Machine Learning 2020 - Homework 15 Report

學號：b08902100, 系級：資工一, 姓名：江昱勳

1. Policy Gradient 方法

   (a) 請閱讀及跑過範例程式，並試著改進 reward 計算的方式。

   (b) 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？

   這裡我採用了 discounted reward 的方式改進 reward 的算法，並且每搜集 64 個 episode 的資料後，才去更新 Actor，這裡 discount 的衰減率我設成 0.9，最後 total reward 為 131.4621654997108，final reward 為 0.5483643130777409。

   而原本沒有改進 reward 的方法 total reward 約為 -54.67351046366531

2. 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。

   原始網路模型: 第一層線性層 8×16 並使用 tanh 作為活化函數、第二層線性層 16×16 並使用 tanh 作為活化函數，最後一層線性層 16×4 並使用 softmax 作為活化函數，使其作為機率輸出。

   修改後的網路模型: 第一層線性層 8×32 並使用 tanh 作為活化函數、第二層線性層 32×32 並使用 tanh 作為活化函數，第四層線性層 32×16 並使用 relu 作為活化函數，最後一層線性層 16×4 並使用 softmax 作為活化函數，使其作為機率輸出。

   嘗試將網路從後發現 model 會變得更加的不穩定，從圖 1可以看到 train 了 300 個 epoch 後 model 還非常的浮動，到 500 epoch 後有稍微穩定一些，不過整體表現還是不好，推測是反而較不容易學到正確的知識，或許每次的 episode 要再加大一些。

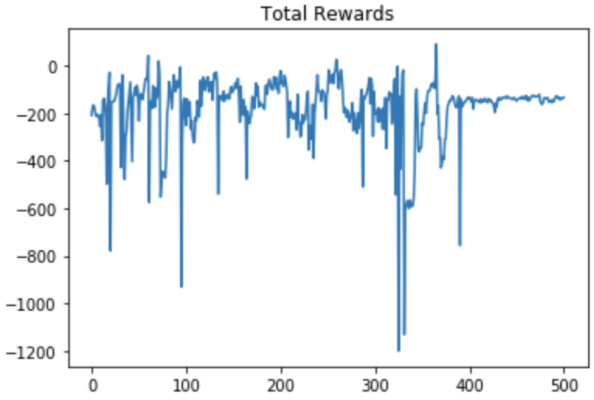

Figure 1: Policy Gradient with larger model

   當初會採用 0.9 作為衰減率的原因是因為使用 random agent 玩了 1000 次後的平均 episode 長度為 91.38，第一個 epoch 到最後一個 epoch 的差距為 $0.9^{91.38} \approx 0.00006586892$，而每 10% 左右的差距為 $0.9^{9} \approx 0.387420489$，我自己估計這樣的結果是可以的。

   從圖 2可以看到，實際上採用 0.8 作為衰減率訓練 total reward 約為-211 左右，而 final reward 約為-100，可以發現結果也不盡理想，推測我們還是會希望採用更遠一點的操作，因為一次的失誤可能會需要很長一段時間的修正。

   若訓練時每個 epoch 只採用 8 個 episode 時，模型在訓練時會變得較不穩定，可以從圖 3看到，起伏非常的大，最後的結果也不好，total reward 為-214.28588577033656，final reward 為-100。並且從圖 4也可以看到結果非常的差，直接翻車了。

3. Actor-Critic 方法

   (a) 請同學們從 REINFORCE with baseline、Q Actor-Critic、A2C 等眾多方法中擇一實作。

Figure 2: Policy Gradient with discount factor 0.8

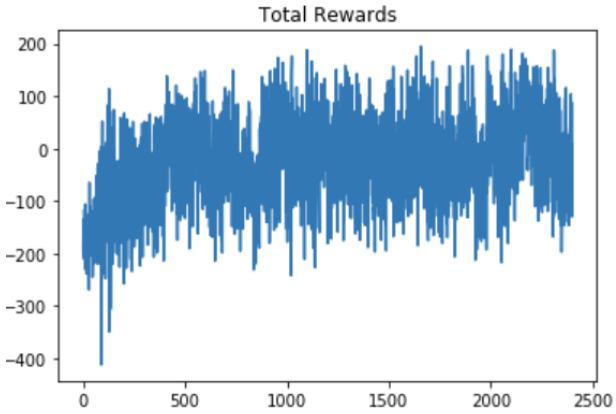

Figure 3: Policy Gradient with 8 episode each batch

嘗試使用 A2C 的方式訓練模型遊玩此遊戲，每次抽取 128 次的 episode 拿來更新，訓練 300 epoch，訓練的時候每次都同時用同一組 episode 的資料去更新 actor 跟 critic，同時衰減率與前面相同我設成 0.9，最後 total reward 為-165.2634328122486，final reward 為-100。

(b) 請說明你的實做與前者（Policy Gradient）的差異。

可以看到這次模型的結果不盡理想，推測是因為我同時訓練 actor 跟 critic，在 actor 還沒學好的時候 critic 學到的內容自然也不盡正確，而 policy gradient 在這個遊戲上可能沒那麼困難，所以加入 discount 後就可以獲得不錯的結果。

此外我也有嘗試先訓練 actor 100epoch 後再訓練 critic 100epoch，再一同訓練 actor + critic 效果也不太好，甚至比原先效果更差，這部分我推測也有可能是因為在訓練 critic 的時候使用的資料主要仍是採用 critic 自己生成出來的部分，而不是來自環境，所以應該改善這部分。

4. 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。

從圖 8可以看到，其實原先只採用 policy gradient 的方法在落地的時候就有不錯的表現了，不過其遊玩過程仍不太好；而改用 0.9 的衰減率後可以從圖 9看到訓練過程大幅的提升，而結果也很不錯 (圖 10)。其他方法我則幾乎沒有看到太多好的表現。最讓我訝異的部分是 A2C 的部分意外的差，完全找不出任何優點。

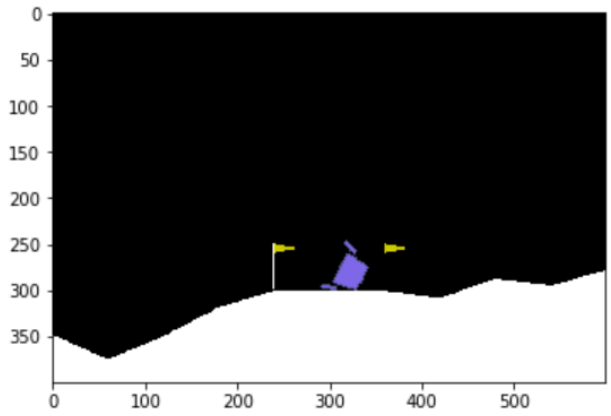

Figure 4: Policy Gradient with 8 episode each batch

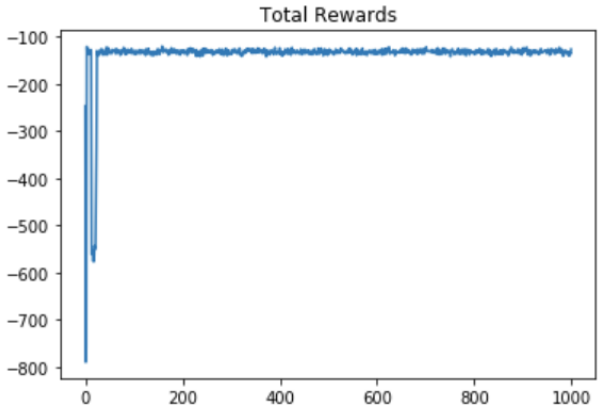

Figure 5: A2C

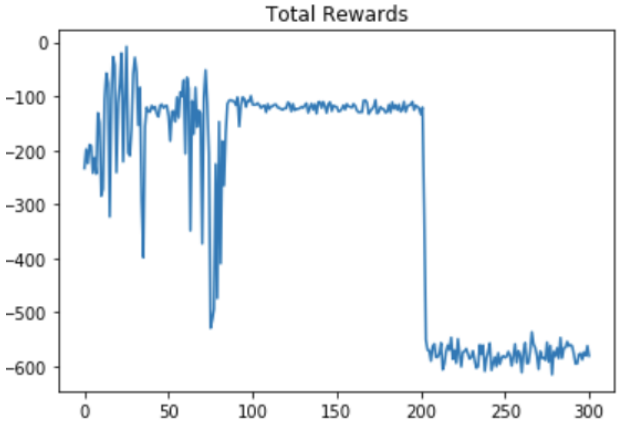

Figure 6: Different training of A2C

Figure 7: Policy Gradient

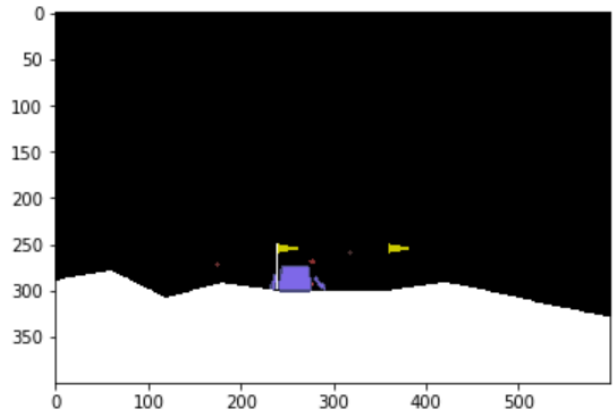

Figure 8: Policy Gradient

Figure 9: Policy Gradient with discount factor 0.9

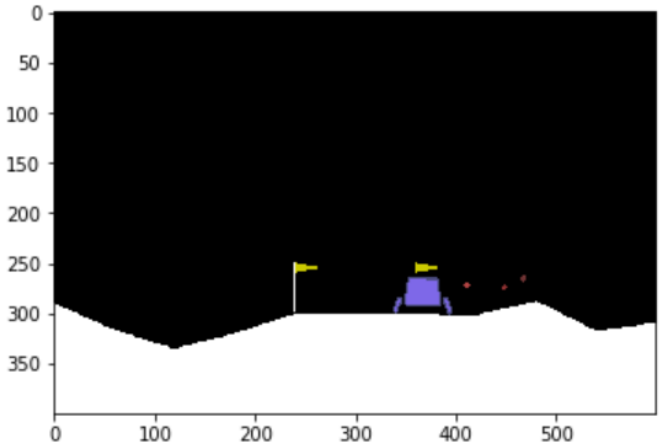

Figure 10: Policy Gradient with discount factor 0.9