

Artificial Intelligence: Programming 3 (P3) Reinforcement Learning

Due Time: 10PM, 4/12/2021

In this project, we aim to implement one of Reinforcement Learning algorithms: the Q-Learning algorithm.

1 Instructions

We extend the windy maze with probabilistic outcome after an action and a few terminal states with rewards +100 and -100 respectively. The maze map is shown as follows:

- 100						
- 100						
- 100						
- 100						
- 100						- 100
- 100						+100

However, we assume that the agent doesn't know either the reward function or the transition model. The agent aims to run many trials in order to obtain Q-value for each (state, action) pair and the optimal action at each state.

Environment In your implementation, you need to simulate the windy maze environment: We assume that the wind comes from the east and the cost of one step for the agent is defined as follows: 1 for moving westward; 2 for moving northward or southward; 3 for moving eastward. The reward will be the negation of the reward. The agent can drift to the left or the right from the perspective of moving direction with probability 0.1. If the drifting direction is an obstacle, it will be bounced back to the original position. If the agent falls into any terminal state, it can't move out.

Q-Learning In your implementation, you will generate many trials, each of which will result in a trajectory of (state, action, reward) tuple. The agent will use the ϵ -Greedy algorithm to choose an action at each state along each trajectory, where $\epsilon = 0.1$: the agent chooses a latest optimal action at each state with 90% and a random action with 10%. The initial state for each trial is chosen randomly and each trial will end at the goal state. Along each trajectory, the agent will use Q-Learning to update the Q-values. Since the reward function $R(s,a)$ here depends on both the state and the action taken at this state, the Q-value update equations should be revised accordingly (we choose $\gamma = 0.9$).

$$N(s, a) \leftarrow N(s, a) + 1 \quad (1)$$

$$Q(s, a) \leftarrow Q(s, a) + \frac{1}{N_{s,a}} \left(R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \quad (2)$$

Testing and Outputs In your testing, generate 50,000 trials starting from a random open square. We initialize the Q-values at any state-action as 0 except for one terminal state with +100 respectively. If the number of steps of a trial is more than 100, you can abort this trial and continue with next trial to save time. After 50,000 trials (including the aborted trials), report the following three outcomes for each algorithm:

the access frequency at each state-action $N_{s,a}$; the

Q-value function at each state-action $Q(s,a)$; the

optimal action at each state-action.

The expected outcome should look like as follows:

Table of $N(s,a)$:

		127		375		119		189		136		85
-100	145	4886	762	246	400	75	377	4050	196	153	229	73
	164		6467		2596		134		5213		2682	
			364		173				244		228	
-100	####		6073	387	369	154	####		336	245	427	189
			8734		5293				9337		8063	
	78		516		453		353		688		384	
-100	88	3066	570	13483	486	429	402	12661	702	685	14336	385
	84		896		14768		343		25908		404	
			343		692				700		5897	
-100	####		1694	10328	707	680	####		773	688	1834	198
			343		24141				26368		185	
	78		357		835		946		808			
-100	58	2330	215	6666	848	30436	915	1482	867	782	-100	
	68		196		842		33274		30408			
	95		219		495		1189		1802			
-100	74	2790	251	6463	467	16600	1273	45174	1735	64171	100	
	76		186		446		1334		1739			

Table of $Q(s,a)$:

-20.5	-8.8	-7.0	-4.6	-0.1	-0.6
-------	------	------	------	------	------

-100	-70.5	-10.8	-8.8	-8.1	-7.2	-6.2	-6.4	-1.5	-2.8	0.5	-1.8	-0.9
	-23.2		-7.2		-3.8		-4.9		5.0		3.3	
			-7.5		-5.5				2.0		1.3	
-100	####		-7.1	-5.2	-5.2	-2.6	####		5.5	5.7	4.4	3.1
			-1.7		2.5				13.5		8.5	
	-15.7		-5.2		-0.1		8.6		8.5		6.2	
-100	-67.2	-4.9	-6.2	3.8	-0.3	6.9	6.0	12.8	10.3	11.3	13.8	6.8
	-26.3		-2.7		11.5		8.3		21.9		7.5	
			1.0		7.6				15.2		9.6	
-100	####		-4.3	13.2	8.5	15.9	####		26.8	12.6	13.0	-10.2
			11.6		22.7				34.1		-62.1	
	-5.8		1.7		20.0		38.3		4.8			
-100	-57.7	12.5	7.5	24.3	19.7	35.8	33.8	39.0	40.9	-49.9		-100
	-14.2		15.3		33.0		46.9		46.9			
	-0.1		15.8		32.3		45.4		51.1			
-100	-61.0	17.3	7.1	32.3	25.5	46.6	40.8	61.5	52.5	78.4		100
	-10.5		19.4		37.1		49.8		67.2			

Table of the optimal policy:

-100	>>>>	vvvv	vvvv	>>>>	vvvv	vvvv
-100	####	vvvv	vvvv	####	vvvv	vvvv
-100	>>>>	>>>>	vvvv	>>>>	vvvv	<<<<
-100	####	>>>>	vvvv	####	vvvv	<<<<
-100	>>>>	>>>>	>>>>	vvvv	vvvv	-100
-100	>>>>	>>>>	>>>>	>>>>	>>>>	100

where <<<<: moving westward; ^^^^: moving northward; >>>>: moving eastward; vvvv: moving southward; 100,-100: the terminal rewards.

For the first two tables, it is expected that the trend of your outputs should match the above while the exact values could be very different from the above due to the random operations. For the last table regarding the optimal policy, most actions of your output should match exactly with above.

2 Submission

Form a group on Canvas if you want to work with another student. You are going to report the following things:

- Describe in detail how you implemented the following modules in the report: environment simulation, -greedy, and Q-learning update.

- (b) Comment your code in detail so that the grader can understand it well.
- (c) Include the screenshots of all above testing outcomes. Each screenshot should include your username and the current time, which show that you did it by yourself.

The report should be written in a “.docx”, “.doc”, or “.pdf” format. Submit both the report and the source code to the assignment folder P3 on Canvas. Any compression file format such as .zip is not permitted.