and the states which are pitfalls and obstacles.
  - You will use ϵ-greedy approach for action selection mechanism. In other words, within a given probability of ϵ, the agent will choose a random action rather than the following its current policy.
- For the given problem,
  - The coordinate of (1,1) will be the left-lower corner of the environment. X-coordinate will increase while going east and Y-coordinate will increase while going north.
  - The domain size is not fixed, the length and width of the environment will be given as inputs.
  - The environment will be static regarding the possible obstacles, pitfalls and goal state. That is, their positions will not change during learning.
  - The agent can move in four directions in the environment: north, east, south and west.
  - If the agent hits a wall around the domain (tries to get out of domain boundaries) or an obstacle, it should stay in the current cell.

- The environment will be deterministic, so any movement of the agent should result in only one way, considering the conditions above.
- The agent may receive four different rewards from the environment: (i) **regular reward** for default step of the agent which don't cause any hit or don't make the agent reach the goal, (ii) **obstacle reward** for hitting a wall or an obstacle, (iii) **pitfall reward** for getting damaged by a pitfall, (iv) **goal reward** for reaching the goal state. They can be **any real number** as long as they are consistent with their meanings.

4 I/O Structure

Your program will read the input from a file whose path will be given as the first command-line argument. The first line of the input file will be the method as "V" for Value Iteration or "Q" for Q-Learning. The rest of file will differ according to requested method. Hence, the structures of these two different input types are given separately.

The structure of input which requires Value Iteration:

```
V
<theta>                        # termination condition factor
<gamma>                        # discount factor
<M> <N>                        # dimensions (M:y-dimension, N:x-dimension)
<k>                            # number of obstacles
<o_1_x> <o_1_y>                # coordinates of obstacle 1
...
<o_k_x> <o_k_y>                # coordinates of obstacle k
<l>                            # number of pitfalls
<p_1_x> <p_1_y>                # coordinates of pitfall 1
...
<p_l_x> <p_l_y>                # coordinates of pitfall l
<g_x> <g_y>                    # coordinates of goal state
<r_d> <r_o> <r_p> <r_g>        # rewards of regular step, hitting an obstacle/wall,
                                 getting damaged by pitfall,reaching the goal
                                    respectively
```

The structure of input which requires Q-Learning:

```
Q
<number of episode>
<alpha>                        # learning rate
<gamma>                        # discount factor
<epsilon>                      # paramater for ϵ-greedy approach
<M> <N>                        # dimensions (M:y-dimension, N:x-dimension)
<k>                            # number of obstacles
<o_1_x> <o_1_y>                # coordinates of obstacle 1
...
<o_k_x> <o_k_y>                # coordinates of obstacle k
<l>                            # number of pitfalls
<p_1_x> <p_1_y>                # coordinates of pitfall 1
...
<p_l_x> <p_l_y>                # coordinates of pitfall l
<g_x> <g_y>                    # coordinates of goal state
<r_d> <r_o> <r_p> <r_g>        # rewards of regular step, hitting an obstacle/wall,
                                 getting damaged by pitfall,reaching the goal
                                    respectively
```

As output, your program will create a file, whose path will be given as the second command-line argument. This file should include the policy represented by as following structure:

```
<s_1_x> <s_1_y> <a_1>      # coordinates of state 1 and chosen action in
                             this state (as 0:'N', 1:'E', 2:'S', 3:'W')
<s_2_x> <s_2_y> <a_2>      # coordinates of state 2 and chosen action
                             in this state
...
<s_i_x> <si_y> <a_i>      # coordinates of state i (i = MxN) and chosen action
                            in this state
```

Important Notes:

- In all files, space character is used as delimiter of multiple entries in a line.
- An example file for each structure can be found in homework file on `OdtuClass`. Make sure your code is compatible with them.

5 Regulations

1. **Programming Language:** You must code your program in Python **2.7**. Your submission will be tested in inek machines. So make sure that it can be executed on them as below.

   ```
   python e1234567_hw4.py input.txt output.txt
   ```

2. **Implementation:** You are allowed to use **sys** and **random** modules from standard python library. You cannot import any other modules.
3. **Late Submission:** No late submission is allowed. No correction can be done in your codes after deadline.
4. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating (any kind of code sharing and codes taken from internet included) will be punished according to the university regulations.
5. **Discussion:** You must follow the `OdtuClass` for discussions and possible updates on a daily basis.
6. **Evaluation:** Your program will be evaluated automatically by simulating the policy you recorded in the output file, so make sure to obey the specifications. As long as your policy reaches an optimal solution for the given testcase, for reasonable amount of trials starting from a random regular state, your code will pass that case. A reasonable timeout will be applied according to the complexity of test cases in order to avoid infinite loops due to an erroneous code.

6 Submission

Submission will be done via `OdtuClass` system. You should upload a **single** python file named in the format <**your-student-id**>_**hw1.py** (i.e. e1234567_hw4.py).