Homework#2: Q-Learning for Maze Traversal

Due date: 05/19/2022 23:59 PM

Overview

The goal of this homework assignment is to implement the Q-Learning algorithm for solving the maze traversal problem using the existing codebase provided in https://github.com/jostbr/pymaze. The implementation should be in Python and the resulting code should be submitted along with a detailed report that explains the implementation, demonstrates the algorithm's performance, and provides insights into the impact of parameters of the Q-Learning algorithm.

Task

The specific tasks that you need to accomplish for this homework assignment are as follows:

1. Implement the Q-Learning algorithm by adding a new function to the existing codebase. The function should be named q_learning and should take as input a maze object, the number of episodes, the learning rate (α), the discount factor (γ), and the exploration rate (ε).
2. Create a function named q_learning_path that takes as input a maze object and a learned Q-table, and returns the found path from the starting position to the goal position, along with its cost.
3. Train the Q-Learning algorithm on 3 randomly generated 20x20 mazes. You should use the visualization functions already provided in the codebase to create a visual representation of each maze and the found paths.
4. Implement the Q-Learning-based maze traversal algorithm for the randomly generated 20x20 mazes using the following outline:

   a. Initialize the Q-Table with zeros and set values of parameters by yourself.

   b. Set the initial state (the starting position of the agent).

   c. Choose an action based on the current state and the Q-Table, using the ε-greedy strategy.

The ε-greedy strategy is a simple technique for balancing exploration and exploitation in reinforcement learning algorithms. In this strategy, the agent chooses between taking the action with the highest estimated Q-value (exploitation) and taking a random action (exploration) based on a probability ε.

With a probability of 1 - ε, the agent selects the action with the highest Q-value for the current state (i.e., the action it believes to be the best according to its current knowledge). With a probability of ε, the agent selects a random action. This encourages the agent to explore different actions and potentially discover better strategies.

By adjusting the value of ε, one can control the degree of exploration versus exploitation. A higher ε value promotes more exploration, while a lower ε value encourages more exploitation. Typically, ε is initialized with a higher value and gradually decayed over time, allowing the agent to explore more at the beginning of the learning process and rely more on its learned knowledge later on.

d. Perform the action and observe the reward and the next state.

e. Update the Q-Table using the Q-Learning update rule.

The Q-Learning update rule is used to update the Q-values in the Q-Table based on the reward received and the estimated future rewards. The update rule can be expressed as follows:

$$Q(s, a) = Q(s, a) + \alpha * (R(s, a) + \gamma * \max_{a'} Q(s', a') - Q(s, a))$$

where:

Q(s, a) is the current Q-value for state s and action a.

α is the learning rate, controlling how much the Q-value is updated in each step.

R(s, a) is the immediate reward received after taking action a in state s.

γ is the discount factor, representing the agent's preference for future rewards over immediate rewards.

$\max_{a'} Q(s', a')$ is the maximum Q-value for the next state s' and all possible actions a'.

s' is the next state resulting from taking action a in state s.

The Q-Learning update rule is used to adjust the Q-values in the Q-Table based on the observed reward and the estimated future rewards. By iteratively applying this update rule, the agent learns the optimal Q-values for each state-action pair, allowing it to choose the best action for each state.

For defining the reward, please use the following function:

```
def get_reward(current_state, action, next_state, maze):
    if next_state == maze.exit_coor:
        reward = 100
    elif maze.is_wall(next_state):
        reward = -1
    else:
        reward = -0.1
    return reward
```

You can change the variable names or function parameters above, but reward values should be used in the same way.

f. Set the next state as the current state.

g. Repeat steps c-f until the agent reaches the goal or a predefined maximum number of iterations is reached.

f. You should use the visualization functions already provided in the codebase to create a visual representation of each maze and the found paths. Please demonstrates the algorithm's performance from the 3 randomly generated mazes.

5. Choose at least two parameters and change values of them to see the impact of the parameters. (e.g., test the performance when (α=0.2, α=0.4, α=0.6, α=0.8) and (γ =0.2, γ =0.4, γ =0.6, γ =0.8)). Discuss the impact of parameters in your report.
6. Write a detailed report that explains your implementation of the Q-Learning algorithm, including any modifications or enhancements you made to the existing codebase.

Submission

Your submission should include the following:

1. A PDF report that describes your implementation and testing of the Q-Learning algorithm, including the visualization of the solutions obtained. For visualization, you can include the final snapshot after each traversal. Your report should be well-structured, clearly written, and should provide sufficient detail for the reader to understand your work.

2. A README file that provides instructions for running your code, including any dependencies or setup required.

3. All the code you wrote for the homework assignment, including any modifications or enhancements you made to the existing codebase.

Grading

Your submission will be graded based on the following criteria:

1. Correctness and completeness of the implementation of the Q-Learning algorithm.

2. Clarity and quality of the report.

3. Depth of discussion about the effect of parameters.

Note

You are free to use any external resources or libraries that you think might be helpful for this assignment, as long as you cite them properly in your report and do not violate any academic integrity policies. However, you should not use any pre-existing implementations of the Q-Learning algorithm. We note that we run AI-based plagiarism checker.

Package Requirements

Please use *Matplotlib == 3.5.4* for running the code base.

If you are running your code on a MAC OS, FFmpeg file errors may occur. Please proceed with the next process after installing the package using homebrew.
https://www.lainyzine.com/ko/article/how-to-install-ffmpeg-on-mac/

Visualization Examples