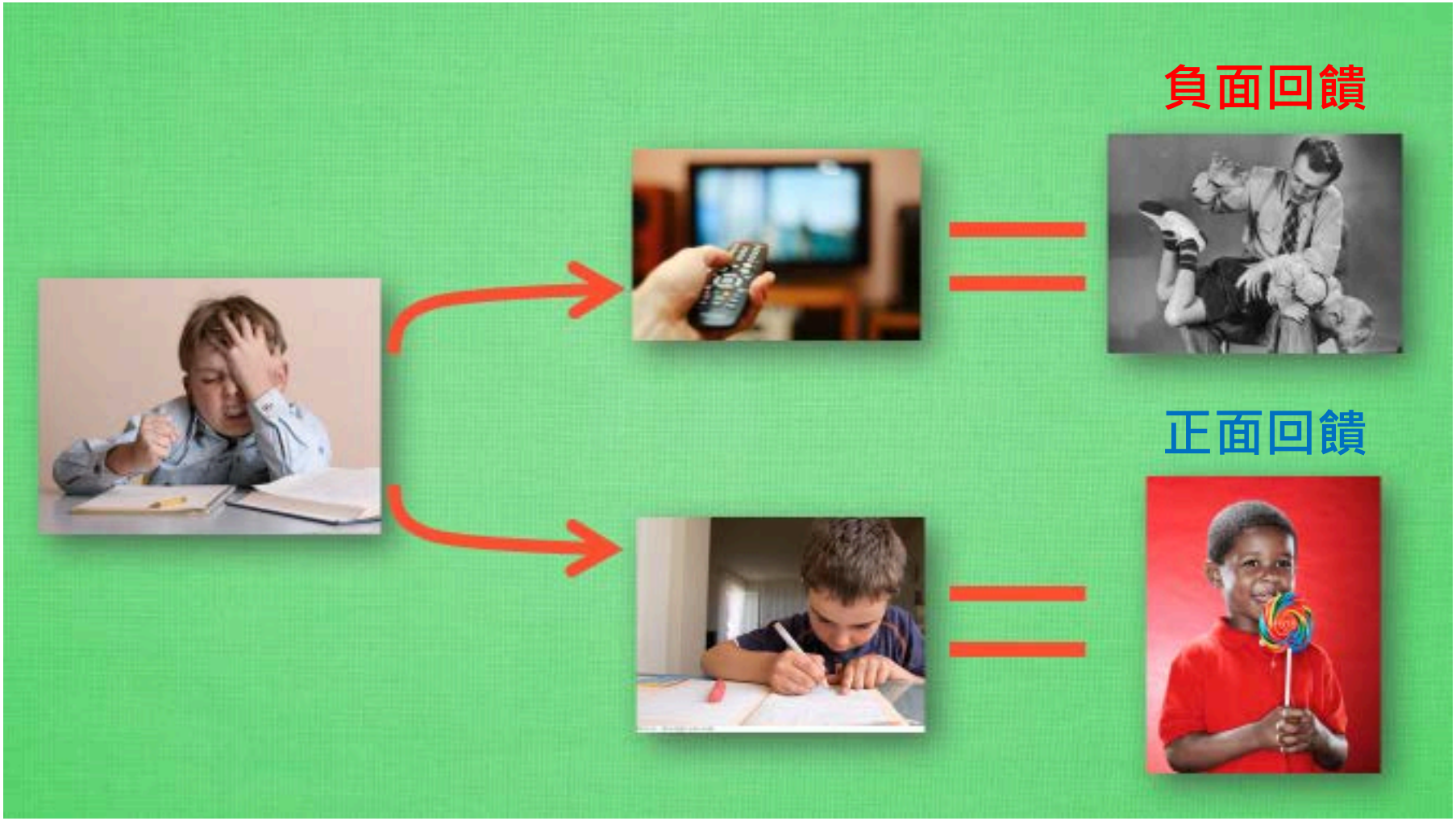
IIE Project 3
RL: Q Learning 以老鼠走迷宮為例及其應用介紹

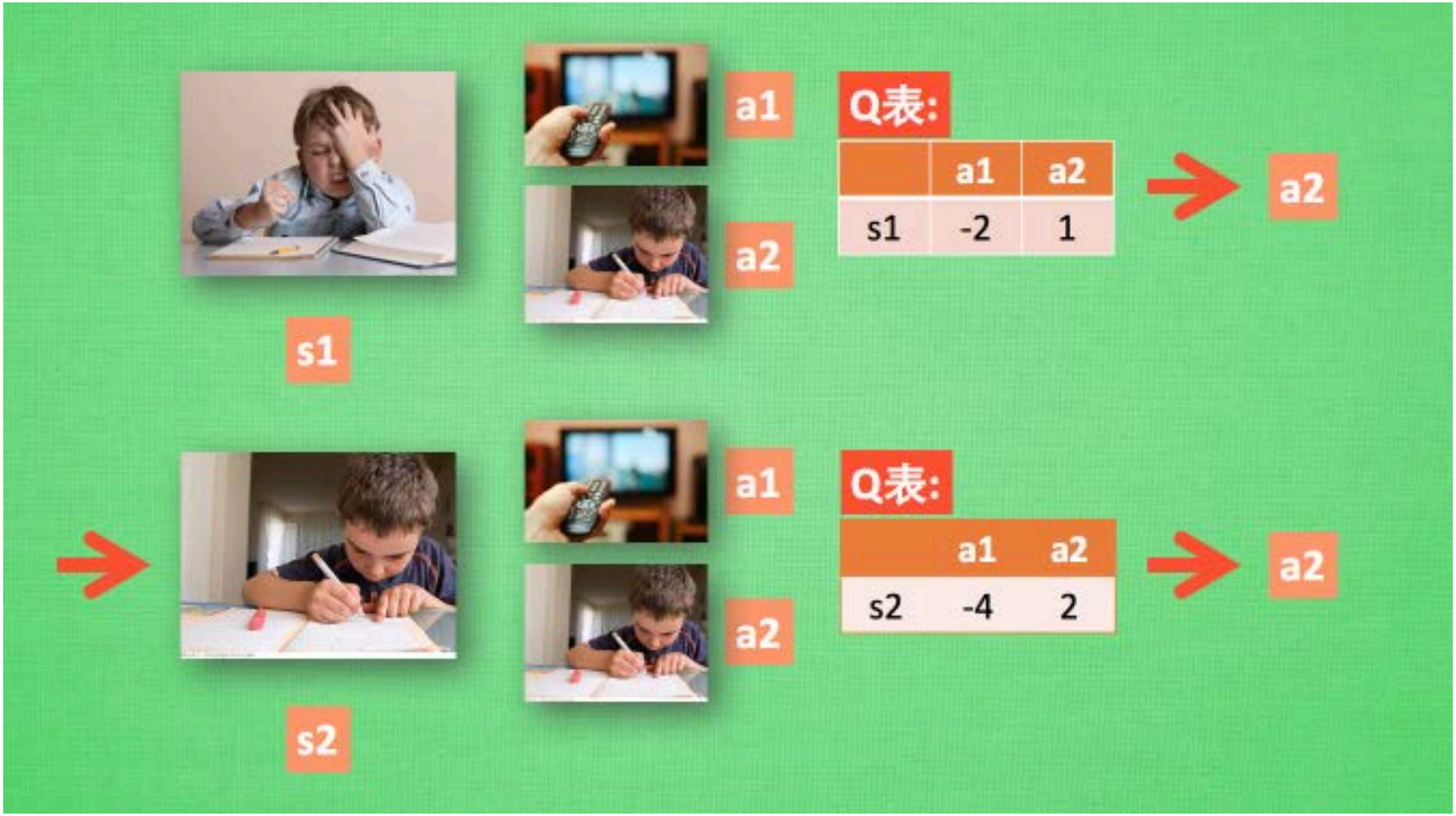
許家銘

2020/01/02

Reinforcement Learning



Q Learning



Q Learning 核心公式

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'

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$$

$s \leftarrow s'$;

until s is terminal

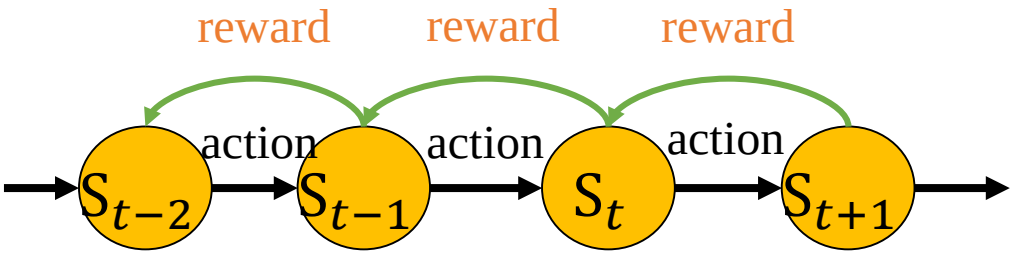
Q(s2)最大估计

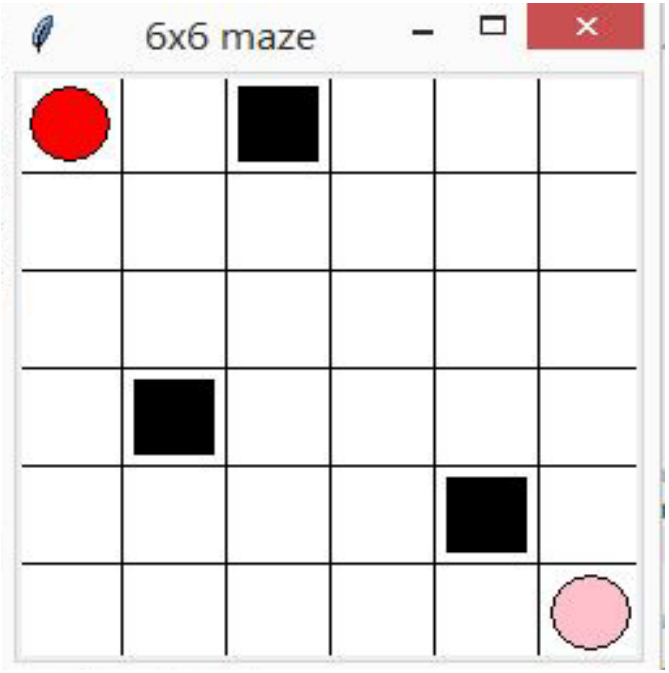
Q(s1, a2) 现实

Q(s1, a2) 估计

情境假設說明

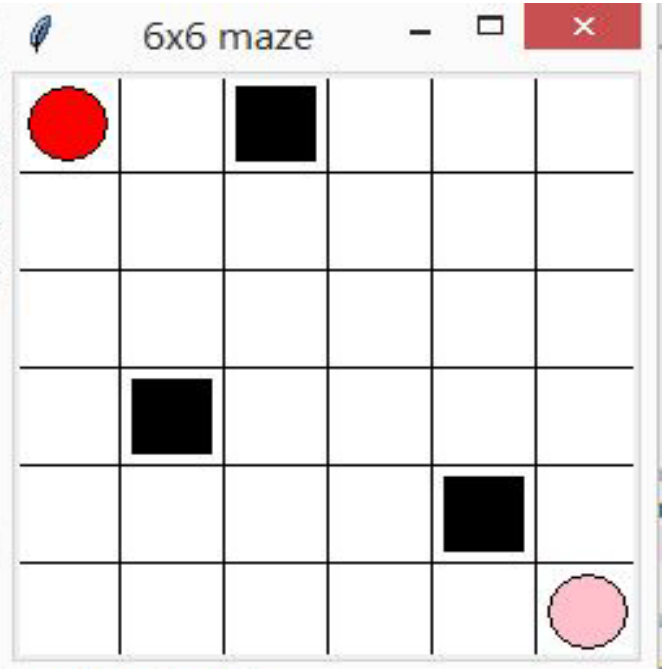
- 本次將利用Q Learning來訓練agent走迷宮。
- 迷宮為6*6的正方形，其中有三個黑色障礙物，若agent經過障礙物，將給予懲罰值(reward= -100)，走到終點則給予獎勵值(reward=100)，目標為訓練agent自行避開障礙點走到終點。

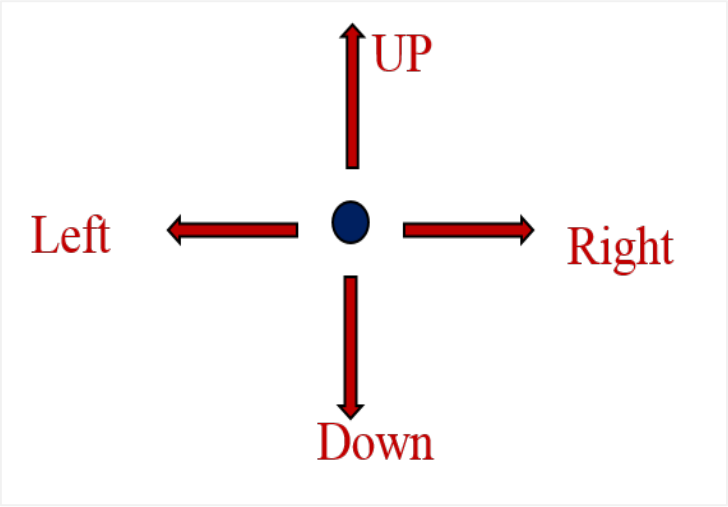




Q table

- State : 6*6的格子 (共36個state)
- Action : 上、下、右、左 (共4個action)





Reward function

- The immediately reward after RL take actions:

$$S_{t+1} = \{T_X, T_Y\}$$

- +100, If Agent arrived at the target

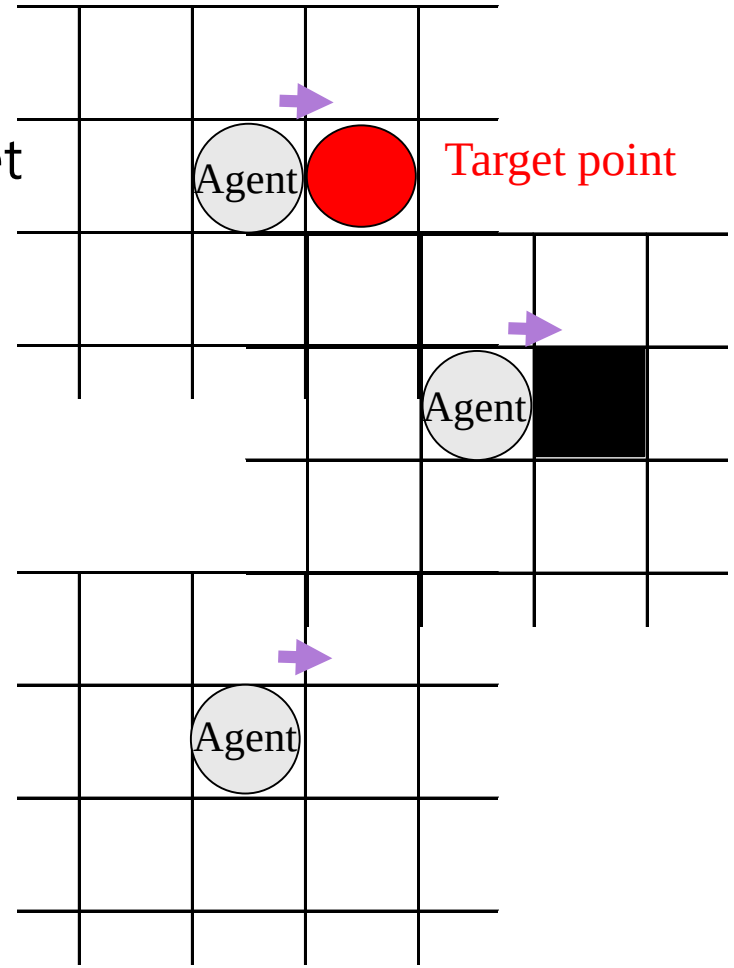
$$\{T_X, T_Y\} = \{C_X, C_Y\}$$

- -100, if Agent arrived at the hell

$$\{T_X, T_Y\} = \{H_X, H_Y\}$$

- 0, otherwise

$$\{T_X, T_Y\} \neq \{C_X, C_Y\}$$



Q table的建構以及更新

- Q table為 36*4的矩陣

State	Action 1	Action 2	Action3	Action4
State 1	Q(1, 1)	Q(1, 2)	Q(1, 3)	Q(1, 4)
…	…	…	…	…
State 35	Q(35, 1)	Q(35, 2)	Q(35, 3)	Q(35, 4)
State 36	Q(36, 1)	Q(36, 2)	Q(36, 3)	Q(36, 4)

Hint: Q table的建構為當經過未走過的格子時，才會將此行加入Q table，故有可能不為36行，且行的順序為agent走過的順序

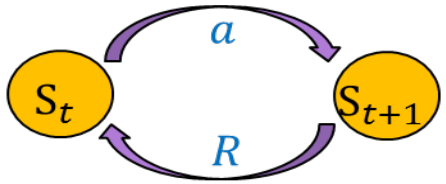
- Q value update:

Learned value

$$Q(S_t, a) \leftarrow Q(S_t, a) + \alpha[R + \gamma max_{a'} Q(S_{t+1}, a') - Q(S_t, a)]$$

Learning Rate

Reward



Q table的建構

$\alpha = 0.1$；$\gamma = 0.9$

1.	上	下	右	左	2.	上	下	右	左	3.	上	下	右	左
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	-10	0	0	0		-10	0	0	0		-19	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	-10	0		0	0	-10	0		0	0	-19	0
	0	0	0	0		0	0	0	0		-10	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	-10
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	-10	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	-10	0	0		0	-10	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	10	0	0		0	10	0	0		0	10	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
						0	0	0	0		0	0	-10	0
						0	0	0	0		0	0	0	0
						0	0	0	0		0	0	0.9	0
						0	0	10	0		0	0	10	0
											0	0	0	0
											-10	0	0	0
											0	0	0	0
											0	0	-10	0
											0	0	0	0
											0	0	0	0

Q table的建構

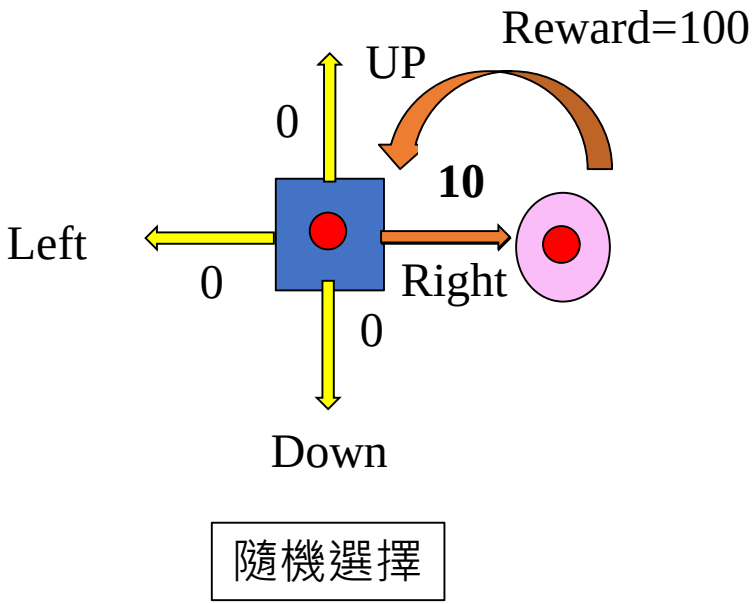
$\alpha = 0.1$；$\gamma = 0.9$

4.	上	下	右	左	5.	上	下	右	左	6.	上	下	右	左
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	-19	0	0	0		-19	0	0	0		-19	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	-19	0		0	0	-19	0		0	0	-19	0
	-10	0	0	0		-10	0	0	0		-10	0	0	0
	0	0	0	-10		0	0	0	-19		0	0	0	-19
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	-10	0	0		0	-10	0	0		0	-10	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	-10	0	0		0	-10	0	0		0	-10	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	19	0	0		0	19	0	0		0	19	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	-10	0		0	0	-10	0		0	0	-10	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	2.52	0		0	0	4.707	0		0	0	7.331	0
	0	0	19	0		0	0	27.1	0		0	0	34.39	0
	0	0	0	0		0	0	0	0		0	0	0	0
	-10	0	0	0		-10	0	0	0		-10	0	0	0
	0	0	0	0		0	0	0	0		0	0	0	0
	0	0	-10	0		0	0	-10	0		0	0	-10	0
	0	0	0	0		0	0	0	0		0	0.00729	0	0
	0	0	0	-10		0	0	0	-10		0	0	0	-10

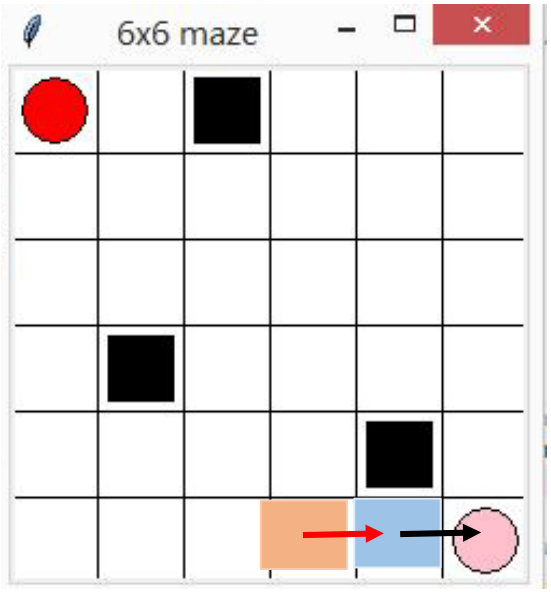
Q table的建構：正向回饋(1)

$\alpha = 0.1$；$\gamma = 0.9$

- 第一次從藍色格子向右抵達終點時，得到reward值100，經由Q table轉換，$\mathbf{Q(St, 右) = 0 + 0.1[100] = 10}$



隨機選擇



Up.	down	right	left
0	0	0	0

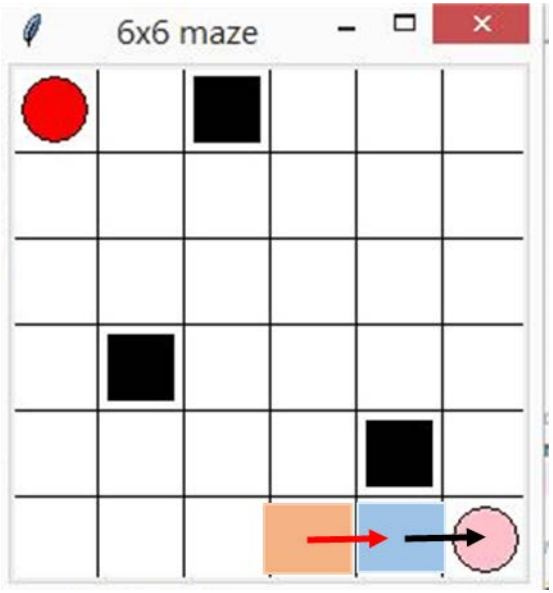
↓

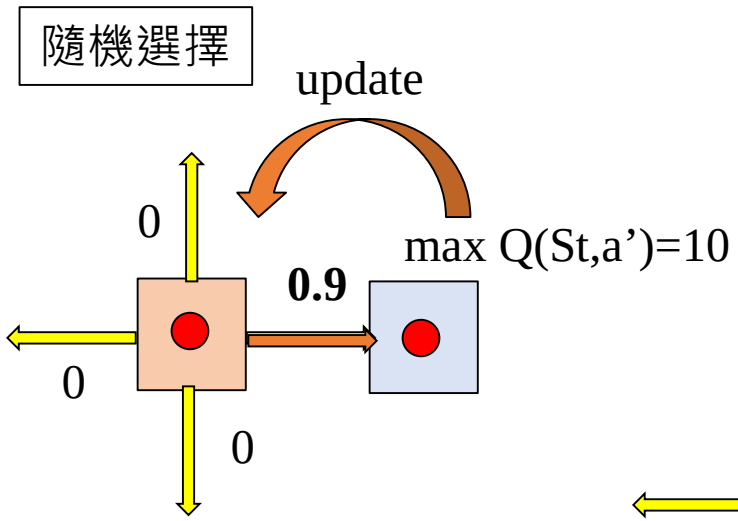
0	0	10	0

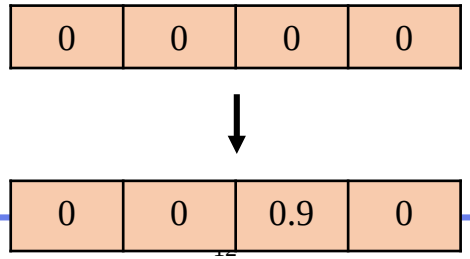
Q table的建構：正向回饋(2)

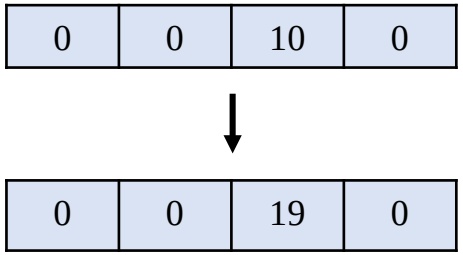
α =0.1；γ = 0.9

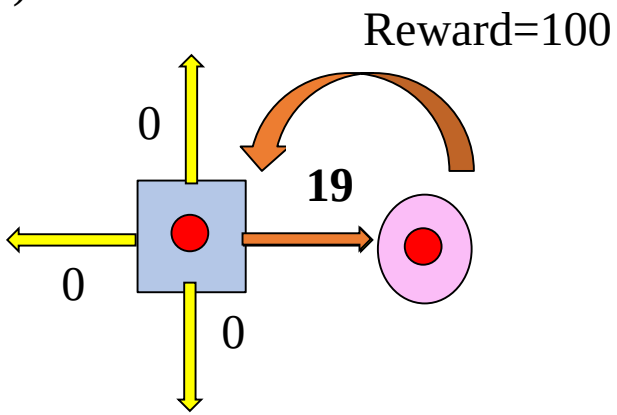
- 第二次經過藍色格子向右抵達終點時，在橘色格子時，若選擇action為右，其Q table值會轉換成Q(St, 右) = 0 + 0.1[0 + 0.9 * 10 − 0] = **0. 9**。
- 接著有0.1的機率會隨機選擇方向，有0.9的機率會選擇往右的方向(往右的Q table值為行中最大)，將再次得到reward值100，藍色格子的Q table值會轉換成Q(St, 右) = 10 + 0.1[100 − 10] = **19**。









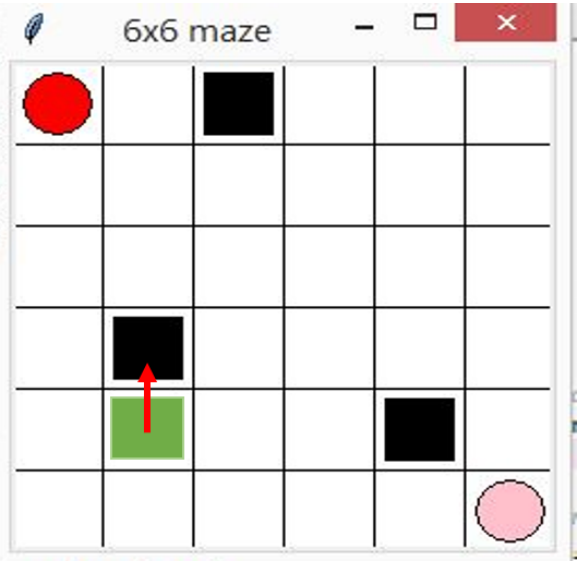


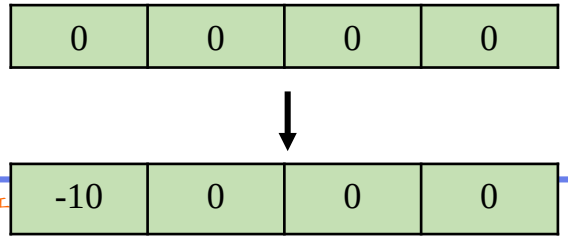
0.9的機率會選擇右

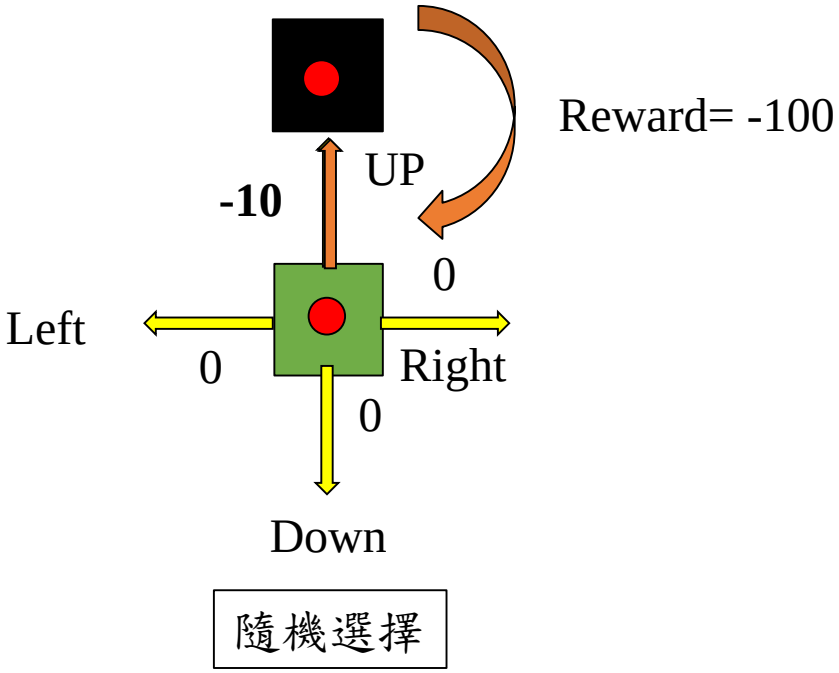
Q table的建構：負向回饋

α =0.1；γ = 0.9

- 第一次由綠色格子向上踏入障礙物時，給予懲罰值-100，其Q table值會轉換成Q(St, 上) = 0 + 0.1[−100 + 0.9 ∗ 0 − 0] = −10。
- 而第二次由綠色格子要選擇下一步時，有0.1的機率會隨機選擇方向(因此還是有可能會再次經過障礙點)，而有0.9的機率會選擇Q table此行中最大值的方向，故向上的方向並不會被選取，agent經過訓練後會避開障礙點。







參數設定

$Q(s1) = r2 + \gamma\, Q(s2) = r2 + \gamma\,[r3 + \gamma Q(s3)] = r2 + \gamma[r3 + \gamma[r4 + \gamma Q(s4)]] = \ldots$

$Q(s1) = r2 + \gamma\, r3 + \gamma^2\, r4 + \gamma^3\, r5 + \gamma^4\, r6 + \ldots$

$\gamma = 1$

$Q(s1) = r2 + 1{*}r3 + 1{*}\, r4 + 1{*}\, r5 + 1{*}\, r6 + \ldots$

$\gamma = (0 \sim 1)$

$Q(s1) = r2 + \gamma\, r3 + \gamma^2\, r4 + \gamma^3\, r5 + \gamma^4\, r6 + \ldots$

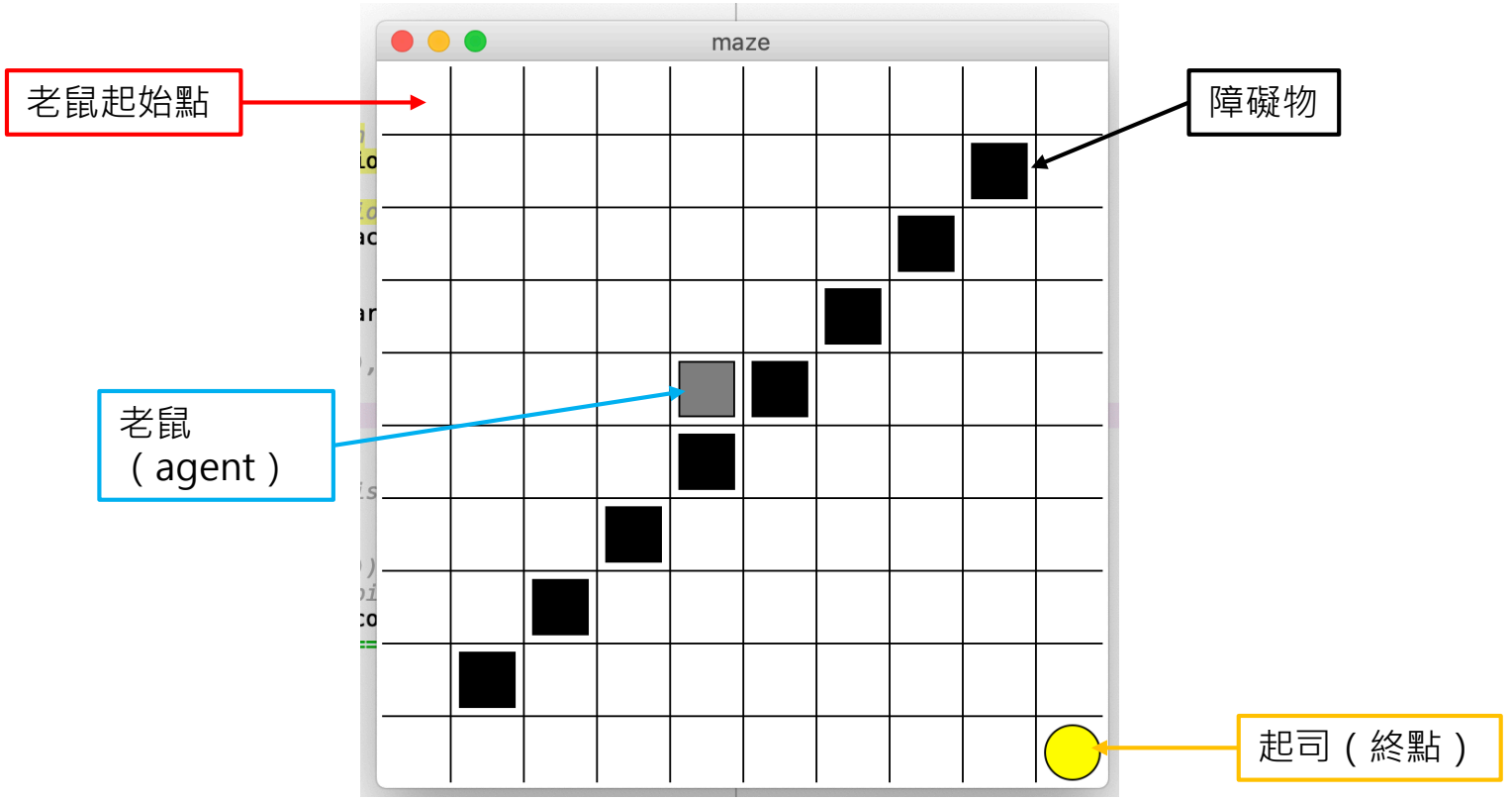
$\gamma = 0$

$Q(s1) = r2$

不同參數組合實驗

不同參數組合對total step之比較

- 為了探討不同學習率與不同衰減值之間的組合對每次迭代之次數是否有影響，故使用了新的迷宮設計來實驗並推論。
- 新迷宮放大為10*10，八個障礙物擋在其中，每種實驗參數組合各運行100次episode。此次實驗目標主要改變學習率與衰減值兩種參數之三種參數組合並比較老鼠行走total step之差異

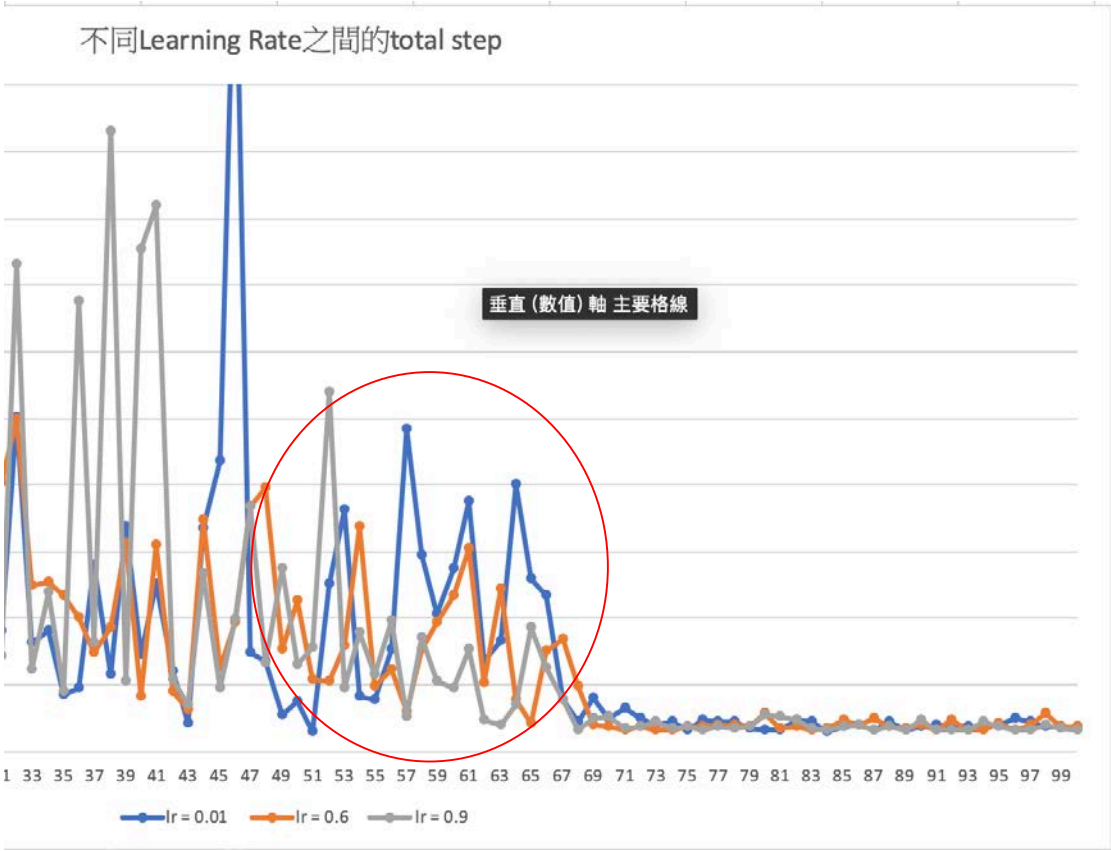


實驗條件

- 參數假設
  - Episode皆固定=100次
  - Greedy rate(ε):本次實驗皆固定=0.9
  - Learning rate(α): 三種不同學習率0.01, 0.6, 0.9
  - Discount rate(γ): 三種不同衰減率0.01, 0.6, 0.9

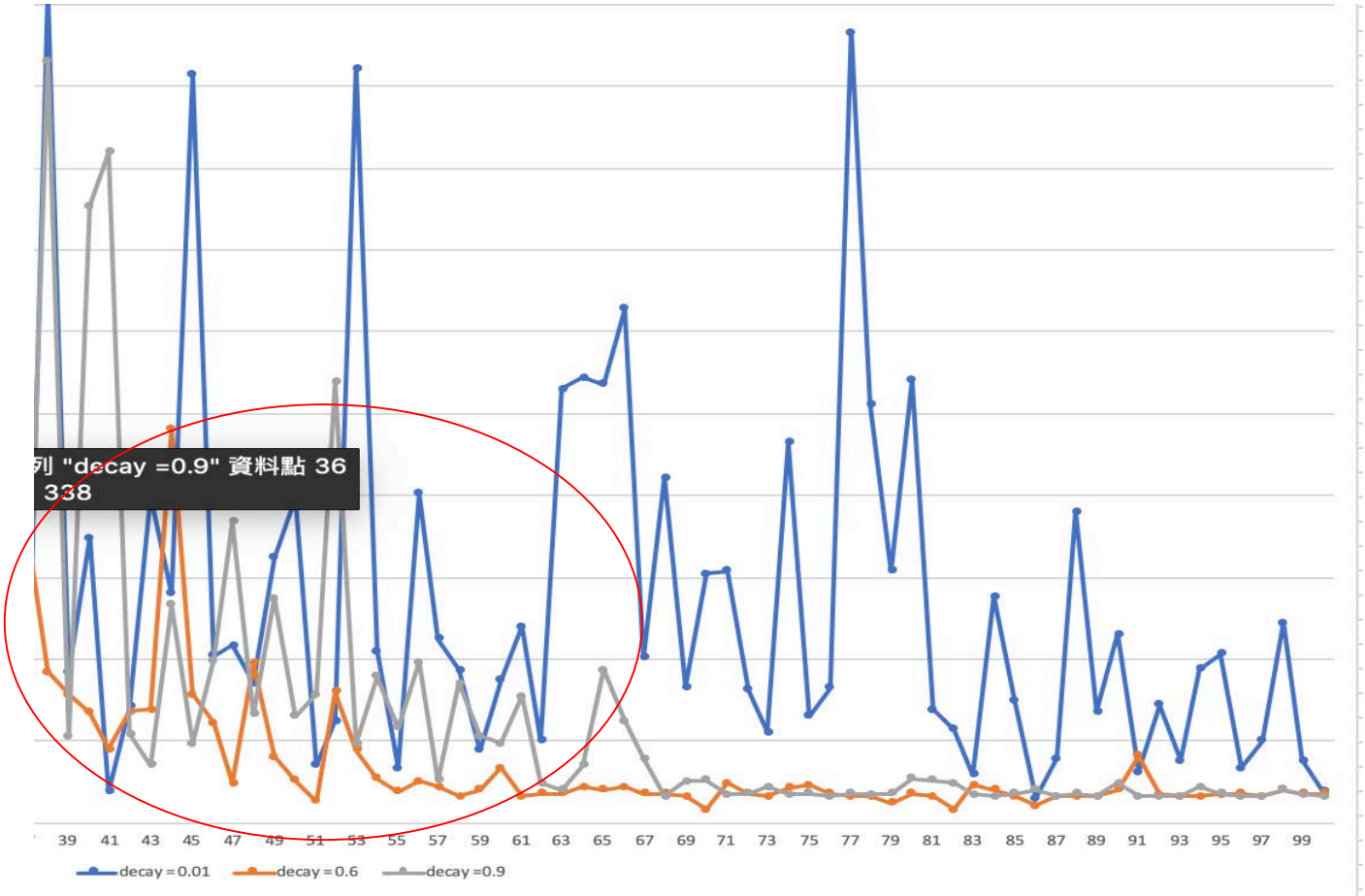
- 目標
  - 改變學習率與衰減值兩種參數之三種參數組合並比較agent(老鼠)行走到終點(起司)t所耗費total step之比較

固定γ, ε，α=0.01, 0.6, 0.9之total step比較



α代表學習率，學習率會影響模型收斂到局部可行解甚至最佳解（total step最小且未碰撞障礙物）的速度。
雖然使用低學習率，可以確保我們不會錯過任何局部最佳；但也有可能意味著我們將耗費很久的時間來收斂。

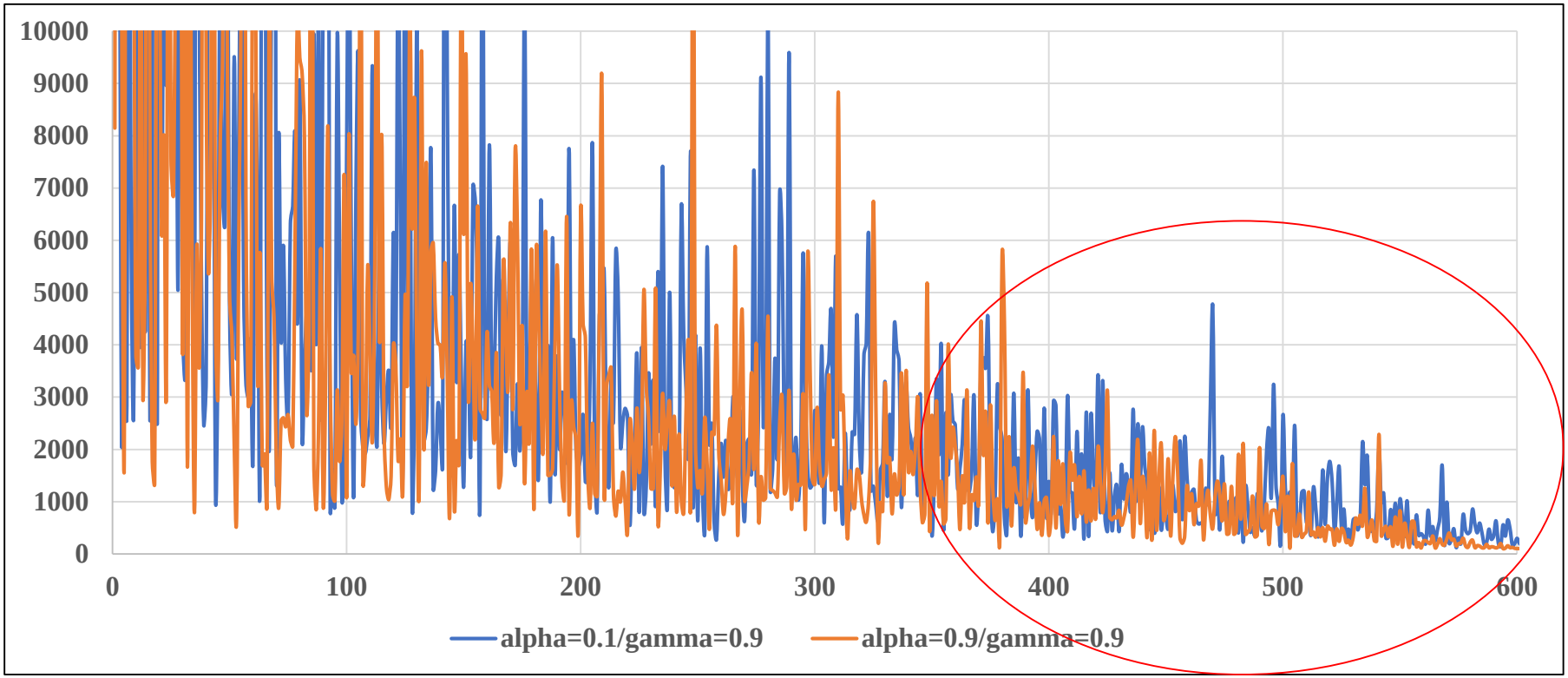
固定α, ε， γ =0.01、0.6以及0.9比較



γ代表考慮future reward的衰減值，愈接近1代表越看重future reward(長期)，愈接近0代表愈看重當前狀況。

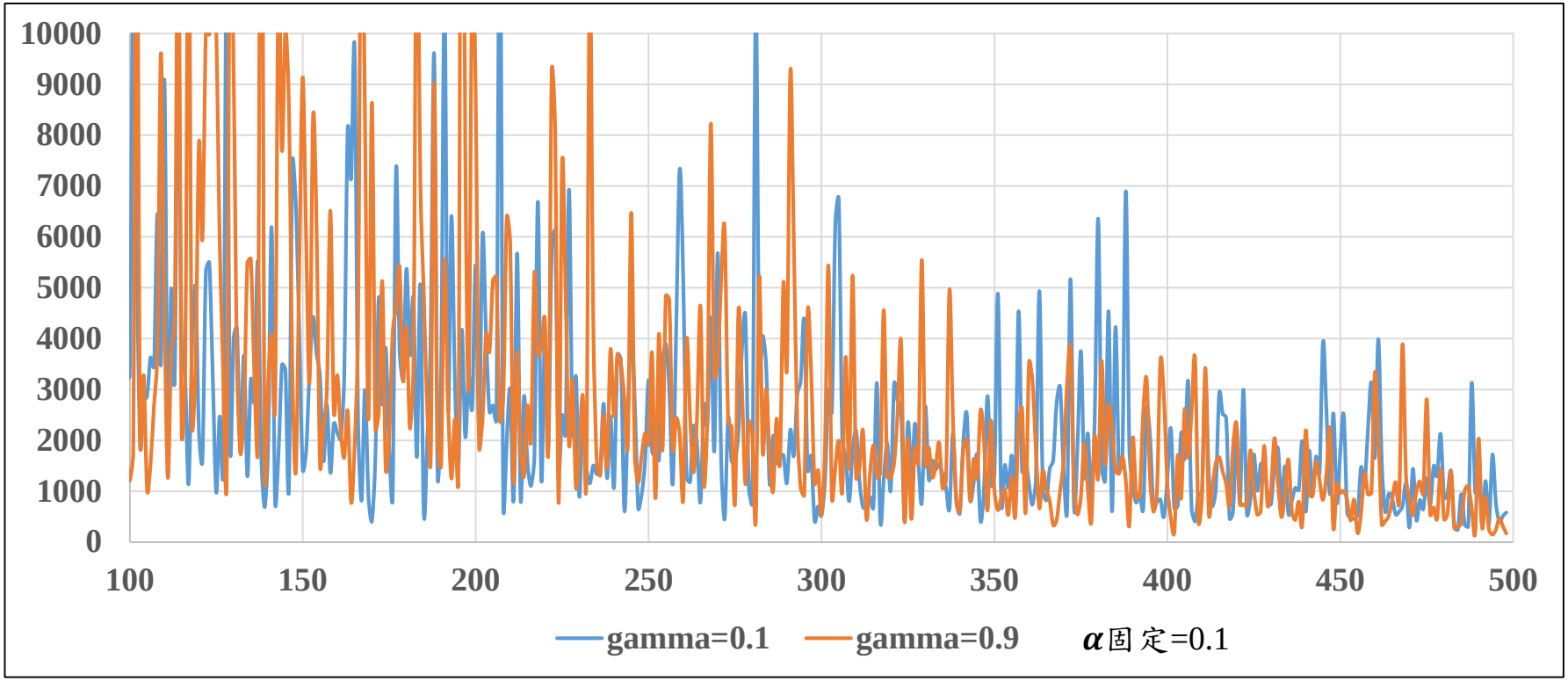
由上圖可見，在γ =0.1時，Step次數降得最慢，是因為只注重當前的期望reward值 ；γ越大，越看重未來$reward$，能更快收斂至區域可行解。

30*30 maze with 20 obstacles
兩種learning rate(α)比較



α越高，$total\ step$越快收斂至一區域可行解甚至最佳。
α低，可以確保不會錯過任何局部最佳值；但也意味著將耗費更多的時間來收斂至最佳解。

30*30 maze with 20 obstacles
兩種discount rate(γ)之比較



γ越大越在乎未來的期望reward，拿到的reward衰減越少，因此能更快收斂

γ小則只注重當下的reward，原地兜圈子、碰壁的比例較高，較慢才收斂

應用

- Inventory Management
  - reduce transit time for stocking as well as retrieving products in the warehouse for optimizing space utilization and warehouse operations.
- Delivery Management
  - Reinforcement learning is used to solve the problem of Delivery Vehicle Routing. Q-learning is used to serve appropriate customers with just a few vehicles.
  - AGV
- Finance Sector
  - Pit.AI is at the forefront leveraging reinforcement learning for evaluating trading strategies. It is turning out to be a robust tool for training systems to optimize financial objectives
- Gaming
  - ATARI- Space Invader
  - Alpha Go

END