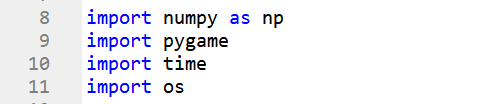
首先看游戏怎么玩。

Hello everyone, my name is Xie Weiru, next I will introduce a Q-learning maze game designed by me.

This is a 6 by 6 maze game, the blue small circle agent from the green starting point, to find the shortest path to the yellow end, while avoiding the red trap and can not go out of the boundary. The key elements of the game design include: the state space consists of 36 grids of the maze, which are positioned with coordinates and determine the actions and rewards of the agent; The action space includes four actions: up, down, left and right. The reward function is set according to 100 points for reaching the end point, 10 points for entering the trap, and 1 point for moving along the ordinary path to guide the agent to find the optimal path.

接下来讲解代码部分

Let's move on to the code section



First, there is the beginning of the code, where a series of import operations are performed.

This is followed by the maze and game window parameters.

This is followed by the definition of color. These colors are used later to draw the maze, agent, starting point, end point, and trap, white for the background, red for the trap, green for the starting point, yellow for the end point, blue for the agent, and black for the border of the maze grid.

动作空间和算法参数设置

Next is the setting of action space and Q-learning algorithm parameters:

ACTIONS defines the action space of the agent, including up, down, left, and right actions. ACTION\_NUM records the number of actions.

GAMMA is the discount factor, set to 0.9, indicating that the agent values the future reward more; ALPHA is the learning rate, 0.1, which controls the speed at which the agent learns new information; EPSILON is the exploration rate, and an exploration rate of 0.1 means that the agent has a 10% probability of randomly choosing an action to explore a new path.

MAX\_EPISODES sets the maximum number of training rounds to 500, and MAX\_STEPS sets the maximum number of steps per round to 100.

Then there is the definition of the maze and the start and end

接着是 Q 值表的加载和初始化

This is followed by the loading and initialization of the Q value table.

This specifies the save file path for the Q value table and the optimal path.

下面是几个关键函数的定义：

Here are the definitions of several key functions:

The get\_state\_index function converts the state to an index in the Q value table, so that the Q value of the corresponding state can be easily searched and updated in the Q value table.

The choose\_action function selects the action based on the ε-greedy policy. If the randomly generated number is less than the exploration rate EPSILON, an action is randomly selected; Otherwise, select the action with the largest Q value in the current state

The get\_next\_state function calculates the next state based on the current state and the selected action. During the calculation, the boundary conditions are checked to ensure that the agent does not walk out of the maze.

The get\_reward function calculates the reward based on the current state of the agent. Get a 100 bonus for reaching the finish line, a -10 bonus for entering the trap, and a -1 bonus on the normal path.

The update\_q\_table function is the core of the Q-learning algorithm and updates the Q-value table based on the current state, the selected action, the next state, and the reward received. It updates the Q table according to the Bellman equation, combined with the predicted Q value and the target Q value.

The next three functions are defined to draw the maze, agent and text respectively.

最后是游戏的主循环部分：

Finally, the main loop of the game:

This code first initializes Pygame and sets the size and title of the game window. Then you enter the training process, and if there is no saved Q value table, you train. In each round of training, the agent starts from the starting point, continuously selects actions, updates status, obtains rewards, and updates the Q value table until it reaches the end or reaches the maximum number of steps. In this process, mazes, agents are drawn in real time, and action, reward, and step information is displayed. After the training, save the trained Q value table and the optimal path. If there is an optimal path, the optimal path is displayed, and the corresponding information is displayed at each step. Finally, exception handling ensures that Pygame exits correctly when the user closes the window.

Through the detailed interpretation of this code, we clearly understand how the maze game based on Q-learning algorithm is realized.

This is the training process，打开train，This is the result of training。

在设计过程中主要遇到了参数调优和UI界面设计两方面挑战并进行了调整：

In the design process, I mainly encountered two challenges in parameter tuning and UI interface design and made adjustments:

First, the learning rate, discount factor and exploration rate affect each other, and improper value will lead to poor learning effect of the agent. For example, the agent may wander and have difficulty finding an effective strategy. In order to solve this problem, the multi-round comparative experiment method was used to set different parameter combinations for training, record the cumulative reward, step number and Q value convergence of the agent, so as to determine the appropriate parameters, and the effect was significantly improved after adjusting the exploration value and other parameters.

Second, there is a situation where the agent walks to the exit but does not show it. By modifying and improving the UI interface design, the agent is displayed above the exit layer when it walks to the exit, and the display effect and experience of the game are optimized.