

1. (20%) Policy Gradient 方法

- 請閱讀及跑過範例程式，並試著改進 reward 計算的方式。
- 請說明你如何改進 reward 的算法，而不同的算法又如何影響訓練結果？

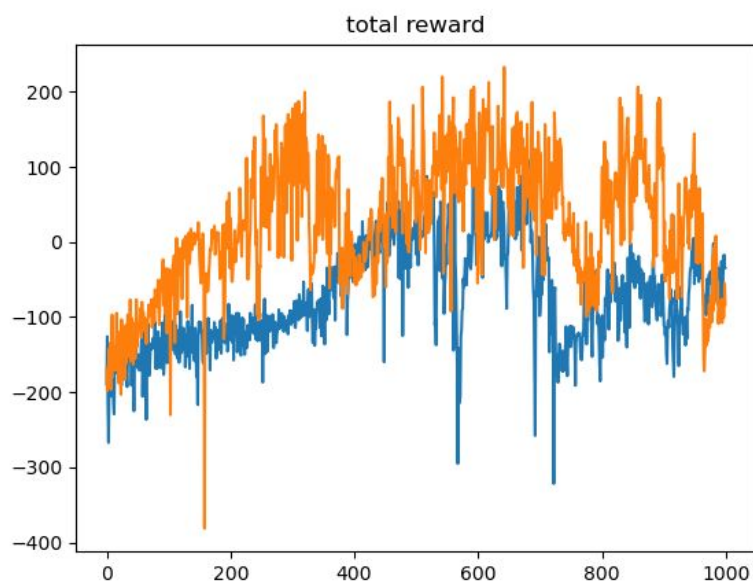
改進reward方式：

原本範例 code 中的 reward 算法是將每個 action 的 reward 都以整個遊戲的 reward (total reward) 來代替，但如同教授在影片中所說，這樣來計算每個action 的 reward 是不合理的，對於第  $i$  個 action，應該只要考慮第  $i$  個action ~ 遊戲結束前的最後一個 action 所得到的reward。此外，對於第  $i$  個 action 而言，對越後面的 action 應該影響力越低，因此應該要有一個 discount 的算法來衡量這件事。最後我參考了教授影片中的公式，也就是將原本的  $R(\tau^n)$  改成：

$$\sum_{t'=t}^{T_n} \gamma^{t'-t} r_{t'}^n$$

並將參數  $\gamma$  設為0.99、EPISODE\_PER\_BATCH = 10、NUM\_BATCH = 1000。而在改進算法過後，訓練結果出現了大幅進步：

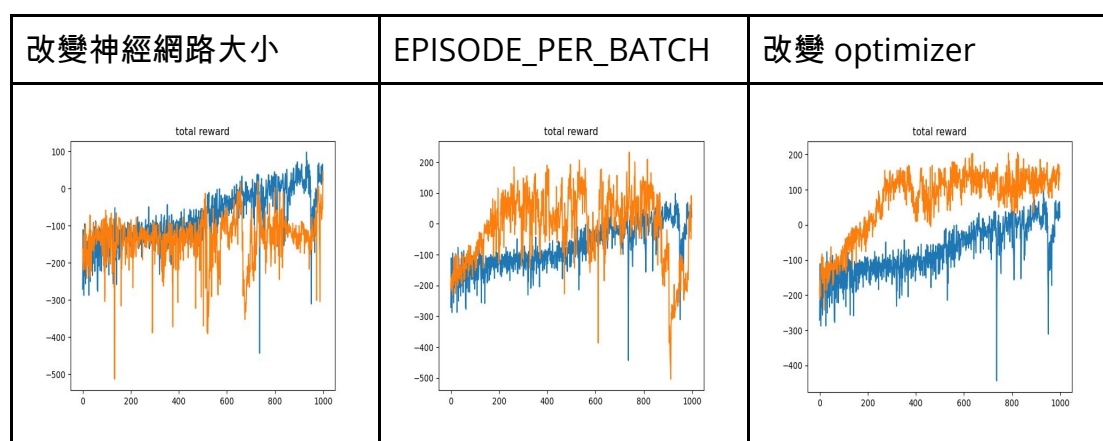
( 藍色線為範例code 的結果，橘色線為加入 discount 算法的結果 )



可以發現範例 code 會在 num\_batch 大約等於 500 時才漸漸開始得到大於 0 的 reward，而加入 discount 算法的 code 約在 num\_batch=250 時就開始得到大於 0 的 reward。整體而言橘色線都在藍色線之上，意味著使用 discount 算法的 code 大致都有著較好的結果。

2. (30%) 試著修改與比較至少三項超參數（神經網路大小、一個 batch 中的回合數等），並說明你觀察到什麼。

我試著改變了三項超參數，分別是神經網路大小、EPISODE\_PER\_BATCH、以及所選用的 optimizer。神經網路的部分，我將原本的神經網路中間又加了兩層，分別是 `nn.Linear(16, 32)` 及 `nn.Linear(32, 64)`，使得神經網路架構更大。在 EPISODE\_PER\_BATCH 的部分，我從原先的 5 調整到 10，最後一個實驗是將 optimizer 由助教給的 SGD 改成 Adam ( $\text{lr} = 0.003$ )。下表為三者各自與原先的 sample code 在 total reward 的差異（藍色線皆為 sample code）：



### （一）改變神經網路

從上圖可以發現，增加了神經網路的規模後，不但沒有變好的趨勢，反而在 total reward 的 performance 上比原先較小的網路更差。我認為有兩種可能：一是我還不瞭解這種遊戲比較適合怎麼樣的網路架構來做訓練，因此盲目的加大神經網路的大小反而無益，二是可能這個遊戲本來就不適合使用較為複雜的網路，畢竟它本身就不是一個有很多變數的遊戲，因此將神經網路複雜化反而可能使他在訓練的表現上得到反效果。

### （二）增加每個 batch 的回合數

由上圖可知，增加每個 batch 的回合數大致可以使表現更好，更重要的是在一次更新內玩越多次遊戲，就可以使 total reward 變成正值的速度越快，由圖也可以發現一開始兩者的曲線幅度是有很大的差距的。

### （三）改變 optimizer

這一組的比較就更為明顯，在前1000個batch中，使用Adam 作為 optimizer明顯在 total reward 的表現比使用 SGD來得更好。因為 Adam 有收斂較快的特性，因此理論上確實在前面更新的部分確實會比 SGD 來得更加優異，但若是更新到比較後面時，也許SGD的方法就比較能逼近收斂點。

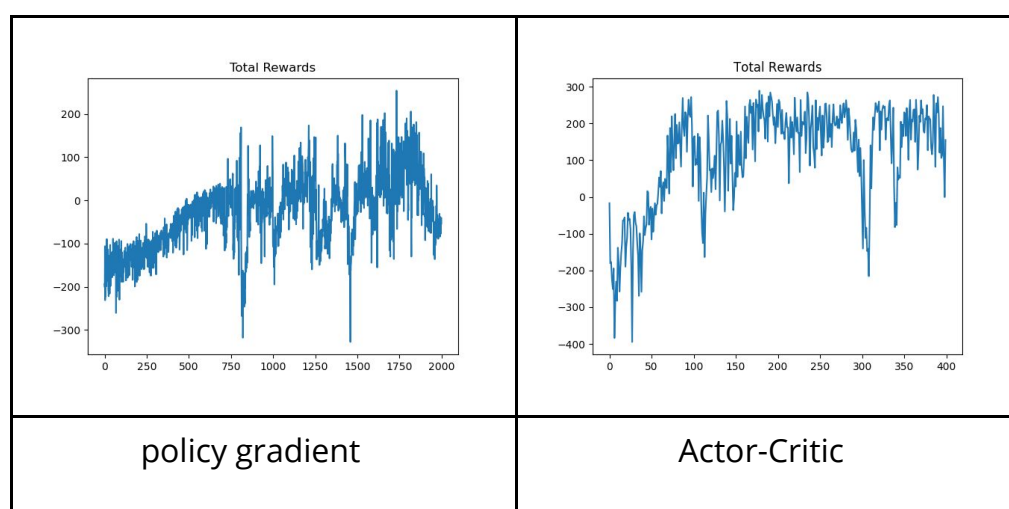
### 3. (20%) Actor-Critic 方法

- 請同學們從 REINFORCE with baseline、Q Actor-Critic、A2C 等眾多方法中擇一實作。
- 請說明你的實做與前者 ( Policy Gradient ) 的差異。

在這一題的實作中，我選擇了 reinforce with baseline 的方法 ( 與 policy gradient 實作出來數據的差異會在下一題呈現，此題僅說明 Actor-critic 實作與 policy gradient 的差異。 )，而 Actor-Critic 與 policy gradient 的最大差別就是 Actor-Critic 的方法比原先的 policy gradient 多了 Critic 的角色。Critic 會基於 Actor 所採取的 action 給分，也就是 Critic 會判斷這次的 action 是不是一個好的行動，如果是才在 gradient ascent 時更新的多一點，反之則 ascent 地少一些。另外，在這個方法中加入 baseline 的概念，使得學習的成長速度更快，而且曲線的波動幅度也較小。因此，如助教投影片的 pseudo-code 所示，兩者實作的差異就是增加一個baseline，並每次都由 minimizing baseline 與 reward 的 distance ( L1\_loss ) 去 refit baseline。

### 4. (30%) 具體比較 ( 數據、作圖 ) 以上幾種方法有何差異，也請說明其各自的優缺點為何。

首先比較 policy gradient 與 Actor-Critic 在實作上有何差異以及各自的優缺點：



由上圖可以發現，除了一開始Actor-Critic 有較大幅度的震盪，甚至一度快掉到-400分以外，在 total rewards 的表現上明顯 Actor-Critic 優於

policy gradient。更驚人的是，在num\_batch = 50左右時，Actor-Critic 的模型就已經能漸漸地得到大於0的 total reward，並在 num\_batch = 100時就已經能達到接近 250 分的高分，這樣的學習速度明顯要比傳統policy gradient 好上數倍！

而兩者的差異已經在第三題大略說明完畢，在反覆查詢了網路上的一些資料過後，發現policy gradient 的優點在於他在收斂方面的能力是較好的、較容易實現隨機的策略、並在高維空間的計算中更為有效。但其缺點也是顯而易見：不易收斂至全域最佳值，容易收斂到局部最大值 (local maximum) 上，另一個缺點是policy gradient 為走完一個回合才更新，效率較差。

至於Actor-Critic，因為是由 policy gradient 演化而來，其Actor部分繼承了 policy gradient 的優點，並能夠完成單步更新，以增加學習的效率。缺點在於新增的 Critic 部分，其模型架構相當難收斂，因此可能較難達到最佳值。

接著比較第一題方法的優缺點：

使用 discount 算法的優點當然是可以使每個 action 的 reward 更能夠貼近其真正造成的影響，可以使訓練的過程中 total reward上升較快。缺點是其與原本的算法相比，似乎在 total reward 的震盪明顯地較大，我猜測可能是因為每次都乘上固定的 gamma 值，因此會使得系統較為不穩定。仔細查詢相關資料後，找到網路上也有相關的論文給出了相對應的解決方法，如逐漸增加 discount factor  $\gamma$  直至最終值等，亦能有效降低過擬合等現象。