

CSCI 360 – Spring 2019 – Introduction to Artificial Intelligence Project 3

Due April 26, 2019



https://www.nasa.gov/mission_pages/msl/images/index.html

Description

You are helping NASA to determine the best navigation policy for the Mars Rover. There are obstacles on Mars that the Rover must avoid. If the Rover crashes into an obstacle, you have to pay \$100 to fund [trouble shooting](#). You also spend \$1 for wear-and-tear each time you move. You know the locations of these obstacles from images, and they will not change over time. The Rover will start from its landing location, and will end at a site of scientific interest. When you arrive at your destination site, you will receive \$100 from NASA. However, there is also uncertainty in the Rover's navigation due to the delay in communication. The Rover will go in the correct direction 70% of the time (10% in each other direction, including along borders). **The task is to compute a policy given the terrain and uncertainty using value iteration.**

We will use a grid coordinate system, which will be indexed starting from the top-left corner. An example of a 5 by 5 grid is given below with each cell's coordinates, and with the directions we will use. Note that we use **(col, row) coordinates**

0,0	1,0	2,0	3,0	4,0
0,1	1,1	2,1	3,1	4,1
0,2	1,2	2,2	3,2	4,2
0,3	1,3	2,3	3,3	4,3
0,4	1,4	2,4	3,4	4,4

N

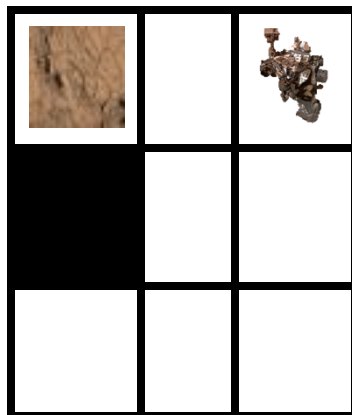
↑

← E


↓

S

Example



+99	-1	-1
-101	-1	-1
-1	-1	-1

	W	W
	N	N
E	N	N

Policy Computation

Mars Rover planning is part of a larger set of problems that involve *planning with uncertainty*, in this case with a fixed, known world. Given knowledge of this world (in the form of a 2-dimensional grid), a movement cost (-1), a reward for reaching the destination (+100), and a penalty for hitting obstacles (-100), you are asked to compute policies given that each movement has uncertainty. You must compute a policy, i.e., a mapping that tells you where your Rover should try to go in each grid location, based on expected utility.

Instructions

You are to compute a policy for a given grid, which has a fixed set of unmoving obstacles and one destination location.

For each grid input, your script will read data from a file (assumed to be named “input.txt” and in current directory) and write results to a file (assumed to be named “output.txt” and in current directory). You will not take arguments from the command line.

Input: The input file will be formatted as follows (all arguments are 32-bit integers):

`<grid_size> // strictly positive`

`<num_obstacles> // non-negative`

`Next num_obstacles lines: <x>, <y> // strictly positive, denoting locations of obstacles`

`<x>,<y> // destination point`

Your submission must be in one file must be named “project3cs360s2019.{py,java,cpp}”

Output:

For each input, you will compute a policy with value-iteration and write a file containing the policy in the following format:

- Obstacles are represented by the letter ‘o’
- EAST is represented by the right-caret character ‘>’
- WEST is represented by the left-caret character ‘<’
- NORTH is represented by the hat symbol ‘^’
- SOUTH is represented by the letter ‘v’
- The destination is represented by a period symbol ‘.’

Example:

input.txt:

4
2
0,0
3,0
2,2

output.txt

ovvo
vvvv
>>.<
>>^<

Evaluation

We will evaluate your code on a set of test cases, which include grids of varying sizes and number of obstacles.

For each grid, we will simulate the rover's movements for 10 starting points that are relatively close to the destination. To simulate, we iteratively move the rover from state to state according to the probabilities given by your policy (random sample: 70% chance in the direction of your policy, 10% each in the other directions). Simulation ends when an obstacle or destination is reached [Note: while we terminate at obstacles in simulation, obstacles are non-terminal during value iteration; See below]. Rewards are added up along the resulting path, and averaged over many simulations.

Your policy passes for a given starting point if it's average reward is within ± 20 of the solution code. You must pass every starting point in order to receive points for a given test grid.

Note that this gives a large window for approximate solutions. Depending on the language you use or your particular implementation, your policy might be different for a large number of states. However, it's behavior close to the destination is roughly constant across implementations.

Implementation Guide:

- In the Bellman equation for value iteration, there are two parameters "Gamma" (γ) and Epsilon (ϵ). For this assignment, set the value of gamma to be **0.9** and epsilon to be **0.1**
- Moving off the grid is considered a valid action (for example, at state (0,0) moving North is off the grid). In this case, consider this a transition from (0,0) to (0,0) with action North.
- Treat obstacles as non-terminal, meaning that the Mars Rover can (theoretically) move into and over an obstacle.
- Rounding: Earlier instructions were to round expected utility. This is no longer critical: You can choose whether or not to round expected utility for your policy selection. Both will be accepted by the grader.
- Tie Breaking: Similarly, you can choose whichever tie-breaking strategy you desire.

Test cases:

- Dev cases are representative (2 each of easy, moderate, and hard cases) of the test cases you will be evaluated on. For all dev and test cases, the correct solution can be computed in less than 1 minute on Vocareum.
- For each test case, you will be able to see whether or not you passed every one of the ten trials. For the smaller inputs, there are duplicates (these can be ignored)