ADLxMLDS HW3: Game Playing

學號：R06922030 系級： 資工碩一 姓名：傅敏桓

I. 基本要求

1. 策略梯度 (Policy Gradient)

策略梯度[1]為實現深度強化學習的一種算法，目標是建立並優化輸入為狀態(遊戲畫面)、輸出為動作概率的神經網路π。模型決策過程可以表示為$a = \pi(a|s,\theta)$，其中 a 代表行動、s 代表狀態、θ代表神經網路參數。由於模型的輸出是選擇動作的機率，這種算法在運行時仍保有一定程度的隨機程度。我們希望最大化獎勵的期望值，在更新模型時目標函數定為

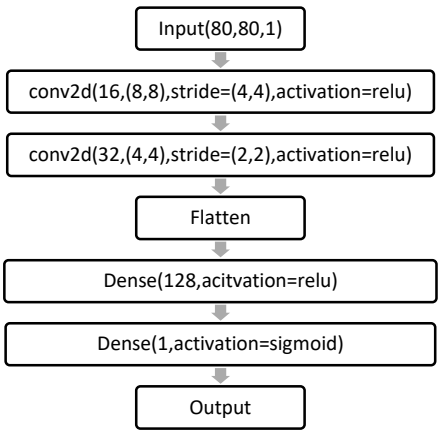
$$\mathcal{R}(\theta) = \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots \mid \pi(\cdot,\theta)\right],$$

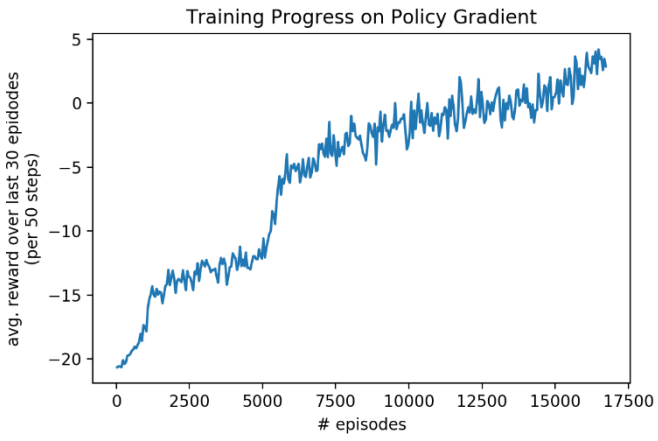
其中 r_i 表示環境給出的獎勵，離當前時間較遠的獎勵值對$R(\theta)$的影響隨衰減率γ遞減。若考慮獎勵遞減的概念，以$r(s,a)$表示獎勵估計函數、對數似然率(log likelihood)表示選擇動作的機率，則上述目標函數之梯度$\nabla_\theta R(\theta)$可以表示為

$$\nabla_\theta R(\theta) = \mathbb{E}\big(r(s,a)\nabla_\theta \log\ \pi(a|s,\theta)\big).$$

實作上因為深度學習的架構在優化模型時多以梯度下降法更新參數，我們可以將模型的損失函數以負的對數似然率改寫後，再做梯度下降的參數更新。

本次作業中透過策略梯度方法在 Pong 遊戲上實現深度強化學習，使用的神經網路的架構如下所示。模型以環境回傳的狀態作為輸入，通過數層卷積層和全連接層後得到輸出，對上述目標函數以梯度下降法 (optimizer=RMSProp, 學習率=1e-4) 更新模型；輸出的部分限定模型只能選擇兩種動作以簡化模型。前處理的部分照助教的做法把原始 RGB 影像處理成 (80, 80, 1) 的灰階影像，取和前一次狀態之差作為模型的輸入。每場遊戲結束時，先將各局獎勵隨時間以衰減率γ進行折扣，再對資料做標準化以減少資料之變異數，訓練過程如下所示，此模型經過約 15000 場遊戲的訓練後，在本機端測試得到 3.23 分的成績。





[1] 此處的策略梯度是指 Stochastic Policy Gradient 的方法。

2. 深度 Q 網路 (Deep Q Network, DQN)

深度 Q 網路是將深層神經網路應用在 Q 學習(Q-Learning)的強化學習算法。在 Q 學習的算法中，為了得出最佳的行動策略，我們考慮在不同狀態 s 下選擇各行動 a 的價值 Q(s,a)有多大，而這個 Q 值（目標 Q 值）和當前得到的獎勵 r 及下個時間點的 Q 值有關

$$Q_{i+1}(s,a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s',a')|s,a],$$

其中 γ 表示獎勵的遞減率，遊戲進行同時我們會根據這些目標 Q 值更新模型。訓練初期通常沒有足夠的資料改善模型，一般會以ε-greedy 的方法選取行動，即模型會有ε的機率不根據 Q 值選擇行動，藉此在環境中探索更多行動的可能性；然後當ε的值隨時間遞減，模型選擇動作的模式會趨於穩定。

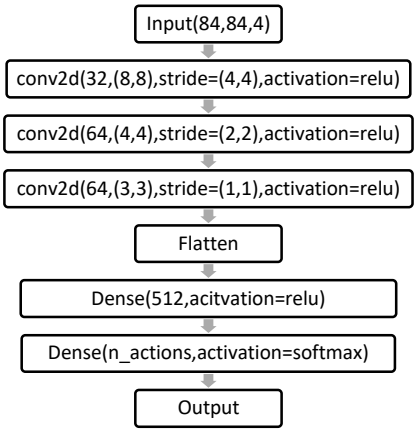
在原始的 Q 學習方法中，需要用矩陣記錄每對(s,a)對應的 Q 值，但若環境回傳的狀態維度大到沒辦法用逐一儲存，則採取價值函數近似 (Value Function Approximation) 的方法，以函數 Q(s,a,w)來模擬 Q 值的分布 (w 為函數 Q 之係數)。若引入深度學習的方法解決函數最佳化的問題，我們可以用深層神經網路來表示這個近似函數，再用梯度下降的方法來優化模型的參數，就形成深度 Q 網路的概念。我們希望縮小模型輸出和目標 Q 值之間的差距，如此目標函數可以寫成以下的形式：

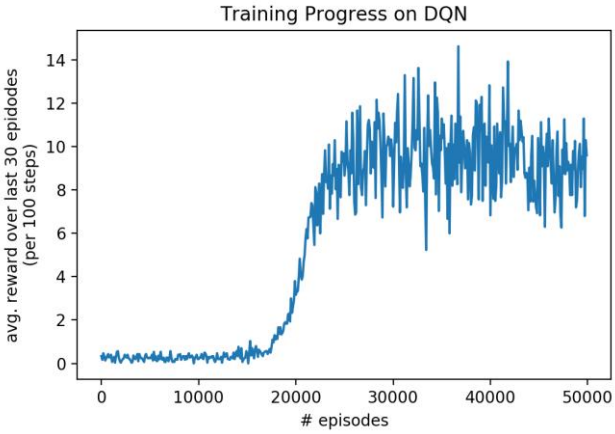
$$\mathcal{L}(w) = \mathbb{E}\Big[\Big(r + \gamma \max_{a'} Q(s',a',w) - Q(s,a,w)\Big)^2\Big].$$

通常以小批量梯度下降法更新模型時會假設樣本之間有相對獨立性，而在 atari 遊戲中環境的回饋是按照時間序列輸出，以致資料樣本之間可能有較大的關聯性。為了降低環境回饋的關聯性對模型的影響，深度 Q 網路在訓練時會將最近發生的環境回饋、行動和獎勵值的對應資料記錄在一個記憶庫中，每次更新參數時從中隨機抽取一些資料進行訓練。且通常會建立兩個架構相同的神經網路，其中一個會頻繁進行更新，每經過一段時間才會將參數同步更新到另一個 (目標網路)，而以目標網路預測 Q 值，以增加訓練過程之穩定性。

本次作業中以深度 Q 網路在 Breakout 遊戲上實現深度強化學習，使用的神經網路架構如次頁左圖所示。模型以經過預處理的四個連續畫面作為輸入，通過數層卷積層和全連接層後再通過 softmax 函數得到輸出，對上述目標函數進行梯度下降 (RMSProp, 學習率=1e-4)更新參數。訓練過程如下右圖所示，可以觀察到約在 20000 場前後模型有明顯進步。

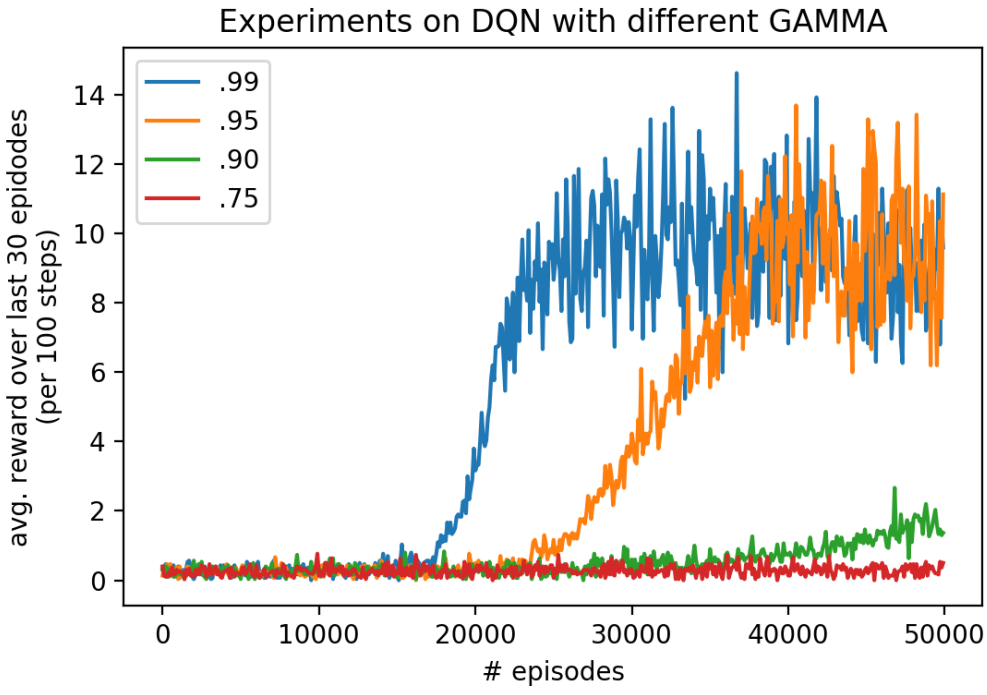
訓練模型時的各種參數設定 (hyperparamters)列舉如下：總遊戲場數 50000、記憶庫大小 100000、第 50000 步開始更新模型、模型更新頻率分別為每 3 步、3000 步，獎勵衰減率 γ=0.99、探索率ε=0.9、經過 1000000 步線性衰減為 0.1，每次抽 32 筆紀錄更新模型。訓練的過程如次頁右圖所示，此模型經過 50000 場遊戲的訓練後，在本機端測試得到 72.0 分的成績。





II. DQN 參數變因探討

本節針對深度 Q 網路的參數變因進行探討。實驗比照 I. 2.所述之參數設定，只改變衰減率γ來觀察模型訓練過程及結果的變化，實驗結果如下圖所示。我們可以觀察到模型的γ值越接近 1，訓練的效果看起來越好，除了訓練速度較快之外，也較有機會出現顯著的進步，而降到 0.75 左右時模型甚至沒辦法收斂。如前所述，γ代表 Q 值估計中獎勵隨時間的遞減率，推測因為γ值太小時獎勵遞減太快，模型較難從訓練資料中掌握預期得分的資訊，也就是如果當下的行動對較晚獲得的獎勵有所貢獻，衰減率越小就越難發現這個行動潛在的得分的可能性（以γ=0.99 和 0.75 為例，假設 10 步之後獲得獎勵，那麼模型認為當次行動獲得獎勵的評價分別為 0.99^{10}=0.904 和 0.75^{10}=0.056）。



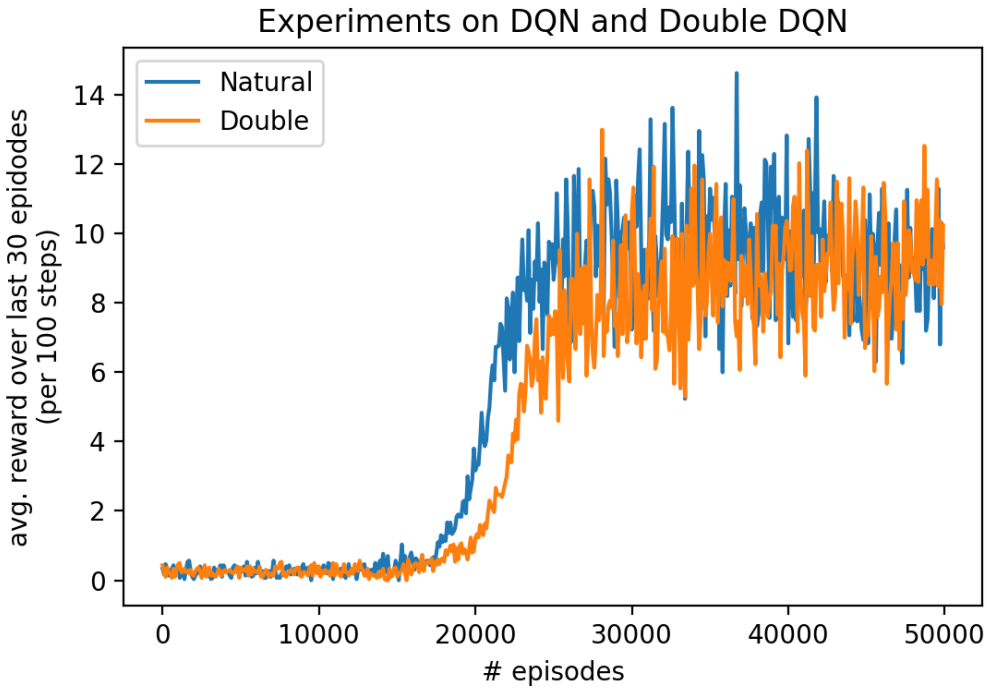
III. 實作進階模型

1. 雙重深度 Q 網路(Double DQN)

如上所述，我們在訓練深度 Q 網路時是朝向目標 Q 值的方向更新參數，但目標 Q 值中包含的 max 操作容易造成模型過度優化 (overoptimism) 的問題，引入雙重 Q 網路的目的是希望能解決這樣的計算偏差，分別用兩個交替更新的 Q 網路來選擇動作和評估動作。我們利用原始的深度 Q 網路模型中的兩個 Q 網路來實作以上的概念：用當前 (頻繁更新) 的 Q 網路來選擇動作，用目標網路來計算目標 Q 值。我們保持原始模型的實驗設定，同樣更新模型使模型輸出逼近目標 Q 值，僅改寫目標 Q 值如下：

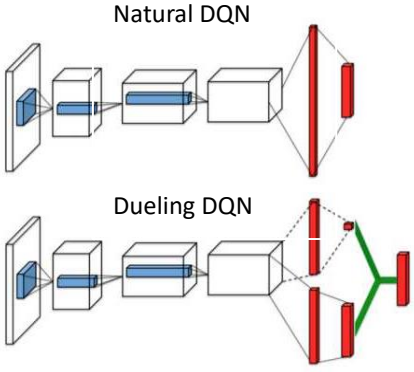
$$y^{DoubleDQN} = r + \gamma Q_i(s', \text{argmax}_a Q(s', a, w_t), w_t^-).$$

訓練過程如下所示，發現在 Breakout 遊戲上新版的模型並沒有特別進步，反而看起來略差於原版的模型，這個部分可能還要花時間實驗不同的參數設定，或是換個遊戲觀察。

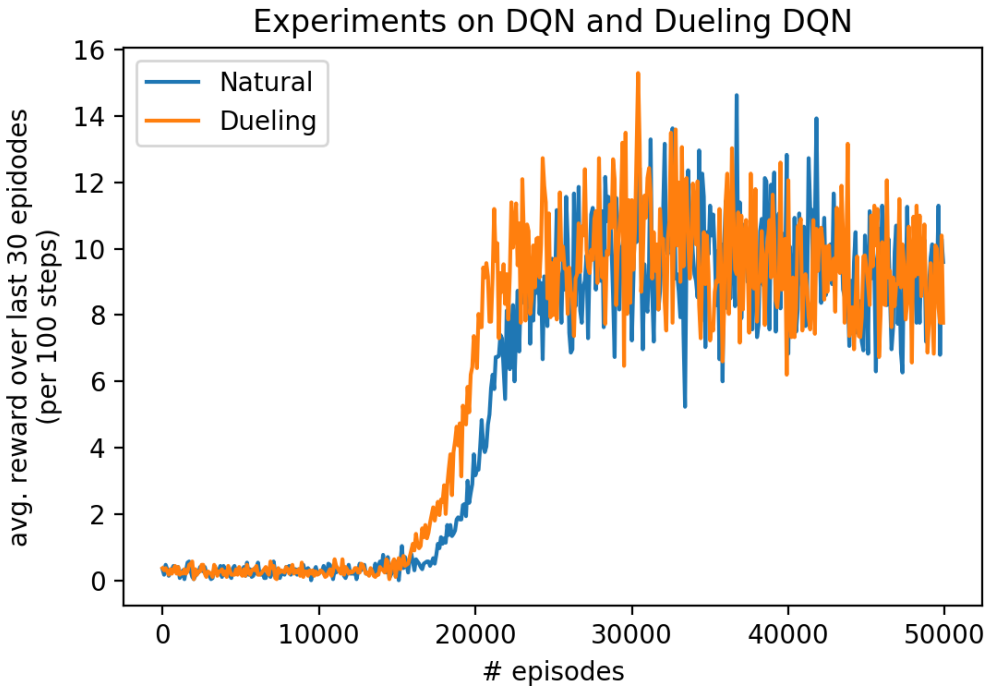


2. Dueling DQN

我們知道某些狀態下，例如打磚塊遊戲在球離擋板很遠的時候，行動的選擇對獎勵的影響其實都不大。Dueling DQN 把網路內部的 Q 值的估計 Q(s, a)分解成 V(s) + A(s, a)，其中價值 V(s)只與狀態有關，而 A(s, a)代表動作相對於該狀態的優勢 (平均獎勵的評估)，以減少上述情形對模型訓練造成的偏差。兩種模型的架構比較如右圖所示。



以下是 Dueling DQN 和原版模型在訓練過程的比較，實驗設定都比照原版模型，僅改寫 Q 值的估計式為 Q(s, a)= V(s) + A(s, a)。可以觀察到 Dueling DQN 雖然表現略好於原版的 DQN，但也沒有很顯著的進步，這個部分可能還要花時間實驗不同的參數設定，或是換個遊戲觀察差異。



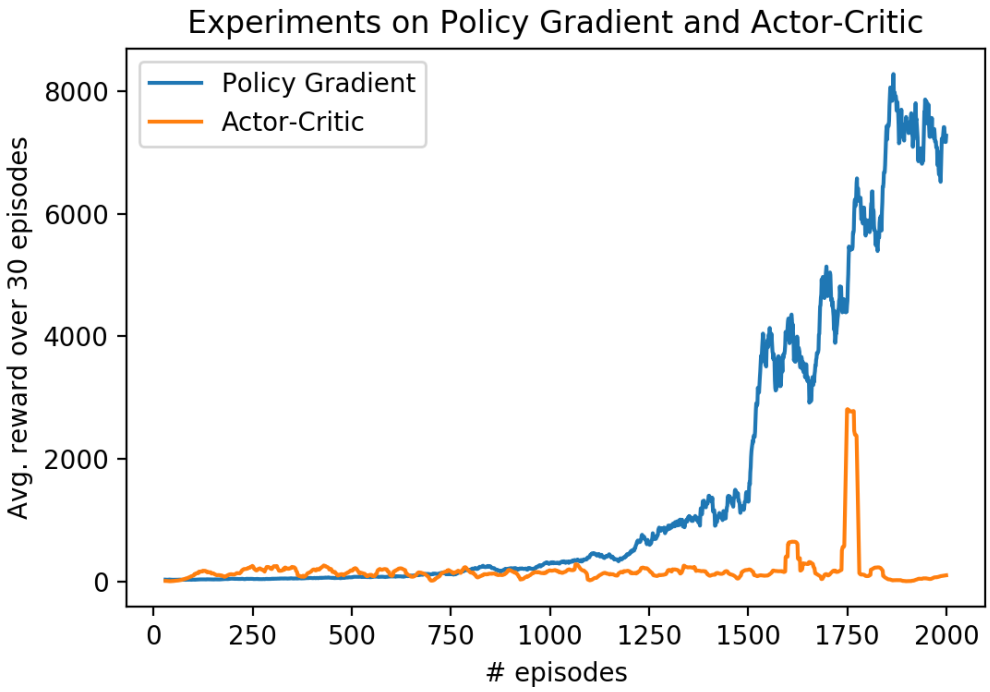
3. Actor-Critic

前面分別實作了基於價值 (Value-based) 的深度 Q 網路、基於策略 (Policy-based) 的策略梯度兩種強化學習的方法，而 Actor-Critic 就是希望結合兩種模型的特性而發展出來的做法。模型架構中，策略網路是行動者 (actor)，負責動作的選擇；價值網路是評價者 (critic)，用來評價行動者所選動作的好壞，再根據時間差分誤差對兩個網路進行更新。

我們在 CartPole-v0 遊戲上比較 Actor-Critic 和策略梯度的差異。這個遊戲的環境狀態和行動種類相對簡單很多，實驗時也只用比較簡易的網路。以下列出實驗設定：

Policy Gradient	全連接層(10)→tanh→實數輸出 /Optimizer=Adam(0.01)
Actor-Critic	Actor: 全連接層(20)→relu→全連接層(# actions)→softmax Critic: 全連接層(128)→relu→實數輸出 /Optimizer=Adam(0.01)

實驗結果如下圖所示，可以發現原本的策略梯度的方法反而比較穩定，Actor-Critic 則一直無法收斂。有人認為可能因為 Actor-Critic 的模型中，價值網路的收斂狀況會連帶影響策略網路的更新，而價值網路本身較難以訓練，使得整個模型更加不易收斂。而後有各種改良版陸續被提出，例如結合 DQN 的 DDPG (Deep Deterministic Policy Gradient)、以多線程同步訓練的 A3C (Asynchronous Advantage Actor-Critic) 等方法。



環境設定

Python 3.5.3 / GTX 1080 Ti / Debian 4.9.51-1

使用套件: numpy, scipy (1.0.0), tensorflow (1.3), Python Standard Library