Middle East Technical University Department of Computer Engineering

CENG 462

Artificial Intelligence

Fall '2020-2021

Homework 3

Due date: 01 February 2021, Monday, 23:55

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

"Good news, everyone. We've got a very special delivery today."

That is what *Prof. Farnsworth* from *Futurama* would probably say about your new assignment. In this assignment you will help a famous robot from the same TV series by using the Reinforcement Learning. Our agent in this case will be *Bending Unit-22* a.k.a *Bender Bending Rodriguez.*

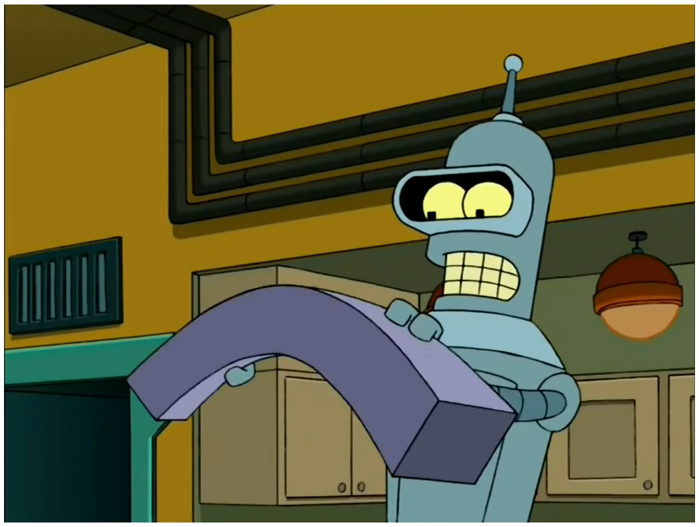

Figure 1: Our famous agent *Bender Bending Rodriguez* by *Matt Groening*

He is actually known for his bad habits and ideas like *"killing all humans"* etc. but he was programmed for bending metals and he is very good at his job. However, even he could fail to bend some special metals and may consume a lot of energy with no use. You will help him to learn which metal he should try to bend, by applying **Value Iteration** or **Q-Learning** for a given problem domain like below. For such a problem, our agent, *Bender*, should reach the goal state by bending the correct metal, avoiding possible pitfalls as *attempts for energy-consuming unbendable metals* and from any state in which he is placed. If you successfully lead him in his challenge, maybe he can add you to his *"Do Not Kill"* list.

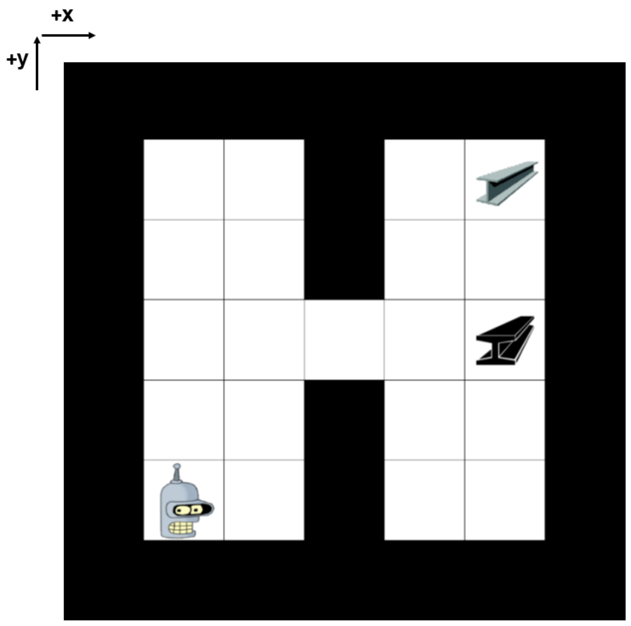

Figure 2: A possible scene from an example problem domain where *Bender*, an *unbendable black metal* and the *bendable bluish metal* are in the coordinates of (1,1), (5,3) and (5,5) respectively.

3 Specifications

- You will write a python3 program which will read the problem configuration from a file and output the policy constructed by the requested method to a file (see next section for the details).
- In **Value Iteration** you will check the maximum difference between old and updated V values within a given threshold 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 locations of the goal, possible 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`. Furthermore, the `bend` action will be available for the locations having metals regardless being bendable or not. Note that this action will **not** be an option for the agent in a regular location.
- A state will be represented with its coordinates. If the agent choose the `bend` action in one of the states where a metal is placed, the new state will be represented with the **negated** coordinates of the old state. For example, in the example above, if the agent choose bend option in (5,3), the new state will be represented as (-5,-3). However, this should not affect any other location. So, if the agent choose to move west from this new state, the next state will be still (4,3). Therefore, checking the sign of a coordination and/or using the absolute function may help you to handle this convention.
- 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 given as **any real number** in the input 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)
<j>                        # number of obstacles
<o_1_x> <o_1_y>            # coordinates of obstacle 1
...
<o_j_x> <o_j_y>            # coordinates of obstacle j
<k>                        # number of pitfalls
<p_1_x> <p_1_y>            # negated coordinates of pitfall 1
...
<p_k_x> <p_k_y>            # negated coordinates of pitfall k
<g_x> <g_y>                # negated coordinates of the goal
<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)
<j>                            # number of obstacles
<o_1_x> <o_1_y>                # coordinates of obstacle 1
...
<o_j_x> <o_j_y>                # coordinates of obstacle j
<k>                            # number of pitfalls
<p_1_x> <p_1_y>                # negated coordinates of pitfall 1
...
<p_k_x> <p_k_y>                # negated coordinates of pitfall k
<g_x> <g_y>                    # negated coordinates of the goal
<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, that is, optimal action for each available state except the goal:

```
<s_1_x> <s_1_y> <a_1>       # state 1 and chosen action in
                              this state (as one of 0:'North', 1:'East', 2:'South',
                              3:'West', and if available, 4:'Bend')
<s_2_x> <s_2_y> <a_2>       # state 2 and chosen action
                              in this state
...
<s_i_x> <s_i_y> <a_i>       # 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 **3**. Your submission will be tested in inek machines. So make sure that it can be executed on them as below.

```
python3 e1234567_hw3.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 converges an expected 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>_hw3.py** (i.e. e1234567_hw3.py).