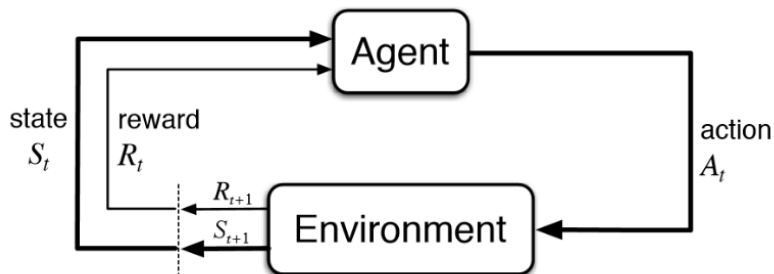


強化式學習(Reinforcement Learning)

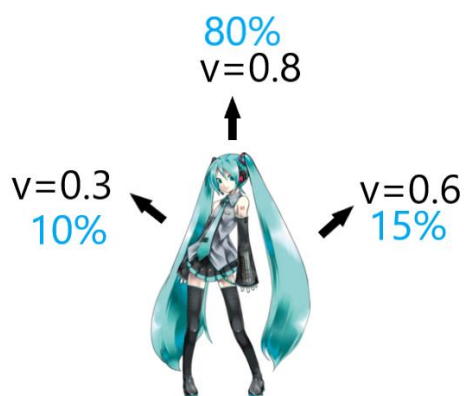
強化式學習是指機器與環境互動過程中，人類不必直接提供解決方案，而是透過電腦不斷嘗試中學習。在訓練中以獎勵機制來強化機器行為，找到最佳的行動策略。




圖：馬可夫決策過程示意圖

- Environment(環境): 根據 Action 給予對應的 Reward
- Agent(代理人): 透過 Action 與 Environment 互動的角色
- Reward(獎勵): Environment 給 Agent 所做 Action 的獎勵懲罰
- State(狀態): Agent 身處的狀態
- Action(行動): Agent 依 Environment 給予的 Reward 做出的決策

不同於演算法中的「貪婪法(Greedy method)」，在每個決策點時只會選擇當下情況的最佳決策。「馬可夫決策過程(Markov decision process)」是決策者在決策過程時，除了選擇當下最佳決策外，也會隨機選擇其他決策。在最佳化問題中會有一個「最佳解(Optimum)」與其他「次佳解(Suboptimal)」的結果，而貪婪法往往只能找到次佳解，但馬可夫決策過程卻可以找到最佳解。



圖：馬可夫決策過程隨機決策概念

			寶藏 (+1)
	懲罰 (-1)		
			


圖：強化式學習中代理人所處環境

假設有一個 初音(Agent) 在 迷宮(Environment) 中，初音依照當下路徑(State) 來選擇走法。初音的目標是要 得到(Action) 寶藏(Reward)並且避開(Action) 懲罰(Reward)，而初音會挑選最短且獎勵最多的路徑來完成目標。


對代理人來說，一個優秀的行動，需要依靠以下兩個因素：

1. 做一個行動，要能夠立刻獲得獎勵。
2. 做一個行動後，轉到下一個狀態時，可能在未來獲得獎勵

環境給予代理人的影響稱為「獎勵(Reward)」，獎勵可能是正值或負值，依照正負大小值來影響代理人的決策。代理人從某個特定狀態開始，預期未來可以取得的獎勵是多少，選擇最大的「值(Value)」作為決策。

	Value = +0.9	
Value = +0.3		Value = +0.6
	Value = -0.3	

圖：值(Value)概念圖


			寶藏 (+1)
	懲罰 (-1)		
			

圖：獎勵(Reward)概念圖

貝爾曼方程(Bellman Equation)

$$V(S) = \max(R(S,a) + \gamma V(S'))$$

- $V(S)$: 初始狀態
- $R(S,a)$: 在當前狀態中，選擇行動的獎勵
- $V(S')$: 根據行動，轉移至下一個新狀態
- γ : 「衰退常數」是控制下個狀態的值對當前狀態值的影響力，通常是用來表示 Reward 對狀態的影響力
 - γ 數值介於 0 到 1 之間
 - γ 數值越大時，代理人更加重視未來獲得的獎勵
 - γ 數值越小時，代理人只在乎目前可獲得的獎勵

$V=0.81_{(3)}$ ↗	$V=0.9_{(2)}$ →	$V=1_{(1)}$ →	寶藏 (+1)
$V=0.73_{(4)}$ ↑	懲罰 (-1)	$V=0.9_{(2)}$	$V=1_{(1)}$
$V=0.66_{(5)}$ ↑	$V=0.73_{(4)}$	$V=0.81_{(3)}$	$V=0.9_{(2)}$
	$V=0.66_{(5)}$	$V=0.73_{(4)}$	$V=0.81_{(3)}$

圖：路徑上各狀態的值($\gamma = 0.9$)

1. 編號(1):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 1 + 0.9 \times 0 = 1$$

寶藏位置前一格值定為 1，因為下一步就已經取得寶藏，所以沒有下一個狀態，故由公式可知此位置的值為 1。

2. 編號(2):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.9 \times 1 = 0.9$$

當前狀態無法獲得寶藏 $R(S, a)$ 為0，將下一個狀態的值 1 乘上 γ ，再相加得出此位置的值為0.9。

3. 編號(3):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.9 \times 0.9 = 0.81$$

當前狀態無法獲得寶藏 $R(S, a)$ 為0，將下一個狀態的值0.9乘上 γ ，再相加得出此位置的值為0.81。

4. 編號(4):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.81 \times 0.9 = 0.73$$

當前狀態無法獲得寶藏 $R(S, a)$ 為0，將下一個狀態的值0.81乘上 γ ，再相加得出此位置的值為0.73。

5. 編號(5):

$$V(S) = \max(R(S, a) + \gamma V(S')) = 0 + 0.73 \times 0.9 = 0.66$$

當前狀態無法獲得寶藏 $R(S, a)$ 為0，將下一個狀態的值0.73乘上 γ ，再相加得出此位置的值為0.66。

之前的方法是將值(Value)儲存在各個狀態(State)中，但「Q-Learning」是將值(Value)儲存在各個狀態(State)所對應行動(Action)當中。

Q-Learning記錄下每個狀態所對應行動的值稱為Q-value，而許多Q-Value所形成的表稱為Q-table。

Q-Function

$$Q(S, a) = R(S, a) + \gamma \max_{a'} (Q(S', a'))$$

- $Q(S, a)$: 在當前狀態下執行行動的Q-value
- $R(S, a)$: 在當前狀態中，選擇行動的獎勵
- $Q(S', a')$: 根據行動，轉移至下一個新狀態的Q-Value
- γ : 「衰退常數」是控制下個狀態的值對當前狀態值的影響力，通常是用來表示 Reward 對狀態的影響力

$$V(S) = \max_a (R(S, a) + \gamma V(S'))$$

將 $R(S, a) + \gamma V(S')$ 提出



$$Q(S, a) = R(S, a) + \gamma V(S')$$

修改 $V(S')$



$$Q(S, a) = R(S, a) + \gamma \max_{a'} (Q(S', a'))$$

圖：Q-Function導出圖

Q-Table

	a_1	a_2	a_3
S_1	$Q(S_1, a_1)$	$Q(S_1, a_2)$	$Q(S_1, a_3)$
S_2	$Q(S_2, a_1)$	$Q(S_2, a_2)$	$Q(S_2, a_3)$
.....

圖：Q-Table示意圖

要將Q-Table中的Q-Value更新優化，就要使用「時值差異(Temporal difference)」這個方法。

為什麼不直接用Q-Function求出新的Q-value? 原因是直接求出來的Q-Value可能不是最好的，會有高估或低估的情況。最後造成Q-Value量值忽大忽小來回跳動，這不是希望看到的情況。

時值差異(Temporal difference)

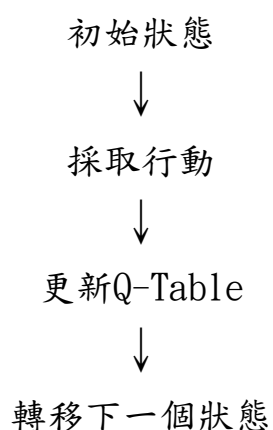
$$\begin{aligned} T(S, a) &= R(S, a) + \gamma \max(Q(S', a')) - Q(S, a) \\ &= Q'(S, a) - Q(S, a) \end{aligned}$$

將已經更新過的Q-Value減去原本的Q-Value，可求得兩者之間的差值。得到的時值差異，可以用來更新Q-Table中的Q-Value。

$$Q_t(S, a) = Q_{t-1}(S, a) + \alpha TD(S, a)$$

- α : 「學習率」決定了新獲得的資訊對現有Q-Value的影響程度。
 - α 數值介於 0 到 1 之間。
 - α 數值越大時，時值差異學習的幅度越大，下一步的Q-Value影響也越大。

在某個狀態下執行行動時，其Q-Value變化在原有的Q-Value中，又會參照時值差異來推估新的Q-Value。



圖：Q-Learning流程圖