CSU33061 AI 1

Homework 2

Submit to Blackboard by Wednesday, 27 March 2024 (23:59)

1 Overview

Homework 2 comes with a Python file

https://www.scss.tcd.ie/Tim.Fernando/AI/hw2.py

If you don't have Python in your computer, you can still do the homework, except for Question Q12 which asks you to run the code.

If you don't want to install Python, you can open it with a text editor such as Notepad for Windows, and GEditor for Linux.

To run the code, do the following:

1. Install python (version ≥ 3.5); this takes about 5-10 mins

   `https://www.python.org/`

   and an IDE of your choice, such as PyCharm (community edition); this takes about 20-40 mins.
2. You need to install 4 packages before running the code[1]

   pandas (for data analysis)

   numpy (for numbers, arrays ...)

   future (for the environment)

   matplotlib (for visualisations)

[1] To install packages for PyCharm, see `https://www.youtube.com/watch?v=e14ilcT79dw`.

2 Problem Description

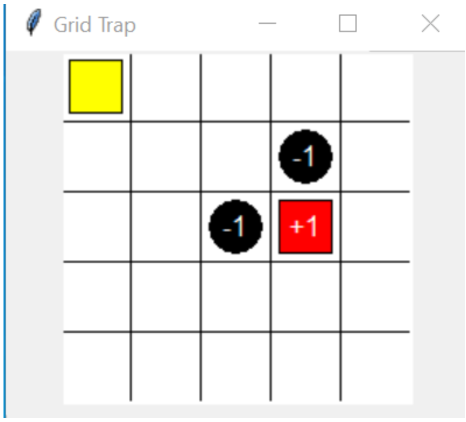

Fig. 1. Game Environment

This is a grid game. The size for this grid is 5*5 shown in Fig. 1.

The yellow square on the top left corner is our agent. We will use Q-learning to help it get to the red square. This is our agent's target. When the yellow agent reaches it, it wins the game and gets reward +1.

The two black circles represent holes which prevent the agent from reaching the red target. When the agent goes into a black hole, it fails and gets reward -1.

For each step, our agent can move one square up, down, left or right. When it goes into the white square, the reward is 0.

When our agent falls into a black hole or reaches the red target, one training episode finishes. The agent uses more than one training episode to learn how to reach the red target.

3 Code Outline

The Python file specifies 2 classes, `Grid` and `QlearningTable`. `Grid` is a grid environment written in Tkinter. You can do Homework 2 without worrying about it (but are encouraged to look into it if you are interested). `QlearningTable` encodes our Q-learning algorithm. **Please note, you have a task here; see Question Q7.** Q7 asks you to complete the definition of the function

`choose_action(self, observation)`.

Some relevant variables in the code are

- `s` and `observation`, for the agent's current state
- `s_` and `observation_`, for the agent's next state.

4 Questions

Q1 (5 points) What is the action space in this problem? ('u' stands for 'up', 'd' stands for 'down', 'l' stands for 'left', 'r' stands for 'right')

A ['u', 'd', 'l', 'r']
B ['u', 'd']
C ['l', 'r']
D ['d', 'l']

Q2 (2 points) How many states are there in this problem?

A 5
B 25
C 20
D 22

Q3 (3 points) What does the Temporal Difference (TD) error represent in Q-learning? (Hint: see Poole & Mackworth, https://artint.info/3e/html/ArtInt3e.html.)

A The difference between the expected and actual rewards
B The error between two estimated Q-values
C The difference between the updated future rewards and the old estimate
D The sum of all rewards obtained in an episode
E The learning rate applied to the Q-value update

Q4 (2 points) Why might we expect the TD error to get smaller in Q-learning?

A The immediate reward is maximized
B The agent always chooses the safest path
C The prediction of the expected future rewards is better
D The learning rate decreases over time
E The discount factor for future rewards increases

Q5 (3 points) How many episodes does the agent need to reach the red target for the first time?

A 7
B 12
C 26
D Not sure

Q6 (5 points) How many columns does the Q-table have after 20 episodes?

A 1
B 2
C 3
D 4

Q7 (5 points) The `QLearningTable` class has a function

`choose_action(self, observation).`

Fill in the missing code for 'if' in Q7.1, and the missing code for 'else' in Q7.2, choosing from

```
A action = self.q_table.loc[observation, :]
B state_action = self.q_table.loc[observation, :]
  action = np.random.choice(state_action[state_action == np.max(state_action)].index)
C state_action = self.q_table.loc[observation, :]
  action = np.random.choice(state_action[state_action == np.min(state_action)].index)
D action = self.q_table.loc[observation, 2]
E action = np.random.choice(self.actions)
```

Q8 (5 points) In the context of Q-learning, how is the TD error calculated for a state-action pair (s, a) transitioning to state s' with reward r?

A $TD(s,a) = Q(s,a) + \gamma \max_{a'} Q(s',a')$
B $TD(s,a) = r + \gamma \max_{a'} Q(s',a') - Q(s,a)$
C $TD(s,a) = r + Q(s,a) - \gamma \max_{a'} Q(s',a')$
D $TD(s,a) = \gamma(r + \max_{a'} Q(s',a') - Q(s,a))$
E $TD(s,a) = r - Q(s,a)$

Q9 (5 points) If we change the number of training episodes, then the number of rows of the Q-table after finishing training will be

A always the same, no matter how many training episodes there are
B not always the same, unless the training episodes are few enough
C not always the same, unless the training episodes are many enough
D always different, no matter how many training episodes there are

Q10 (5 points) In episodes before the training stops, the number of steps our agent needs to get to a terminal state (a black hole or red target)

A is always the same
B decreases monotonically
C increases monotonically
D has a decreasing trend with oscillation
E has an increasing trend with oscillation
F oscillates all the time without any decreasing/increasing trend.

Q11 (5 points) Suppose we use a larger size grid (e.g., 7*7) without changing the other settings. For our agent to reach the target the first time, it needs

A the same number of training episodes
B more training episodes
C fewer training episodes
D Not sure (who knows?)

Q12 (5 points) Run the code for 50 episodes, and show your training result (i.e. the figure 'Steps needed for win'). This figure will be plotted automatically when you run the code. Please paste the figure in your submission.
With reference to your figure, select the correct option from the following:

A The peak of the curve is when the Q-learner is exploiting.
B The ϵ-greedy value is fixed throughout the training.
C A solid horizontal line after 50 episodes suggests an optimal policy.
D Higher reward per episode (minimum steps) is a better optimality criterion than the TD error.
E The Q-learner can learn the value of the optimal policy without much exploration.

5 Your solution

Answer Q1-Q12 in the following format

Q1	Q2	Q3	Q4	Q5	Q6	Q7.1	Q7.2	Q8	Q9	Q10	Q11	Q12

and then paste your figure from Q12.