Reinforcement Learning on HKUSTurkey

Zheng Dongjia	Kuang Bingran	Gao Yang	Xie Zhongkai	Wang Yutong
20546149	20565874	20550946	20550477	20541402

1. Project Introduction

This is a project about maze solving problem. In this project we are going to apply the techniques of reinforcement learning to train the Turkey in the maze to find the logo of HKUST. We have two goals for this project: one goal is to train the Turkey to find the HKUST logo; the other goal is to find the optimized path for solving this problem.

2. Technical Background

2.1 Reinforcement learning

Reinforcement learning is a branch of machine learning that is concerned with how learners perform tasks by taking actions in an environment so as to maximize a numerical reward signal. Unlike supervised learning, the learner in reinforcement learning is not told which actions to take nor is it told what the task to be completed is. Instead, it tries out a series of actions and must eventually learn which actions will give the most reward in the absence of a supervisor. The agent essentially learns by trial and error in this kind of learning.

Reinforcement learning consists of interactions between an agent and the environment. The agent is the learner and the decision maker, while everything that the agent interacts with is part of the environment. While the goal in unsupervised learning is to find similarities and differences between data points, in reinforcement learning the goal is to find a suitable action model that would maximize the total cumulative reward of the agent. The figure below represents the basic idea and elements involved in a reinforcement learning model.

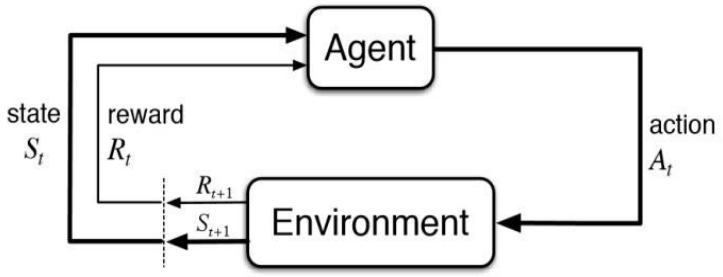

Figure.1 Process of reinforcement learning

Some key terms that describe the elements of a reinforcement learning problem are:
Environment: Physical world in which the agent operates
State: Current situation of the agent
Reward: Feedback from the environment
Action: The agent's movement in each step

2.2 Q-Learning

Q-Learning is a value iteration algorithm which calculates the value or utility of each "state" or "state-action" and then tries to maximize it when performing the action. Therefore, an accurate estimate of each state value is at the heart of our value iterative algorithm. Usually not only the rewards of the current action but also the long-term rewards are taken into consideration to maximize the rewards. In the Q-Learning algorithm, the optimized reward is recorded as the Q value, its calculation formula is as below:

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

3. Project Implementation

3.1 Applications

In our project, the goal is training the turkey in the map that can automatically find the optimal path to the HKUST logo. In the sample map (figure 2), the turkey represents the agent, and the typhoon is the barrier which the turkey needs to avoid and the HKUST loge is the final destination.

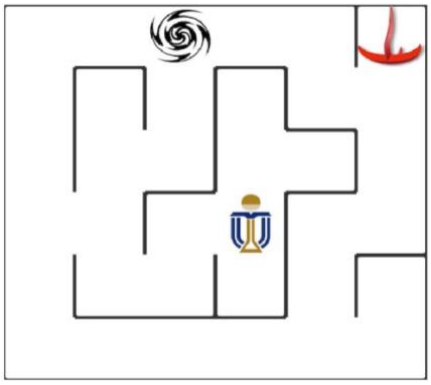

Figure.2 The sample map

In reinforcement learning applications, the coordinate of the turkey in the maze is used to present the state. The actions of turkey can go upper, down, right or left. The rewards are the feedback from the outside, in the project, the turkey get -30 when it meets the typhoon and -10 reward happened when the turkey hits the wall. When the turkey finally gets the destination, there is +50 reward to the table. In other cases, the reward is -0.1 which helps to get the optimal path. The turkey can move automatically with the new values of the Q-table in turkey ‘s “brain”.

3.2 File descriptions

Maze.py	In this file, we use Prim's algorithm to generate the maze.
Turkey.py	This is the main file with application of reinforcement learning.
Runner.py	This is the files to generate training visualizations.

3.2.1 Maze

3.2.1.1 Maze Algorithm

- **Randomized Prim's algorithm**
- Depth-first search
- Recursive backtracker
- Randomized Kruskal's algorithm
- Modified version
- Wilson's algorithm

1. In computer science, Prim's (also known as Jarník's) algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph.
2. Example implementation of a variant of Prim's algorithm in Python/NumPy: Prim's algorithm starts with a grid full of walls and grows a single component of pathable tiles. In this example, we start with an open grid and grow multiple components of walls.
3. This algorithm works by creating n (density) islands of length p (complexity). An island is created by choosing a random starting point with odd coordinates, then a random direction is chosen. If the cell two steps in the direction is free, then a wall is added at both one step and two steps in this direction. The process is iterated for n steps until p islands are created. n and p are expressed as float to adapt them to the size of the maze. With a low complexity, islands are very small and the maze is easy to solve. With low density, the maze has more "big empty rooms"

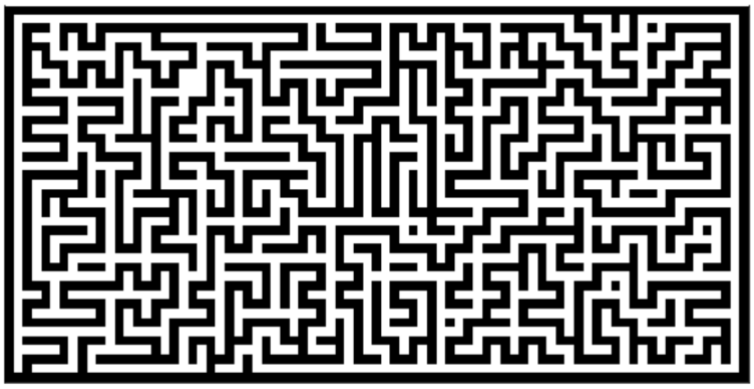

Figure.3 The map layout

3.2.1.2 Usage

The Maze class can help us create the maze with different size and different situation. It can help you create the maze according to your specification. The following list shows the instruction and code for creating a maze.

1. Maze(“file_name”): create the maze file
2. Maze(maze_size=(height, width)): randomly create the maze
3. trap_number: the number of traps
4. g=Maze(“xx.txt”):just use the variable name to print out the maze

3.2.2 Turkey

Turkey Class is the main part of our implementation. We have applied the techniques of Reinforcement Learning to train the turkey so that it can reach the HKUST LOGO in an optimal route. Generally speaking,we try to move the turkey by making command before. However, after adding the Q-Learning Reinforcement Learning, the turkey can now find out the correct route and move by itself! It can learn by itself to update the parameters and Q-table.
The Turkey Class mainly contains the following functions:
-set_status(self, learning=False, testing=False);
Determine whether the turkey is learning its q table, or exceuting the testing procedure.
-update_parameter(self);
This function can help update the value of epsilon during the training process. Usually, we will decrease epsilon after each epoch so that the turkey will spend less time to explore the optimal route. This is a useful method the make a balance between Exploration and Exploitation.
-choose_action(self);
Return an action according to given rules.
-update_Qtable(self, r, action, next_state);
Update the qtable according to the given rule.

-update(self);
This is the most important function for training the turkey. It describes the procedure what to do when updating the turkey. It is called every time in every single step. Finally, it will return current action and reward.

3.2.3 Runner

Runner class is s a video generator. It can note down every single step in each epoch so that we can have an overview of the training process.

4. Project Results

During training, we can select different values of parameters to find the difference of success times chart, accumulated reward chart , running times chart.
The main parameters include:

- Training parameter: Training epoch
- Turkey parameters: Epsilon, Alpha, Gamma
- Maze parameters: The size of the maze and the quantity of trap

The success times chart represents the cumulative number of times our Turkey succeeds in training, which should be an incremental line chart. Accumulated rewards chart represents the accumulated reward value of our turkey in each training epoch, which should be an incremental line chart, and running times chart represents the sequence of training in each training epoch, which should be an incremental line chart.

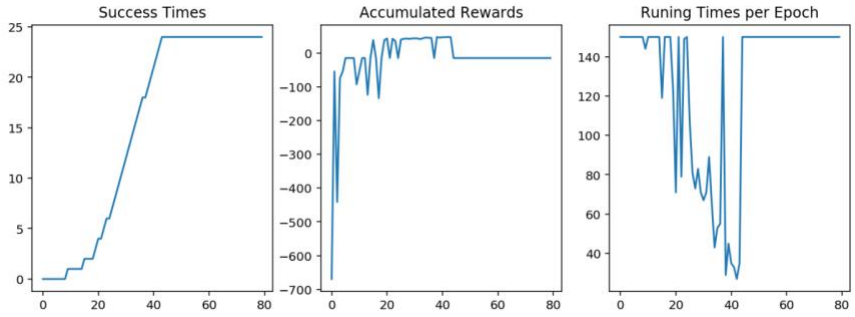

Figure.4 The result performance

Finally, we select alpha = 0.5; gamma = 0.9; epsilon = 0.1. The reasons are as follows, alpha=0.5 because both the old Q value and the new Q value should be taken in to account, so that the robot can learn from the old experience while accepting the new experience averagely. Gamma=0.9 because we want to make the Turkey take fully account of experience of future steps and epsilon=0.1 because we want to make the Turkey make a choice according to the highest reward, but we also want it to try different way randomly at the beginning when it has less experience. This is a trade-off between exploration and exploitation.

5. Collaborations

	Zheng Dongjia 20546149	Kuang Bingran 20565874	Gao Yang 20550946	Xie ZhongKai 20550477	Wang Yutong 20541402
Maze.py	√	√	√	√	
Runner.py	√		√		√
Turkey.py	√	√		√	√
Report	√	√	√	√	√
Video	√	√	√	√	√

6. Youtube link：

HKUSTurkey on Youtube

References

[1] Richard Sutton and Andrew Barto (1998). Reinforcement Learning. MIT Press.
[2] Reinforcement Learning on Wikipedia
[3] Maze Solving Algorithm on Wikipedia
[4] Maze Generation Algorithm on Wikipedia
[5] Path Finding Q-Learning Tutorial