FROZEN LAKE

目標

Frozen Lake Q-Learning:在不改變 num_episodes 和 max_steps_per_episode 的前提下，實現 一致的成功率 > 0.70

演算法與參數設定

演算法：Q-Learning

gamma: 0.99020

Epsilon 衰減率: 0.00007642

Alpha (>0.8): 0.5000, Alpha (>0.5): 0.6391

Alpha (>0.1): 0.4250, Alpha (>0.05): 0.1000

Alpha (>0): 0.0456

訓練過程與探索

- 新增動態調整學習率:減少後期學到不好行為的情況
- 增加gamma:增加模型對未來的重視程度
- 使用gene algorithm:依據成功率調整各個參數的數值

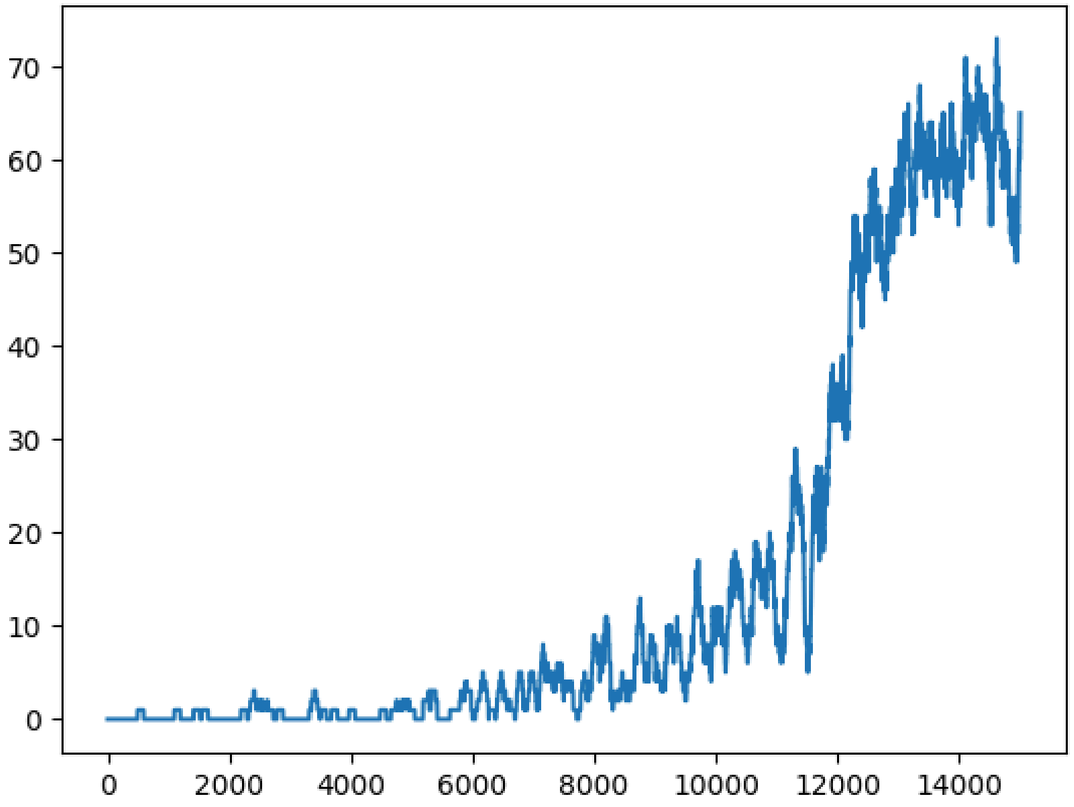
最終評估結果+Demo

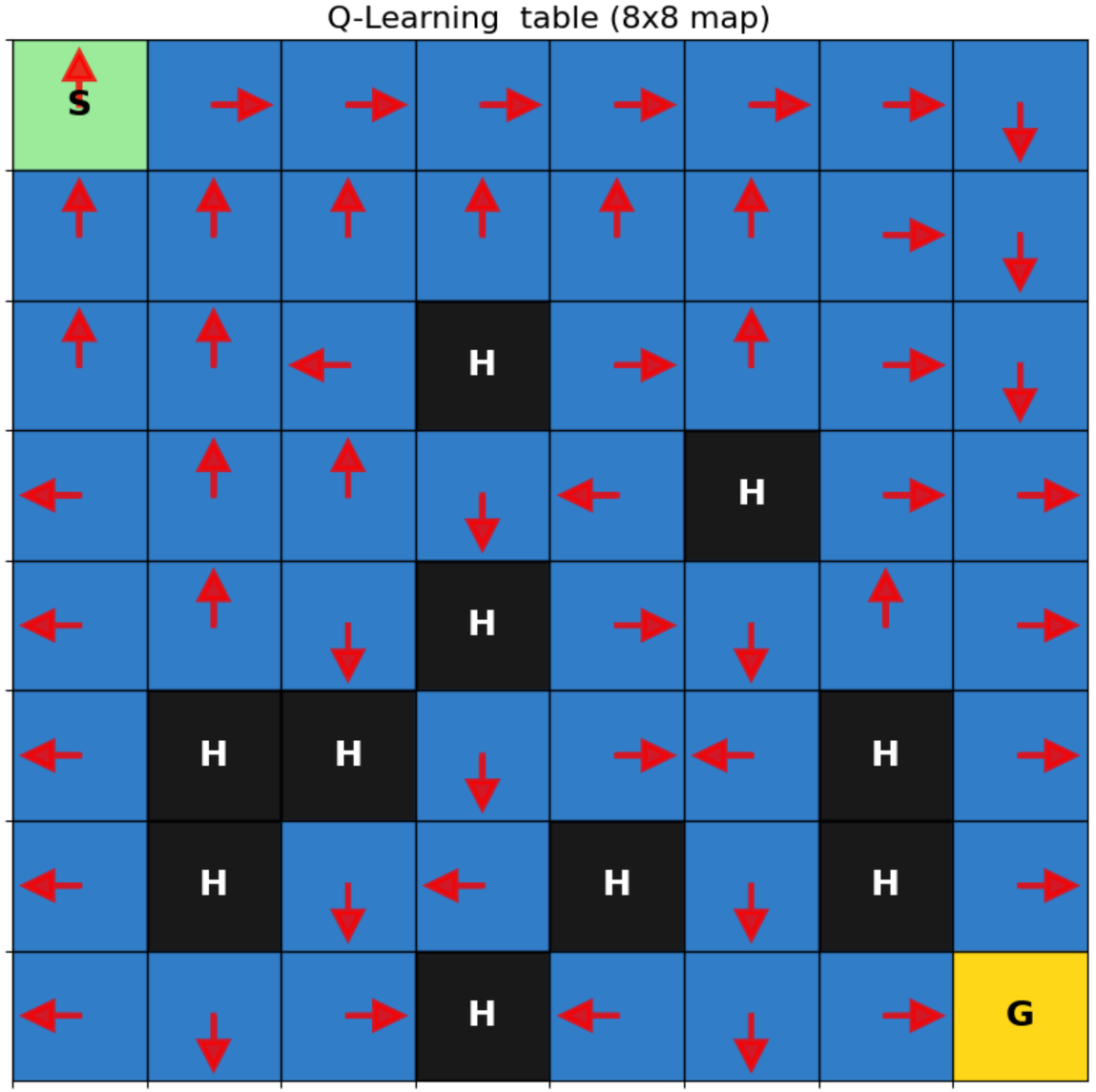
評估方法：使用1000 episodes
評估條件：探索率近似為 0
最終平均成功率(五次平均):60.56%

最終評估結果+Demo


⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：⬅ Left
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
✅ Agent 意圖：➡ Right (意圖) | 實際移動：➡ Right
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：⬅ Left
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬇ Down
✅ Agent 意圖：➡ Right (意圖) | 實際移動：➡ Right
⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：⬅ Left
✅ Agent 意圖：➡ Right (意圖) | 實際移動：➡ Right
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：❌ Stayed (滑倒/原地不動)
⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：❌ Stayed (滑倒/原地不動)
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：❌ Stayed (滑倒/原地不動)
⚠ **【滑倒】** Agent 意圖：⬇ Down (意圖) | 實際移動：⬅ Left
✅ Agent 意圖：➡ Right (意圖) | 實際移動：➡ Right
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬆ Up
✅ Agent 意圖：⬇ Down (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬇ Down
⚠ **【滑倒】** Agent 意圖：➡ Right (意圖) | 實際移動：⬇ Down
Frozen Lake

最終評估結果+Demo


70
60
50
40
30
20
10
0
0
2000
4000
6000
8000
10000
12000
14000


Q-Learning table (8x8 map)
S
H
G

結論

- 高度隨機性 :只有 1/3 的機率能執行預期的動作，而有 2/3 的機率會滑向其他方向 =>即使處於最佳狀態和執行最佳動作也會因而滑入洞中
- 稀疏的獎勵:只有到達終點時才有獎勵=>在訓練初期，代理人很難獲得正向回饋，導致 Q 值更新效率低
- 8 X 8 的迷宮:路徑長度長，隨機性累積效應顯著加上中途無獎勵，導致成功率上限被壓低。
- 使用gene algorithm:採用了 5 次獨立運行取平均的嚴謹評估方式。找到了在當前框架下最穩健且最高效的參數組合。
- 嘗試Sarsa:因為過於保守，常常走不到終點加上稀疏的獎勵，導致成功率更低。
- 嘗試reward shaping:能提供中間指引，但高隨機性使得這些微小的中間獎勵淹沒，無法更新到Q-table。

TIC TAC TOE

Slide 1｜專題簡介

專題簡介

遊戲模式：

- Human vs Human
- AI vs Human
- AI vs AI

AI 難度：

- Easy：Random Strategy
- Medium：Rule-based Strategy
- Hard：Minimax Strategy

操作流程：

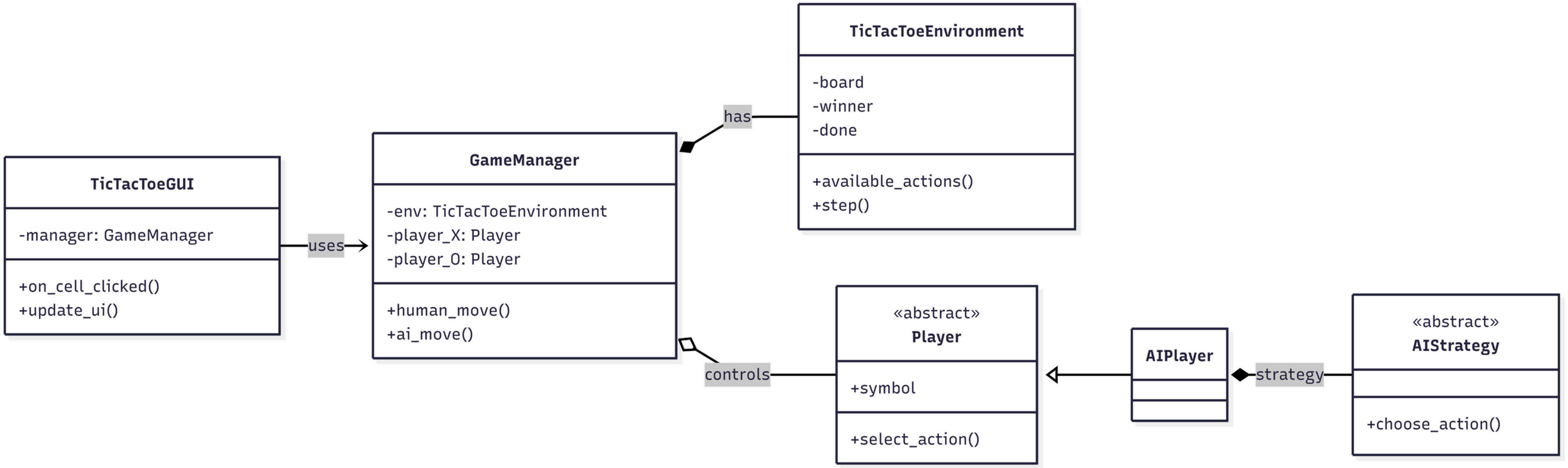
1. 選擇遊戲模式與難度
2. 系統初始化棋盤
3. 玩家輪流下棋
4. 判斷勝負或平手

Slide 2｜系統展示 Demo

```
5   ### Part 3: Tic-Tac-Toe (GUI and AI)
6
7   This project implements a fully modular **Tic-Tac-Toe (井字棋)** game using object-oriented design principles.
8
9   The system is carefully struc
10
11  * Game environment and rules
12  * Player strategies (Human ar
13  * Game flow control
14  * Graphical user interface (G
15
16  This design improves code rea                                    nstrates proper use of OOP concepts.
17
18  ### Supported Game Modes
19
20  * Human vs Human
21  * Human vs AI
22  * AI vs AI
23
24  ### AI Difficulty Levels
25
26  * **Easy**: Random move selec
27  * **Medium**: Rule-based stra
28  * **Hard**: Minimax algorithm
29
30  The graphical user interface                                     teractively select game modes and diffi
31
32  ---
33
34  ## Project Structure
35
36  ```text
37  .
38  ├── environment.py      # Tic-Tac-Toe environment (board state, rules, win/draw checking)
39  ├── players.py          # Player implementations (Human and AI players)
```

Slide 3｜整體 OOP 架構（UML 簡化版）

UML簡化版


TicTacToeGUI
-manager: GameManager
+on_cell_clicked()
+update_ui()
uses
GameManager
-env: TicTacToeEnvironment
-player_X: Player
-player_O: Player
+human_move()
+ai_move()
has
TicTacToeEnvironment
-board
-winner
-done
+available_actions()
+step()
controls
«abstract»
Player
+symbol
+select_action()
AIPlayer
strategy
«abstract»
AIStrategy
+choose_action()

Slide 4｜Environment：遊戲規則封裝

Environment：遊戲規則封裝

- 儲存棋盤狀態（Board State）
- 提供合法行為（available actions）
- 判斷勝負與平手
- 控制遊戲是否結束

Slide 5｜Player + Strategy Pattern

Player + Strategy Pattern

Player Hierarchy（多型設計）

- Player（Abstract Class）
  - 定義統一介面：select_action(env)
- 實作類別
  - HumanPlayer
  - AIPlayer

AI Strategy Pattern（策略模式）

- AIStrategy（Abstract Strategy）
  - 定義介面：choose_action(env)
- 具體策略
  - RandomStrategy（隨機）
  - MediumStrategy（規則 + 防守）
  - MinimaxStrategy（最佳解搜尋）

Slide 6｜GameManager：流程控制

GameManager：流程控制

GameManager 職責

- 建立 Environment 與 Player
- 控制玩家輪替
- 協調 GUI 與 Model