MLDS: HW3

December 16, 2017

Professor Hung-Yi Lee

電機四 B03901018 楊程皓

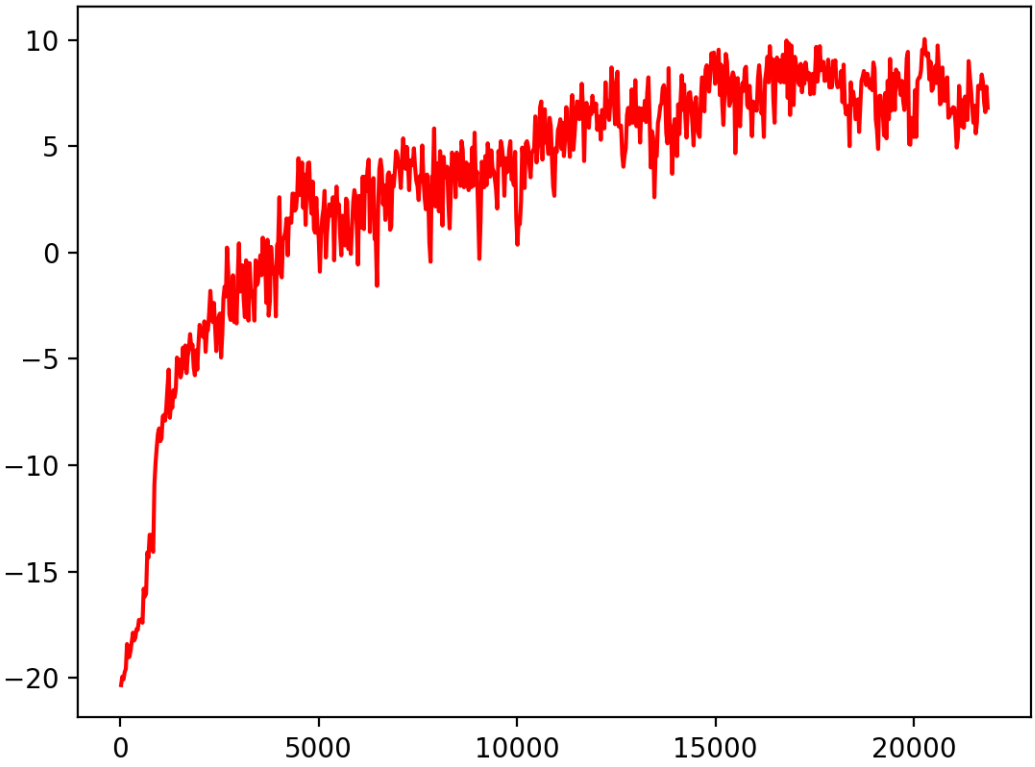
Policy Gradient & DQN Models

- Policy Gradient
  我的架構先將 observation 做 preprocess, 具體作法為先從 210x160x3 切為 35:195,:,0 (160x160)，即為將有球跟擋板的部分擷取出來後取第一個顏色，再對長寬各做兩倍的 down sample 得到 80x80，再把 background 顏色去掉，最後將值不為 0 的部分都取為 1，即為球與兩擋板的位置。在做完 preprocess 之後，將前一個 frame 與現在的 frame 的差值及現在的 frame 合併，作為 model 的 input (80x80 + 80x80)。經過一個單層的 fully connected hidden layer (200 neurons) 後，生成代表三個 action 機率的 output layer (不動, 往上, 往下)。在給 model train 的 reward 使用的是 discounted 後且 normalized 的，更新頻率則是每個 episode 結束後都 update model parameters.
- DQN
  基本上我的 DQN 跟助教提供的 model 一樣:
  Conv2d, out_channels=32, kernel_size=8, stride=4, activation=relu
  Conv2d, out_channels=64, kernel_size=4, stride=2, activation=relu
  Conv2d, out_channels=64, kernel_size=3, stride=1, activation=relu
  fully-connected, neurons=512, activation=relu
  fully-connected, neurons=actions number

  除了 target network update frequency=4000 外，其餘參數也都跟助教一樣，就不一一贅述。

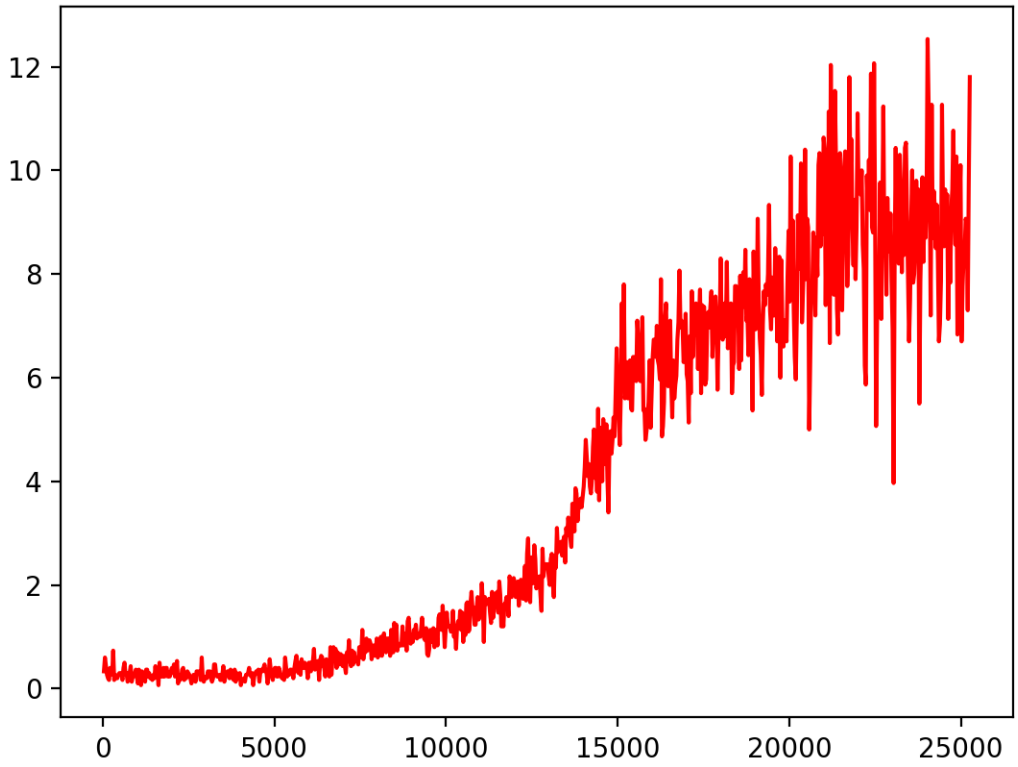
Learning Curve of Policy Gradient

我的 Policy Gradient learning curve on Pong 如下圖，橫軸為 trained episodes 數量，縱軸為前 30 episodes 的平均得分，共 train 21870 episodes, 最後 testing 平均分約為 6-7 之間。



Learning Curve of DQN

我的 DQN learning curve on Breakout 如下圖，橫軸為 trained episodes 數量，縱軸為前 30 episodes 的平均得分，共 train 25000 episodes, 最後 testing 平均分約為 47 。由於 training, testing 的計分標準不一樣，在 learning curve 上看不到逼近於 47 的值。



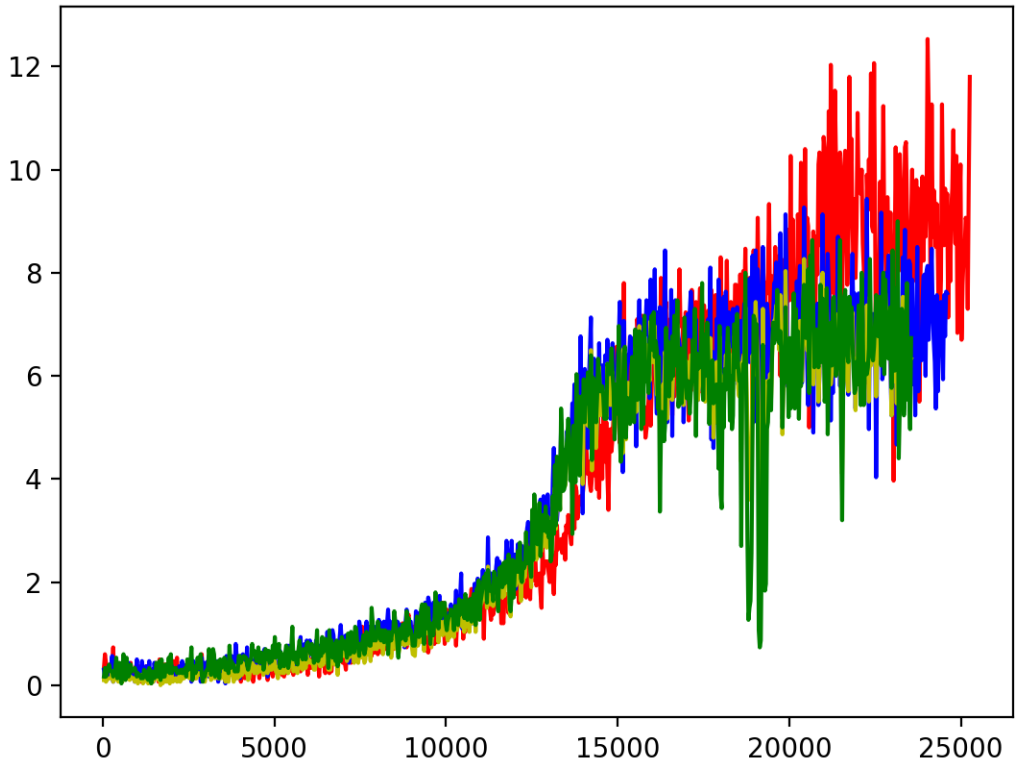
Experimenting with DQN hyperparameters

My hyperparameters:

Environment steps: about 3M steps (由於 training 時間有限，不是每組參數都有跑到 2.5M)
Experience replay size: 10000
Learning start size: 10000
Target network update frequency: 4000
Online network update frequency: 4
Gamma: 0.99
Batch size: 32
Exploration rate: 1 to 0.05 (decay: 10^{-6}
RMSProp, learning rate: 10^{-4}, decay: 0.99

My key hyperparameter: Target network update frequency
Origin: 4000, color: red

1. Target network update frequency: 1000. Color: blue.
2. Target network update frequency: 500. Color: yellow.
3. Target network update frequency: 40. Color: green



Reason: 我認為 Q-network 與 target-network 兩個 identical 的 network 的更新頻率會很大程度的影響 DQN 的進步，針對不同問題給不同的 target network update frequency 很可能造成很大的影響。以此問題為例，在最小的 frequency: 4000 steps per update 在後期接近 20000 episodes 時有最好的 training 效果，雖然前期的上升速度沒有 frequency 較高的幾個 model 來得快，這也是正常的，因為畢竟前期 update 的頻率快一些很可能讓他在急遽上升期 (新的 information 進來的時候) 學的更快，然而在到達一個瓶頸後，比較小的 update frequency 會讓 Q-network, target-network 能夠穩定成長，另外幾個 model 可能就遇到 moving destination 或其他問題而無法如紅色的 learning rate (最小 freq: 4000) 繼續上升。

Bonus: Improvement of DQN

我實作了兩種架構: Dueling DQN (Blue) & Double DQN (Red), 紅色的 learning curve 則是我的基礎 DQN 作為比對。由於 training 的時間不夠，我的 Dueling DQN & Double DQN 還沒有成長到能夠超越原本 DQN 的程度。

1. Dueling DQN: Dueling DQN 在 network 中將 Q value 拆分成兩項: $Q(s,a) = V(s) + A(s,a)$, 雖然在 update 與 choose action 和其他所有行為都不會個別看 V, A 的值，但對 model 而言，V 代表的是這個 state 的好壞，A 代表的是這個 action 在這個 state 有沒有讓情況變好。這樣使得 Agent 在學習的時候，可以將選擇拆分為，先進入好的 state，再想辦法在這個 state 中找到最好的那個行為，使得 Agent 進步得比原來的 DQN 更快。

2. Double DQN: 基礎的 DQN 更新方式為:

$$Q(s,a) \rightarrow r + \gamma \max_a Q_t(s', a')$$

但因為我們 model 所存的 Q-value 相較於真實的值有一定程度的 bias, max 這個 function 就會讓我們容易取到最大的 positive error, 從而往往 overestimate Q values。Double DQN 則是利用 q-network & target-network, 將 update 更改為:

$$Q(s,a) \rightarrow r + \gamma Q_t(s', argmax_a Q(s', a))$$

這種方式已經被證明能很大程度地降低上述的 maximization bias.