

應用於吃豆人之強化學習

課程名稱：人工智慧

組員：陳亭禎、郭冠宏、李宗穎

授課教師：曾士桓 博士

摘要

本專題旨在模擬兩個 Agent 如何利用所處環境中的資訊，進行對抗並達成各自目標。我們運用了強化式學習中的 Q-learning 與 SARSA 演算法進行實作，作為兩個 Agent 的學習方式，並透過 Approximate Q-learning 與 Approximate SARSA 改善其效能，實驗結果顯示：Approximate Q-learning 與 Approximate SARSA 成功改善 Q-learning 與 SARSA 的不足之處，實現兩個 Agent 互相對抗並達成各自目標之目的。

1.前言

1.1. 研究動機

隨著時代的遷移，為了享受科技所帶來的方便與進步，人工智慧開始崛起，而其中，讓機器擁有自我學習的能力是現正要面臨的一大挑戰。從人工智慧所衍生出來的機器學習與深度學習也是值得探討的學習方式，而本篇將針對機器學習進行研究。

在機器學習當中，又分為監督式學習、非監督式學習以及強化式學習，其三者差異分別是，監督式學習需有特定答案讓電腦學習；非監督式學習則是讓電腦從特徵當中自行尋找關聯性；而強化式學習則是基於環境讓電腦學習[1]。因本專題所研究之吃豆人遊戲主要是依據環境來建置，故使用強化式學習進行撰寫。

1.2. 研究問題

根據上述所提到的研究動機，本專題之研究目標在於利用強化式學習

讓吃豆人可達成避開怪物且吃完所有豆子之目標。因強化式學習之精神在於學習每種不同情況下，所該採取何種對應動作，使得結果最佳化[2]，固本專題主要方法包括 (1) Q-Learning；以及 (2) SARSA。利用上述兩種演算法，加以改進後，以此達到吃豆人成功避開怪物並吃得豆子之目的。

2.文獻探討

2.1. Q-Learning

Q-learning 演算法為眾多強化學習演算法中最經典的一個，除了本身為主動(active)探索環境的特性外，也是個 off-policy 的演算法；主要藉由代理人(agent)在環境中根據不同的狀態(state)執行選取的行為(action)，學習出能產生較佳的策略(optimal policy)，使得 agent 能在環境中回饋(Reward)達到最佳，其中 Q-value 演算公式[3]如下：

$$Q(s,a) = Q(s,a) + \alpha[R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \quad (1)$$

公式中，含有基礎變數定義(如： s =CurrentState, a =CurrentAction, s' =Next State, $Q(s,a)$ 稱之為 Q-value：為一估算值，用來評斷 agent 在 s state 上做出 action(a)成效的好壞)，其中較為重要的參數有 α (learning rate)通常數值介於 0 至 1 之間，用來決定我們每次更新參數時我們改變原來 Q-value 的程度、 γ (discount factor)為控制未來獎勵對 Q-value 更新的影響、 $R(s)$ 為 agent 在目前 state 上所獲得的 reward，而在公式 Q-value 公式中可得知 Q-learning 已不再利用 Transition Matrix 以及 Re

ward 當成 model 進行學習，而是透過與環境互動學習出自己的策略，故也被稱之為是個 model-free 的演算法。

2.2. SARSA

除上述所提及的 Q-Learning 外，另一個在強化式學習中頗具盛名的演算法即為 SARSA。SARSA 與 Q-Learning 極為相似，其更新規則皆是以 temporal-difference (TD) 為基礎。而兩者的不同之處在於：Q-Learning 為 off-policy，代表著其更新規則並不同於 agent 實際的 state 與 action；相反的，SARSA 則是根據 agent 實際的 state 與 action 對 Q-value 進行更新，更新公式如下：

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \quad (2)$$

其中，可以發現對於下一個 state 與 action 的更新不再是選擇最大 Q-value 的 state 與 action，而是實際到達的下一個 state 與 action，有鑑於此，相較於 Q-Learning，SARSA 的收斂速度較快，但其代價便為容易陷入區域最佳解。在 [4] 與 [5] 亦有提及，在某些應用中，SARSA 具有比 Q-Learning 更好的結果。

3. 研究方法及步驟

本專題中利用了 [6] 所建立的 Pacman 程式為基底進行撰寫，為了因應各階段工作性質之不同，我們將專題工作內容區分為三項區塊以利後續執行，其分別為：(1) 定義各項 State、Action 及 Reward；(2) 演算法撰寫；以及 (3) 動畫圖形繪製。本專題整個流程於附錄一當中有詳細流程圖，而吃豆人與怪物之演算法流程圖也皆有附於其後提供參照。

3.1. 定義各項 State、Action 及 Reward

本專題主要呈現畫面為一 5X6 的地圖空間且含有 4 顆主要的豆子，吃豆人每吃一顆豆子加上分數 (10 分) 作為回饋，當吃完所有豆子時，則 Reward 全部加上 500 分，各項 state、action 及 reward 皆如圖 1、2 及 3 所示。

input	state	(x,y)				
	action	上	下	左	右	停止
output	state	(x,y+1)	(x,y-1)	(x-1,y)	(x+1,y)	(x,y)

圖 1. 吃豆人 state 與 action

input	state	(x,y)				
	action	上	下	左	右	停止
output	state	(x,y+1)	(x,y-1)	(x-1,y)	(x+1,y)	(x,y)

圖 2. 怪物 state 與 action

Agent	Event	Reward
吃豆人	吃到豆子	Score加10分
	吃完豆子	Score加500分
	被怪物抓到	Score減500分
	每經過一個Action	Score減1分
怪物	抓到吃豆人	怪物Score加500分
	每經過一個Action，且沒有抓到吃豆人	怪物Score減1分

圖 3. Agent Reward

3.2. 演算法撰寫

3.2.1. 參數設定

本專題中，我們對於參數的設定方式採取動態進行設定，因每個參數皆有各自的涵義與作用，故將於以下分別進行說明：

- α (learning rate): 作為 Agent 的學習率，其負責於更新過程中控制對於目前 Q-value 數值改善之程度，若 α 愈高，則改善程度較大；反之則較低。在本專題中，我們希望 Agent 一開始具有較強的學習能力，使其不斷朝著全域最佳解邁進，因此便將 α 之初始值設定為 0.6，而每當經過 20% 的總訓練迭代次數後，將其乘以 0.6，直到 80% 的總訓練迭代次數後， α 值為 0.078，趨近於 0，藉此加強其開發的行為。
- γ (discount factor): 用於控制未來獎

勵對於目前 Q-value 更新的影響，故可得知，當 γ 較高時，未來獎勵對於本次更新的影響較大；反之則較小。我們希望 Agent 於初期能夠探索不同的可能，以找尋全域最佳解，有鑑於此， γ 的初始值設為 0.4，藉此使 Agent 更注重於長期回饋，並同樣每經過 20% 的總訓練迭代次數後，將其乘以 1.2，直到 80% 的總訓練迭代次數後，其值約為 0.829，得以讓 Agent 於後期更傾向短期回饋，朝著開發之行為前進。

- ϵ ：用於控制 Agent 選擇 Action 時的參數，同時也是左右 Agent 進行探索或開發之關鍵，而其選擇 Action 的方式為產生一隨機數，與 ϵ 比較，以決定要選取探索抑或開發的 Action，一般而言， ϵ 較高，則表示探索的機會較高；反之則是開發的機會較高。由於我們想使 Agent 於前期不斷地進行探索，因此將 ϵ 設為 0.6，使其具有 6 成之機率嘗試不同之 Action，同樣，每經過 20% 的總訓練迭代次數後，將其乘以 0.6，直到 80% 的總訓練迭代次數後，其值約為 0.078，趨近於 0。透過此方式，期望能使 Agent 於前期具有高度的探索空間，並隨時間推移，配合開發之行為，往全域最佳解邁進，使 Agent 建立出一套屬於自己的最佳策略。

3.2.2. 吃豆人 Algorithm1(Epsilon-greedy Q-learning)

首先，我們嘗試將 Epsilon-greedy Q-learning 運用於吃豆人上，作為其 Agent，下表為其虛擬碼：

表 1. Q-learning 演算法虛擬碼
Algorithm1 : Q-learning

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step of episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 
  until  $s$  is terminal

```

在上述演算法當中，首先會將 Q-value 初始化為 0，本身為一 lookup table 資料結構，而在行為選取(action selection)的策略上，我們運用了 Epsilon-greedy 的方式來進行，使 agent 在學習時能夠同時擁有探索(exploration)及開發(exploitation)的能力。

3.2.3. 怪獸 Algorithm1 (SARSA)

在實作過程中，我們將 SARSA 演算法應用於怪獸的 Agent 上，其虛擬碼如下：

表 2. SARSA 演算法虛擬碼

Algorithm1 : SARSA

```

Initialize  $Q(s, a)$  arbitrarily
Repeat (for each episode):
  Initialize  $s$ 
  Repeat (for each step of episode):
    Choose  $a$  from  $s$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $a$ , observe  $r, s'$ 
     $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$ 
     $s \leftarrow s'$ 
  until  $s$  is terminal

```

演算法的最一開始，會先將 Q-value 初始化為 0，並根據每次的 state 與 action 進行更新，而在行為選取的策略上，同樣是採用 Epsilon-greedy 的方式，藉以控制開發與探索間的平衡。

然而，將 Q-learning 與 SARSA 實作完成後，我們進行訓練並測試，過程中發現，吃豆人與怪獸的行為並不如我們所預期，明顯發現結果並無收斂，因此，我們嘗試更改迭代次數，從原本的 2000 次，不斷遞增至 8000 次，亦將參數進行數次的調整 (α , γ , epsilon)，但仍舊無法得到一個好的結果，因此，我們嘗試利用 Q-learning 的 exploration 及 exploitation 作為修正，進而產生了吃豆人與怪獸的 Algorithm2。

3.2.4. 吃豆人

Algorithm2(Approximate Q-learning)

為一種基於 Q-learning 發展的演算法，透過將 Q-function 從以查表法 (look up) 的方式換成以權重 (Weight)* 特徵 (Features) 的方式，除了不僅僅給予電腦關於 state, action, reward, and next state 試圖找出最佳策略外，Approximate Q-learning 利用了所謂的特徵萃取 (Feature Extract) 將較為重要的特徵進行提取，同時將 Q-function 轉變成以 function approximation 的方式進行計算，如下列公式所示：

$$Q(s,a) = \sum_{k=1}^n f_k(s,a)w_k \quad (3)$$

在實作中，我們共設定了四個特徵 (豆子是否被吃掉、距離下一個豆子的距離遠近、是否即將撞上怪獸、是否與怪獸只差一步的距離)，以及各自特徵的權重 w_k 計算 Q-value；而更新策略上變成以更新各特徵的權重數值來加強吃豆人在策略上的優化，如下列公式所示：

$$w_i \leftarrow w_i + \alpha \square difference \square f_i(s,a) \quad (4)$$

$$difference = (r + \gamma \max_{a'} Q(s',a')) - Q(s,a) \quad (5)$$

透過 Widrow-Hoff rule 上將 Q-learning 上的 difference 數值進行 MSE (Mean Square Error) 的數值求出誤差數值後，將其更新於各權重，使得 Q-function 計算能夠愈來愈近似 (approximate) 真實的 Q-function。

3.2.5. 怪獸

Algorithm2(Approximate SARSA)

此演算法為我們基於 Approximate Q-learning 改良而成，故稱為 Approximate SARSA。其概念皆與 Approximate Q-learning 相同，僅於更新公式中略有差異，其公式如下：

$$difference = (r + \gamma Q(s',a')) - Q(s,a) \quad (6)$$

在實作中，我們為怪獸設定了兩個特徵，分別為：(1) 怪獸是否吃掉吃豆人，與 (2) 怪獸與吃豆人的距離，透過這兩種特徵與各自計算完成的權重值，改善原本 SARSA 所遭遇之瓶頸。

3.3. 動畫圖形繪製

本專題所製作之強化學習遊戲需利用到兩個角色，分別為「吃豆人」與「怪物」。畫面視覺呈現除了吃豆人與怪物之外，還有吃豆人所需完成的「吃豆」動作，而地圖中就需有豆子供吃豆人進行吃豆動作。為讓遊戲有實際操作的體驗，故在一開始有選單提供使用者進行選擇，其選單內容分別為：(1) 開始遊戲；(2) 關於遊戲；以及(3) 離開遊戲，如若使用者點擊開始遊戲，則程式將會開始呈現演算法訓練過程，訓練完畢後，將會以每訓練 50 次即呈現 10 次結果；若是點擊關於遊戲，則會顯示本專題的成員與遊戲介紹；最後若是點擊離開遊戲，則結束所有遊戲畫面，其中介面是使用 Python 套件當中的 tkinter 不斷更新遊戲畫面，其中畫面更新率為每 0.05 秒更新一次；而選單畫面則由 Python 套件當中的 Pygame 進行更新。本專題中的動畫圖形繪製步驟將分別針對：(1) 選單製作；(2) 怪物繪製；以及(3) 吃豆人與豆子繪製進行以下說明。

3.3.1. 選單製作

本專題初始介面為一選單 (如圖 4)，此選單是利用 label() 函式所建置各項選擇，並利用 Python 當中的 Pygame 套件來不斷偵測鍵盤中的上下鍵以及 esc 按鍵，若按下“下鍵”，index 狀態加 1 (若 index 超過目前所擁有選項個數，則不繼續進行加法)；反之，若按下“上鍵”，index 狀態減 1 (若 index < 0，則不繼續進行減法)；而按下

“esc”，則 index 狀態設為 2。根據所做動作對應結果，每項選擇都有各自對應之 index 作為當下狀態，例如：初始狀態設為 0；Start 被選中之狀態為 0，亦作為初始狀態，當狀態為 Start，則會呼叫演算法執行訓練；About 被選中之狀態為 1，當狀態為 About，則會顯示已設定之訊息；Exit 被選中之狀態為 2，若遊戲偵測狀態為 2，則會停止所有遊戲（如表 3）。

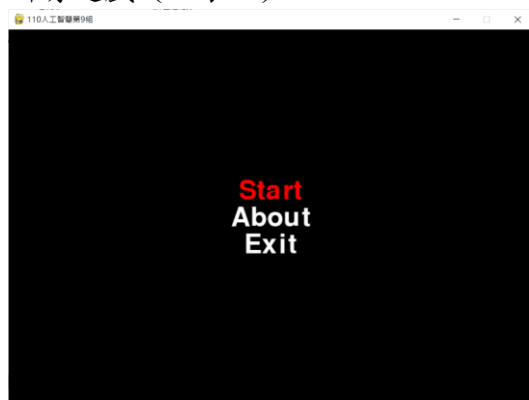


圖 4. 初始介面選單

表 3. 選單對應表

Index 狀態	選單 項目	對應結果
0	Start	呼叫演算法
1	About	呼叫顯示訊息頁面
2	Exit	停止所有遊戲

3.3.2. 怪物繪製

本專題怪物之圖形因不是常見的幾何圖形，故利用 Python 當中的 polygon() 函式進行怪物圖形繪製，而其規則是須以順時針或逆時針將圖形勾勒出來，故先行使用陣列將所設計之怪物圖形以座標方式進行順時針的儲存，儲存完畢後，即可呼叫 polygon() 進行繪製；而其眼睛是以 circle() 先行繪製並以 append() 加入，最後再根據所移動方向進行 x、y 修正，最終可得怪物圖形。

3.3.3. 吃豆人與豆子繪製

本專題吃豆人與豆子圖形皆是以 circle() 進行繪製，而其中吃豆人與豆子不同的是，吃豆人須有吃豆的動畫效果，效果呈現方式是根據狀態改變嘴巴開合的方向，其作法是先取得現在應面朝的方向，再利用更新畫面重新定義 circle() 的 endpoint，透過取得方向及 endpoint 不斷重新繪製以更新畫面，便可達成吃豆效果。

4. 實作成果

本專題初始介面會以選單提供使用者選擇，若根據圖 5 點擊【About】，畫面會呈現簡介及組員資訊（如圖 6）；若根據圖 7 點擊【Exit】，則遊戲將全部關閉；最後若是根據圖 8 點擊【Start】，吃豆人演算法執行完畢後，畫面會顯示出演算法結果（如圖 9）。

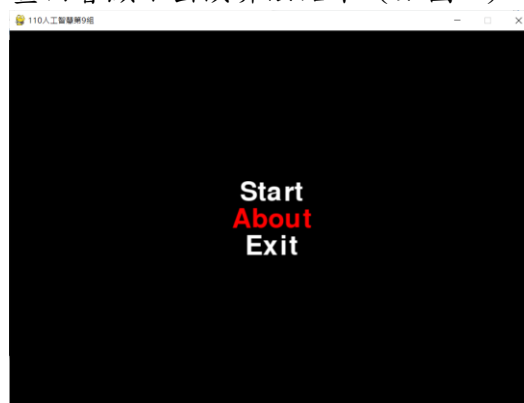


圖 5. 選單介面_About

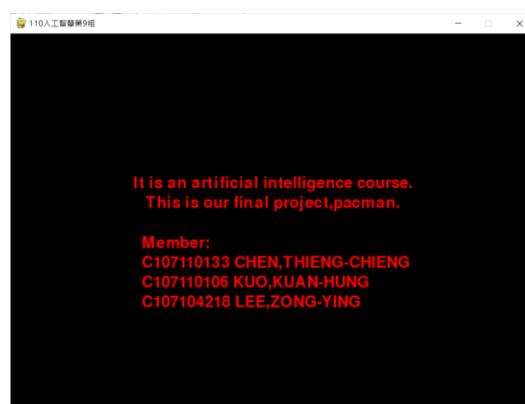


圖 6. 選單介面_About_組員資訊

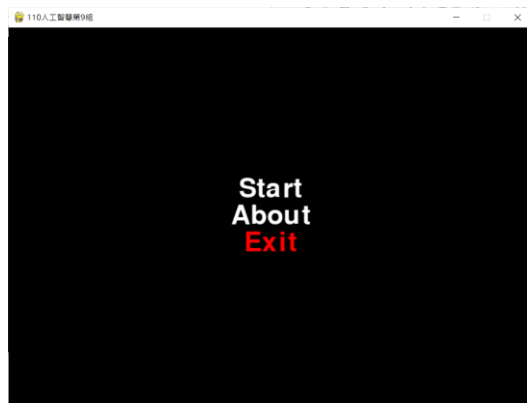


圖 7. 選單介面_Exit

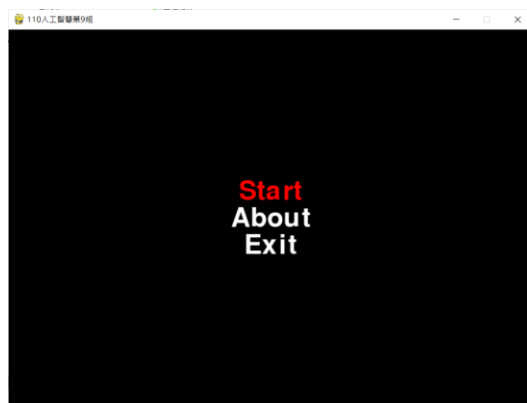


圖 8. 選單介面_Start



圖 9. 演算法執行結果

在訓練過程中，一開始吃豆人因訓練次數不夠導致 Score 不是最高，亦指吃豆人還無法完全掌握到最適合的路徑；而同樣的，在訓練第一次時，雖然可發現分數已到達最高點，但由怪物的位置可看出，怪物亦未訓練至最佳，未能緊跟吃豆人身後追逐吃豆人

(如圖 10、圖 11 及圖 12，此圖為訓練一次之結果)，當訓練完 50 次後，經由圖 13 可發現，吃豆人能夠自行避開怪物並吃完所有豆子達成目標，且怪物能不斷追逐吃豆人，而吃豆人的 Score 亦可達到巔峰。



圖 10. 訓練一次之初始位置



圖 11. 訓練一次之精靈結束位置



圖 12. 訓練一次之怪物結束位置



圖 13. 訓練五十次之結束位置

為何最終使用 Approximate Q-learning 而不使用 Q-learning，從圖 14 及圖 15 可發現，以 50 次作為相同訓練次數，再取 10 次作為測試，以 Approximate Q-learning 的成效會比 Q-learning 來得好，故本專題使用 Approximate Q-learning 作為吃豆人之演算法。

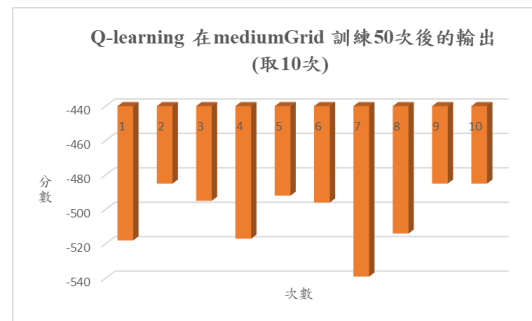


圖 14. Q-learning 訓練結果 (依 score 繪製)

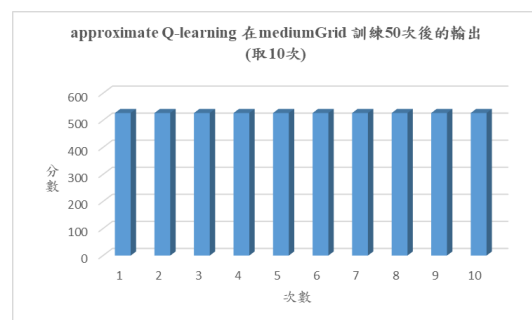


圖 15. Approximate Q-learning 訓練結果 (依 score 繪製)

5. 結論與建議

本專題核心目標在於吃豆人能自我學習避開怪物並將豆子全數吃光，而為了完成此目標，本專題利用強化式學習之 Approximate Q-learning 演算法，使吃豆人能以當下所得環境進行目標得學習。從實作成果當中可發現，本專題除了成功使得吃豆人可避開怪物並吃掉豆子，還利用強化式學習當中的 Approximate SARSA 演算法讓怪物也可自我學習，以達成追逐吃豆人之目標。我們冀希此專題後續能達成：(1)怪物數量增加；(2)吃豆人吃到大力丸時，能夠反追怪物，以獲得更高的 Reward；以及(3)地圖環境更為龐大。完成上述目標後，本專題發展可更為寬廣。

6. 參考文獻

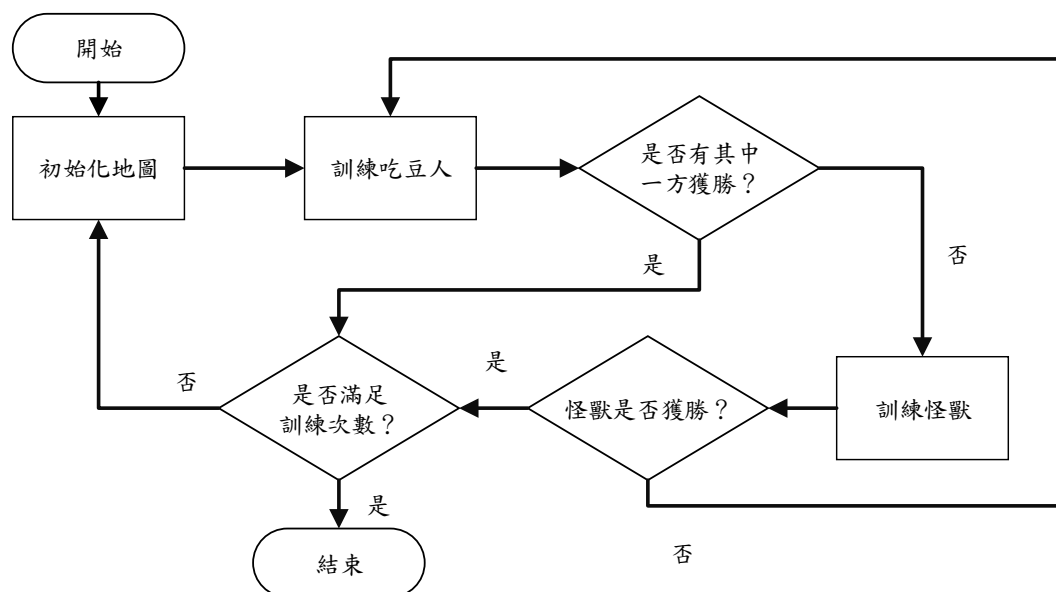
- [1] Jo, T. (2021). Machine Learning Foundations: Supervised, Unsupervised, and Advanced

- Learning. Springer Nature.
- [2] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
- [3] Stuart, R., & Norvig, P. (2010). Artificial intelligence: A modern approach 3rd Edition. Upper Saddle River, New Jeysey.
- [4] Xu, Z. X., Cao, L., Chen, X. L., Li, C. X., Zhang, Y. L., & Lai, J. (2018). Deep reinforcement learning with sarsa and q-learning: A hybrid approach. IEICE TRANSACTIONS on Information and Systems, 101(9), 2315-2322.
- [5] Alfakih, T., Hassan, M. M., Gumaiei, A., Savaglio, C., & Fortino, G. (2020). Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA. IEEE Access, 8, 54074-54084.
- [6] Mesut Yang, & Carl Qi. (2021). CS188|Summer 2021. Retrieved from <https://inst.eecs.berkeley.edu/~cs188/su21/>. (November 8,2021)

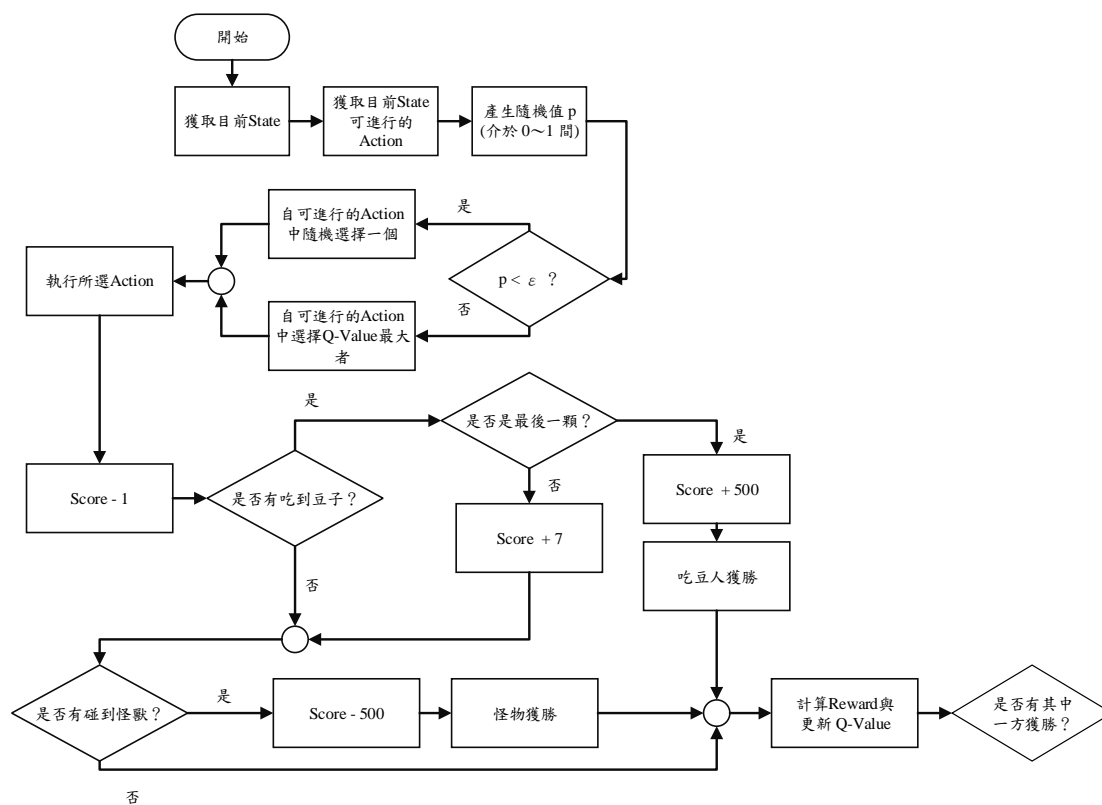
組員貢獻度(全部組員貢獻度合計 100%)

學號	姓名	主要工作項目	貢獻度
C107110133	陳亭禎	初始介面製作、吃豆人和怪物繪製、PPT 製作、論文撰寫。	33.33%
C107110106	郭冠宏	怪物演算法撰寫、PPT 製作、論文撰寫。	33.33%
C107104218	李宗穎	吃豆人演算法撰寫、PPT 製作、論文撰寫。	33.33%

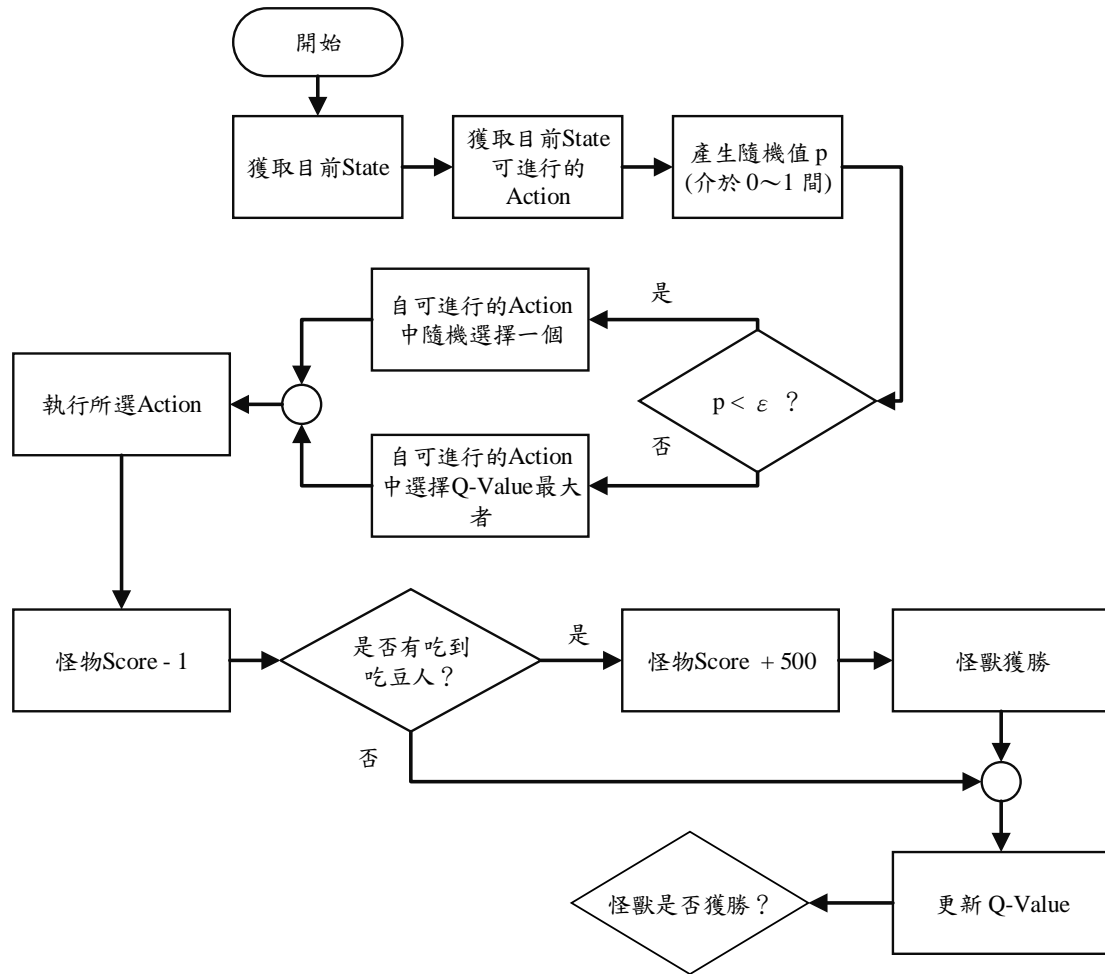
附錄一、流程圖



附圖 1-1. 演算法整體架構



附圖 1-2. 吃豆人演算法架構



附圖 1-3. 怪物演算法架構

附錄二、PPT 簡報

REINFORCEMENT LEARNING OF PACMAN FINAL PROJECT

陳亭禎、郭冠宏、李宗穎
TEAM09
電腦與通訊工程系

2022.01.10

CONTENTS

- BACKGROUND OR TREND
- MOTIVATION
- OBJECTIVE
- CHALLENGE
- POTENTIAL SOLUTIONS
- RESOURCE REQUIRED
- SCHEDULE
- DEMO TIME
- CONCLUSION
- REFERENCES

BACKGROUND OR TREND

- 發現問題:如何使電腦擁有自我學習的能力?
 - 於期中前所學:
 - ✓ Informed Search
 - ✓ UnInformed Search

還不算讓電腦擁有自我學習的能力

- ✓ Berkeley CS188 Pacman 程式中 Project3 Reinforcement learning 的部份當成我們期末專題的實作內容。

MOTIVATION

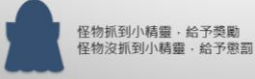
- 根據前頁的問題，說明解問題的方向
 - 近些年因人工智慧(ALPHAGO、DRIVERLESS CAR)應用的風行，許多人工智慧相關的應用也接連不斷的出現於生活中，而我們團隊在本次的期末專題採用強化學習的動機主要有兩大重點：



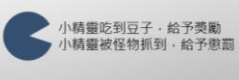
MOTIVATION

- 根據前頁的問題，說明解問題的方向
 - 近些年因人工智慧(ALPHAGO、DRIVERLESS CAR)應用的風行，許多人工智慧相關的應用也接連不斷的出現於生活中，而我們團隊在本次的期末專題採用強化學習的動機主要有兩大重點：

1. 利用強化學習作出行為(action)與環境(Environment)互動接收不同的反饋(獎勵、懲罰)來對未知環境做出最好(optimal)的決策。



怪物抓到小精靈，給予獎勵
怪物沒抓到小精靈，給予懲罰



小精靈吃到豆子，給予獎勵
小精靈被怪物抓到，給予懲罰

OBJECTIVE

- 根據動機，說明解問題的方式

2. 大家可能會問? 那為何不用Supervised learning? Unsupervised learning?

監督式學習---有特定答案讓電腦學習

身高	體重	性別
162	60	女
172	60	男
172	55	男

身高	體重	性別
152	48	?

非監督式學習---沒有特定答案讓電腦學習，從特徵中尋找關聯

數學	物理	國文	英文
76	89	36	42
88	92	89	70
52	35	25	40
27	15	82	79

➡ 數學好 可能 物理好
國文好 可能 英文好

OBJECTIVE

- 根據動機，說明解問題的方式

強化式學習---根據所處環境學習，無特定特徵，只設定所需達成目標，並建立獎勵機制以利學習



CHALLENGE

- 這類問題有什麼普遍的挑戰或解這問題方式的挑戰

- 如何定義獎勵機制?
 - ✓ 小精靈吃到豆子，給予獎勵—SCORE加7分。
 - ✓ 小精靈吃完豆子，給予獎勵—SCORE加500分。
 - ✓ 小精靈被怪物抓到，給予懲罰— SCORE減500分。
 - ✓ 怪物吃到小精靈，給予獎勵—怪物分數加500。
 - ✓ 怪物沒吃到小精靈，給予懲罰—怪物分數減1。
 - ✓ 每經過一個ACTION，SCORE會減1分。
- 達成目標後，如何判斷是不是最佳的?
 - ✓ 利用SCORE判斷，SCORE越高，訓練效果越好。
- 如何互換怪物以及小精靈的學習成果，並加以訓練?
 - ✓ 怪物和小精靈的學習是同步的，故無交換之可能。
- 是否好收斂?
 - ✓ 原先使用Q-LEARNING發現成效不彰，故使用APPROXIMATE Q-LEARNING。

POTENTIAL SOLUTIONS

- 你會採取的演算法、技術等等
 - 演算法
 - 小精靈---APPROXIMATE Q-LEARNING
 - 怪物--- APPROXIMATE SARSA
 - 繪圖技術
 - 小精靈、豆子---CIRCLE()函式繪製
 - 怪物---POLYGON()函式繪製
 - 初始畫面---PYGAME函式庫、LABEL()建立MENU

6

POTENTIAL SOLUTIONS

- Q-LEARNING / SARSA
 - 演算法方面我們將會使用Q-LEARNING 及SARSA來實作。
 - 首先兩演算法內部基礎概念來源於TD(TEMPORAL DIFFERENCE)-UPDATE RULE，由下圖所示：

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S',A') - Q(S,A))$$

藉由應用該公式，我們能透過程式在每一次的STATE-TRANSITION中進行Q-VALUE的更新，使得Q-VALUE能夠被逐漸訓練成在PACMAN遊戲中能表現出最好(OPTIMAL)成效的Q-VALUE。而實作過程中，我們也會應用到EPSILON GREEDY以機率決定AGENT在STATE上擁有隨機選擇ACTION的能力能夠來探索地圖世界。

7

POTENTIAL SOLUTIONS

- QLEARNING / SARSA
 - 對各別AGENT(怪物及小精靈的)演算法差異說明

Q-Learning Off-policy	Initialize $Q(s,a)$ arbitrarily Repeat (for each episode): Initialize s Repeat (for each step of episode): Choose a from s using policy derived from Q (e.g., ϵ -greedy) Take action a , observe r, s' $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$ $s = s'$ until s is terminal
Sarsa On-policy	Initialize $Q(s,a)$ arbitrarily Repeat (for each episode): Initialize s Choose a from s using policy derived from Q (e.g., ϵ -greedy) Repeat (for each step of episode): Take action a , observe r, s' Choose a' from s' using policy derived from Q (e.g., ϵ -greedy) $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)]$ $s = s', a = a'$ until s is terminal

8

POTENTIAL SOLUTIONS

- Q-LEARNING與APPROXIMATE Q-LEARNING差異
 1. 從Q-FUNCTION原先以Q-TABLE的查表法(LOOK UP TABLE)的方式換成以權重(WEIGHT)*特徵(FEATURES)的方式來進行APPROXIMATION。

$$Q(s,a) = \sum_{k=1}^n f_k(s,a) \omega_k$$

2. 透過將Q-FUNCTION轉變成FUNCTION APPROXIMATION的方式且透過將更新的規則改以更新權重的方式來進行計算，能夠有效地強化小精靈在行為決策上的優化。

$$w_i \leftarrow w_i + \alpha \cdot \text{difference} \cdot f_i(s,a)$$

$$\text{difference} = \left(r + \gamma \max_{a'} Q(s',a') \right) - Q(s,a)$$

3. 而在本次期末專題報告中我們組設定了四個特徵 (豆子會不會被吃掉、距離下一個豆子的距離遠近、是否即將撞上怪物、是否與怪物只差一步的距離) 來進行訓練後，使得小精靈在4X6MEDIUMGRID地圖的表現有了更顯著的提升。

9

POTENTIAL SOLUTIONS

- Q-LEARNING與APPROXIMATE Q-LEARNING差異
 1. 從Q-FUNCTION原先以Q-TABLE的查表法(LOOK UP TABLE)的方式換成以權重(WEIGHT)*特徵(FEATURES)的方式來進行APPROXIMATION。

$$Q(s,a) = \sum_{k=1}^n f_k(s,a) \omega_k$$

2. 透過將Q-FUNCTION轉變成FUNCTION APPROXIMATION的方式且透過將更新的規則改以更新權重的方式來進行計算，能夠有效地強化小精靈在行為決策上的優化。

$$w_i \leftarrow w_i + \alpha \cdot \text{difference} \cdot f_i(s,a)$$

$$\text{difference} = \left(r + \gamma Q(s',a') \right) - Q(s,a)$$

4. 而在怪獸的部分，我們設定了兩個特徵，分別為：怪獸是否吃到吃豆人、怪獸與吃豆人的距離，藉此提升怪獸的效能。

9

POTENTIAL SOLUTIONS

- 小精靈與怪物之STATE&ACTION

小精靈

input	state	(x,y)				
	action	上	下	左	右	停止
output	state	(x,y+1)	(x,y-1)	(x-1,y)	(x+1,y)	(x,y)

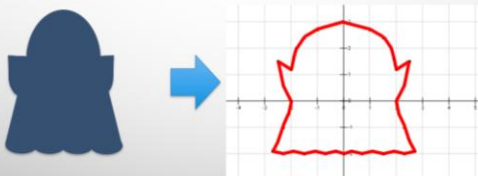
怪物

input	state	(x,y)				
	action	上	下	左	右	停止
output	state	(x,y+1)	(x,y-1)	(x-1,y)	(x+1,y)	(x,y)

10

POTENTIAL SOLUTIONS

- 繪圖技術
 - 怪物：使用座標點建構出圖形，使用PYTHON內建POLYGON()函式描繪。



11

POTENTIAL SOLUTIONS

- 繪圖技術
 - 小精靈：使用PYTHON內建CIRCLE()函式描繪。
 - 豆子：使用PYTHON內建CIRCLE()函式描繪。
 - 初始畫面：利用PYGAME函式庫建立文字選單。
 - 使用LABEL()建立文字---開始、結束、關於，各項選擇分別有各自的INDEX。
 - 利用PYGAME來不斷偵測“上鍵”、“下鍵”以及“ESC”之操作。

Index狀態	選單項目	對應結果
0	Start	呼叫演算法
1	About	呼叫顯示訊息頁面
2	Exit	停止所有遊戲

12

RESOURCE REQUIRED

- 介紹專題需要的軟硬體設備和開發工具
 - 軟體：PYTHON3.6
 - 硬體：INTEL(R) CORE (TM) I7-2600CPU 3.4GHZ
- 人員的工作分配
 - 演算法設計
 - 郭冠宏、李宗穎
 - 動畫圖畫設計
 - 陳亭禎
 - 成果簡報製作
 - 陳亭禎、郭冠宏、李宗穎

13

SCHEDULE

- 專題的規畫時程

	11/08	11/15	11/22	11/29
陳亭禎	對於所分配程式進行了解與討論			針對怪物、小精靈與初始選單進行撰寫與修正
郭冠宏				針對怪物演算法進行撰寫與修正
李宗穎				針對小精靈演算法進行撰寫與修正

14

SCHEDULE

- 專題的規畫時程

	12/06	12/13	12/20	12/27	01/03
陳亭禎	針對怪物、小精靈與初始選單進行撰寫與修正	串接各個程式與修正			期末報告
郭冠宏	針對怪物演算法進行撰寫與修正				
李宗穎	針對小精靈演算法進行撰寫與修正				
			報告撰寫及程式最後確認(最後期限)		

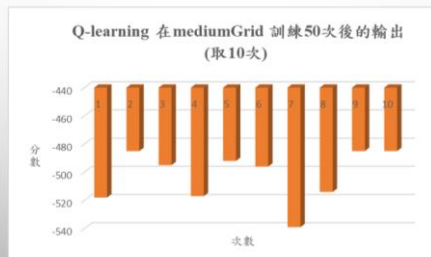
15

DEMO TIME

16

DEMO TIME

- Q-LEARNING/APPROXIMATE Q-LEARNING訓練結果



17

DEMO TIME

- Q-LEARNING/APPROXIMATE Q-LEARNING訓練結果



17

CONCLUSION

- 本專題核心目標在於吃豆人能自我學習避開怪物並將豆子全數吃光，而為了完成此目標，本專題利用強化式學習之 APPROXIMATE Q-LEARNING 演算法，使吃豆人能以當下所得環境進行目標學習。從實作成果當中可發現，本專題除了成功使得吃豆人可避開怪物並吃掉豆子，還利用強化式學習當中的 SARSA 演算法讓怪物也可自我學習，以達成追逐吃豆人之目標。
- 我們冀希此專題後繼續能達成：
 - 怪物數量增加；
 - 吃豆人吃到大力丸時，能夠反追怪物，以獲得更高的 REWARD；
 - 地圖環境更為龐大。
 完成上述目標後，本專題發展可更為寬廣。

18

REFERENCES

- JO, T. (2021). MACHINE LEARNING FOUNDATIONS: SUPERVISED, UNSUPERVISED, AND ADVANCED LEARNING. SPRINGER NATURE.
- SUTTON, R. S., & BARTO, A. G. (2018). REINFORCEMENT LEARNING: AN INTRODUCTION. MIT PRESS.
- STUART, R., & NORVIG, P. (2010). ARTIFICIAL INTELLIGENCE: A MODERN APPROACH 3RD EDITION. UPPER SADDLE RIVER, NEW JERSEY.
- XU, Z. X., CAO, L., CHEN, X. L., LI, C. X., ZHANG, Y. L., & LAI, J. (2018). DEEP REINFORCEMENT LEARNING WITH SARSA AND Q-LEARNING: A HYBRID APPROACH. IEICE TRANSACTIONS ON INFORMATION AND SYSTEMS, 101(9), 2315-2322.
- ALFAKIH, T., HASSAN, M. M., GUMAEI, A., SAVAGLIO, C., & FORTINO, G. (2020). TASK OFFLOADING AND RESOURCE ALLOCATION FOR MOBILE EDGE COMPUTING BY DEEP REINFORCEMENT LEARNING BASED ON SARSA. IEEE ACCESS, 8, 54074-54084.
- MESUT YANG, & CARL QI. (2021). CS188/SUMMER 2021. RETRIEVED FROM [HTTPS://INST.EECS.BERKELEY.EDU/~CS188/SU21/](https://inst.eecs.berkeley.edu/~CS188/SU21/). (NOVEMBER 8, 2021)

19

The background of the slide is a light gray gradient with several realistic water droplets of various sizes scattered across it, primarily concentrated in the top-left and bottom-right corners.

REINFORCEMENT LEARNING OF PACMAN FINAL PROJECT

陳亭禎、郭冠宏、李宗穎

TEAM09

電腦與通訊工程系

2022.01.10