

OOP Final Project Group 12

林柏儒 B123040045

吳紹彰 B123040048

張承勛 B123040053

1. 摘要

這次的作業分成三個部分，part 1 為測試環境是否有問題，在我們的測試中這部分是沒問題的。part 2 為 Frozen Lake 去嘗試取得大於 0.7 的成功率，經過我們反復嘗試後成功率停留在 0.66 左右。而 part 3 我們使用大量 OOP 的概念去實現了貪食蛇。

2. part 1

根據 README.me 要求以正確的方式啓動虛擬環境與下載需要的套件即可成功執行 carPole 程式並圖像化顯示

3. part 2

在 part 2 中，原本 learning rate 是固定的，我們將其改為會隨著時間降低，希望模型在訓練早期能夠更傾向外探索，到後期慢慢開始收斂後就不要跳動太大。我們還改了 reward 機制，由於原本的 reward 只有在到終點時才會有獎勵，其他情況無論怎麼失敗 reward 都一樣，這會導致一定要有一次隨機走到終點才學得到東西，在這部分我們改成了失敗時會根據與終點的距離決定給予的 reward，解決了只有到達終點才會有 reward 的問題，計算距離的部分我們是採用 BFS 計算距離。調整超參數的部分我們是用 optuna 這個套件來協助我們訓找超參數。

4. part 3

4.1. Agent

我們的 Agent 分成以下 6 種，繼承自 BaseAgent

1. **RandomAgent**
每次移動都選擇隨機的一個方向
2. **GreedyAgent**
每一步都貪心選擇讓蛇頭更接近食物的位置。使用曼哈頓距離選擇離食物更近的方向。
3. **RuleBasedAgent**
優先保持安全（不撞牆、不撞自己），在安全方向中選擇與食物最近方向前進。
4. **PathfindingAgent**
以 BFS 在地圖上規劃從到食物的最短路徑。把蛇身也一起視為障礙物，每次移動前都會執行

BFS 尋找路徑，找到最短路徑後，再延路徑的方向移動。

5. **HamiltonianCycleAgent** 此 Agent 只適用於沒有障礙物的地圖，對於無障礙物且長寬為偶數的地圖可以很容易構建出哈密頓迴圈，這個迴圈可以走過地圖上所有點恰一次且起點終點相連，如此一來按照哈密頓迴圈移動就可以保證不會撞到自己，理論上可以吃到最多的食物，但是由於移動步數有上限，因此通常到一半還沒撞到前就會因為步數上限提早結束。
6. **ReinforcementLearningAgent** 這個 Agent 使用 Q-learning 演算法，在選擇移動方向時會根據 Q-table 的值選擇或是有一定機率隨機選擇，在每一步後會再去調整 Q 值讓模型去學習。

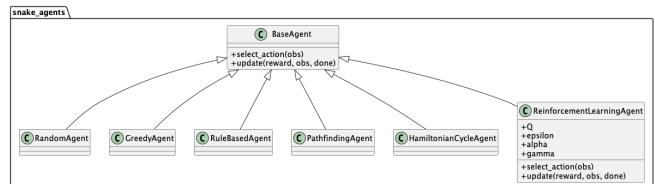


圖 1: Agent UML 圖

4.2. Map type

地圖分成以下 2 種，繼承自 BaseMap

1. **EmptyMap**
地圖上沒有障礙物
2. **ObstacleMap**
地圖上有障礙物，障礙物會在地圖中生成，生成出來的障礙物也可能會自己消失

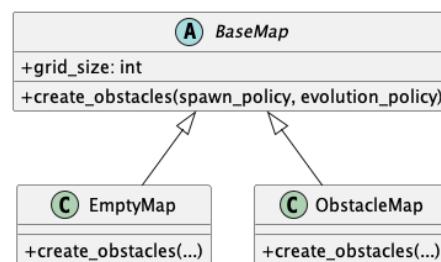


圖 2: Map UML 圖

4.3. Obstacle Shape

若地圖是使用 ObstacleMap 則地圖上會有障礙物，障礙物共有 10 種，繼承自 ObstacleShape，其障礙物形狀參考自康威生命遊戲中的常見狀態，這邊僅列舉三個。

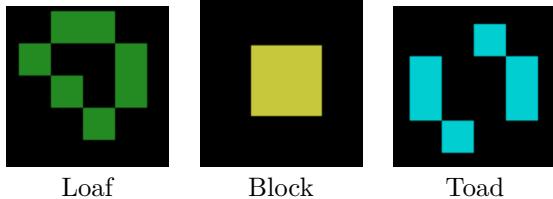


圖 3: 障礙物

圖 4 部分由於障礙物物件太多，UML 圖中僅畫出圖 3 中的障礙物

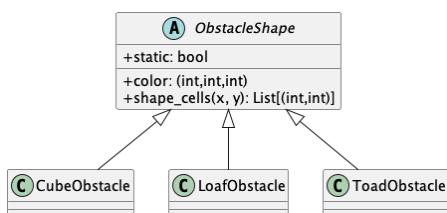


圖 4: Obstacle Shape UML

4.4. Evolution Policy

Evolution Policy 分成兩種，繼承自 EvolutionPolicy

1. NoEvolutionPolicy

地圖上的障礙物形狀不會隨時間變化（但還是會隨時生成或消失）

2. ObstacleMap

地圖上的障礙物形狀會受康威生命遊戲規則影響，以下為康威生命遊戲的規則：

1. 對於每個活著的細胞，若身邊活著的細胞數量小於 2 個，在下個階段則會死亡
2. 對於每個活著的細胞，若身邊活著的細胞數量為 2 或 3，下個階段則會繼續活著
3. 對於每個活著的細胞，若身邊活著的細胞數量大於 3，下個階段則會繼續死亡
4. 對於每個死亡細胞，若身邊活著的細胞數量恰等於 3，下個階段則會復活

按照以上康威生命遊戲的規則，將活著的細胞當作障礙物，死亡細胞則是蛇可以行走的地方，就可以讓地圖中的障礙物活起來，有些像是 Toad 等細胞根據規則會不斷的變換狀態，讓地圖上的障礙物看起來像是活過來了。

5. OOP concept

5.0.1. Abstraction (抽象化)

- 1) Gym： SnakeEnv 繼承 gym.Env，固定使用 reset/step/render/close。外部（訓練程式、demo、agent runner）只使用這個 interface。

2) Other Interfaces :

- RewardPolicy：抽象化了 reward / terminate / truncated，ClassicRewardPolicy 是其中 reward policy (含 step penalty、吃到食物加分與 max_steps truncated 等等)。
- EvolutionPolicy：抽象化障礙物的演化規則 (NoEvolutionPolicy、ConwayEvolutionPolicy)。Obstacle 只依賴各自 override 的 evolve(...)function。
- BaseMap：抽象化地圖是如何建立障礙物的規則，由 EmptyMap 與 ObstacleMap 決定是否啓用障礙物。

- 3) 底層 object interface：GridObject(ABC) 作為幾乎所有在地圖會出現的抽象 object，統一提供 grid size 與 reset() interface，讓 Snake/Food/Obstacle 能以統一被 SnakeEnv 管理。

5.0.2. Encapsulation (封裝) .

- 1) Snake 的封裝：Snake 封裝 body (deque)、direction、alive、grow_pending。

- Snake.change_direction(action) 封裝了蛇不能直接 180 度迴轉的規則：用向量相加判斷是否為反方向。
- Snake.move() 封裝邊界死亡、自撞死亡、以及身體移動的更新 (先 push head，再視 grow_pending 決定是否 pop tail)。
- 外部不需要也不應該直接改動 deque，只要呼叫 move/grow/reset 即可維持其正確的移動。

- 2) Food 的封裝：Food 封裝 position / value / color 與 respawn()。

- respawn 將不能生存在蛇身體上、不能生存在障礙物上封裝為內部的一個 set 以利之後檢查。
- 透過 value (分數) 與 color (顯示) 讓環境與 renderer 能以一致方式處理不同食物。

- 3) Obstacle 的封裝：Obstacle 封裝多個複雜行為：形狀、生成合理的位置、移除、康威演化規則，以及提供 positions (位置) 與 check_collision()。

- 4) Renderer 的封裝：Renderer 隱藏 pygame 初始化、視窗/clock 管理、畫面更新等細節。SnakeEnv 不需要使用任何 pygame，只要呼叫 draw 與 close。

5.0.3. Inheritance (繼承) .

- 1) **Food** : Food 為 base class , RedFood 與 GoldFood 透過 override 初始參數來表達不同 attribute (value 與 color)。
- 2) **Map** : BaseMap 為障礙物決定的 base class , EmptyMap 與 ObstacleMap 繼承 BaseMap, 並以不同方式 return 給 Obstacle 決定是否有障礙物。
- 3) **Policy** : EvolutionPolicy 為演化 policy 的 base class , NoEvolutionPolicy 與 ConwayEvolutionPolicy 分別代表不演化與康威規則演化。
- 4) **Agent** : BaseAgent 是所有 agent 的 base class , 提供移動方向 interface 與 method 。

5.0.4. Polymorphism (多型) .

- 1) **Food** 多型 : SnakeEnv._spawn_food() return 的東西是 Food , 但可能是 RedFood 或 GoldFood 。
- 2) **Policy** 多型 : SnakeEnv 透過 constructor 參數使 reward_policy 、 map_strategy 、 spawn_policy 、 evolution_mode 對應不同 EvolutionPolicy 。呼叫端可在不修改 SnakeEnv 的情況下替換規則。
- 3) **Agent** 多型 : Runner 主程式只要有 BaseAgent 介面就可。不同 agent 實作各自的 select_action(obs) (例如貪婪、路徑搜尋、 RL) , 但對外的使用方式一致。

6. AI tool I used

在此次 final project 中 , 程式與文本撰寫均有使用 AI tool 。程式部分舉 part 3 為例 , 基本 OOP 架構大部份為我們所設計的 , 而部分 method , 貪吃蛇的行為、障礙物、各個 Agent 等等有使用到 AI tool , 註解則有些是自己家的有些是使用 AI tool 加的 , part 2 也是類似的做法。文本部分 README 中的環境與執行方式部分是由 AI tool 生成 markdown 後由我們進行修改的 , UML 也是類似做法 , 由 AI 生成 IUML 的程式碼再由我們進行增減。另外文本部分也有使用 AI tool 的協助 , 以 OOP concept 為例 , 我們先把我們已知有使用到的 OOP concept 列出 , 再使用 AI tool 分析我們文本中有沒有漏掉的 OOP concept , 再將其補上 。