ADLxMLDS-hw3

孫奧 R05922147

1 Basic Performance

1.1 Describe your Policy Gradient & DQN model

1.1.1 Policy Gradient

PG 的部分跟助教的架構是一樣的，在用來逼近 policy function 的部分使用的是多層的 NN 的架構，架構如下：經過預處理的圖像通過一層

CNN(filters=16,kernel=(8,8),strides=(4,4),act=RELU)

然後

CNN(filters=32,kernel=(4,4),strides=(2,2),act=RELU)

然後 Flatten 接上一個 Dense(128)，最後接到一個 Dense(3)，雖然 pong 有 6 個有小動作，但是其實(0,1)(2,4)(3,5)每一組對應的是同一個動作。所以恰好選擇 index+1 即可。(即[0,1,2]->[1,2,3]而 1,2,3 分別對應了一個不同的動作)。

1.1.2 DQN model

DQN 的輸入是[84,84,4]的輸入圖像。在逼近 value function 的部分，首先圖像經過一個：

CNN(filters=32,kernel=(8,8),strides=(4,4),act=RELU)

然後：

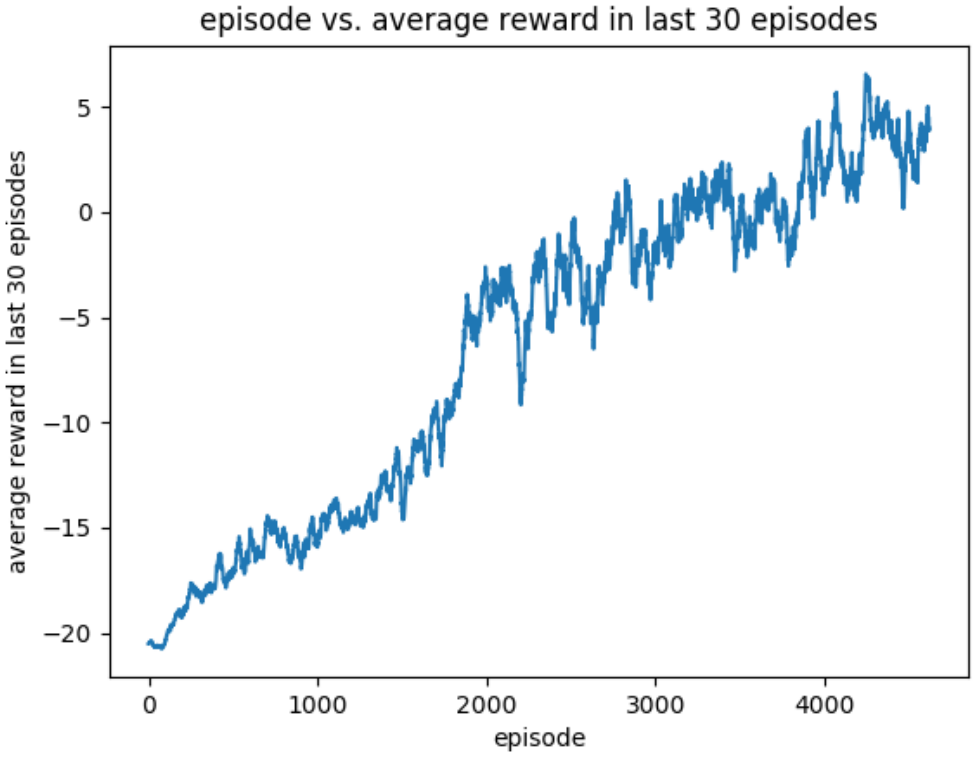
CNN(filters=64,kernel=(4,4),strides=(2,2),act=RELU)

然後

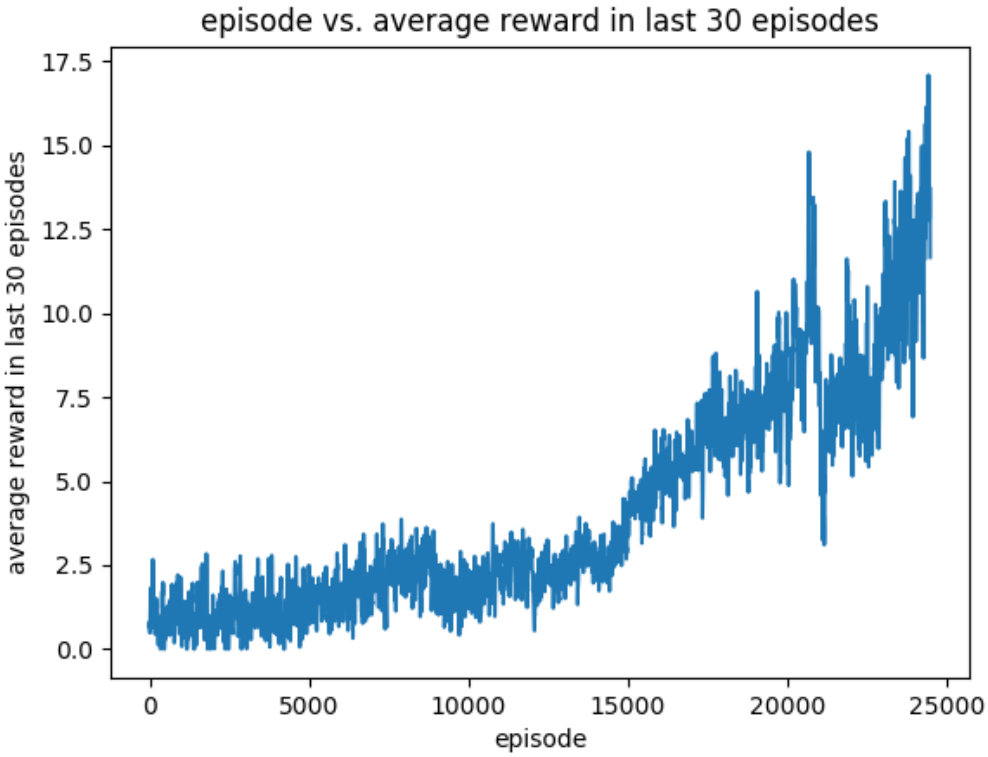
CNN(filters=64,kernel=(3,3),strides=(1,1),act=RELU)

然後 Flatten 接上一個 Dense(512)，最後接到一個 Dense(4)，其中 4 是 breakout 這個遊戲的 action 的數量。

1.2 Plot the learning curve to show the performance of your Policy Gradient on Pong



1.3 Plot the learning curve to show the performance of your DQN on Breakout



2 Experimenting with DQN hyperparameters

2.1 Plot all four learning curves in the same graph

這邊測試的不是breakout這個遊戲 而是一個簡單的迷宮 如 Figure 2-1 所示的迷宮，紅色的是 agent，黑色的 hell 以及黃色是 goal，當 agent 到 hell 得到 reward=-1，當 agent 到 goal，得到的 reward 是 1，其餘情況的 reward 是 0。

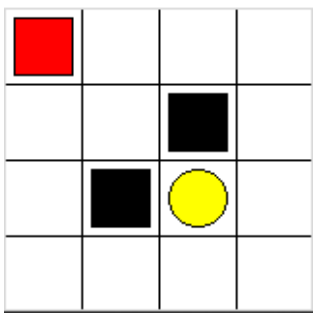


Figure 2–1

這邊我選取的 hyperparameter 是權重的初始化的 std，在 tensorflow 中，我用的初始化使用語句

`tf.random_normal_initializer(0., weight_mean)`

則測試的 hyperparameter 是上面的 `weight_mean`。結果如 Figure 2-2

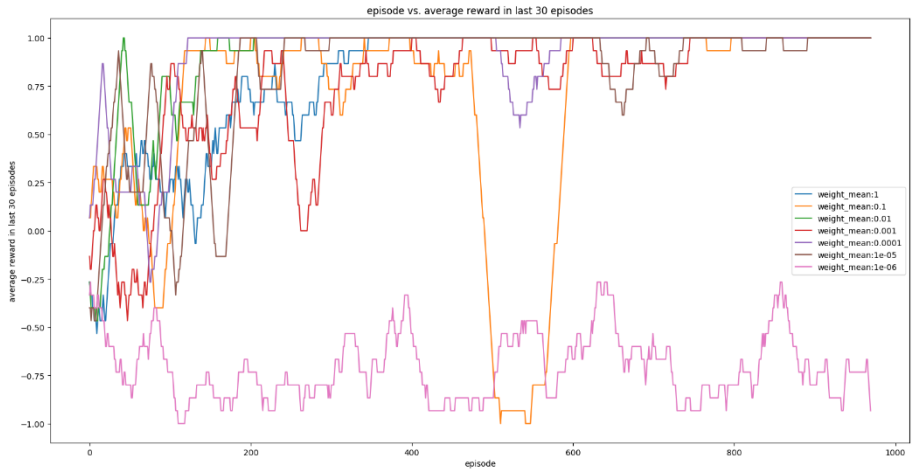


Figure 2–2

此外，還有測試 learning rate，如圖 Figure 2-3

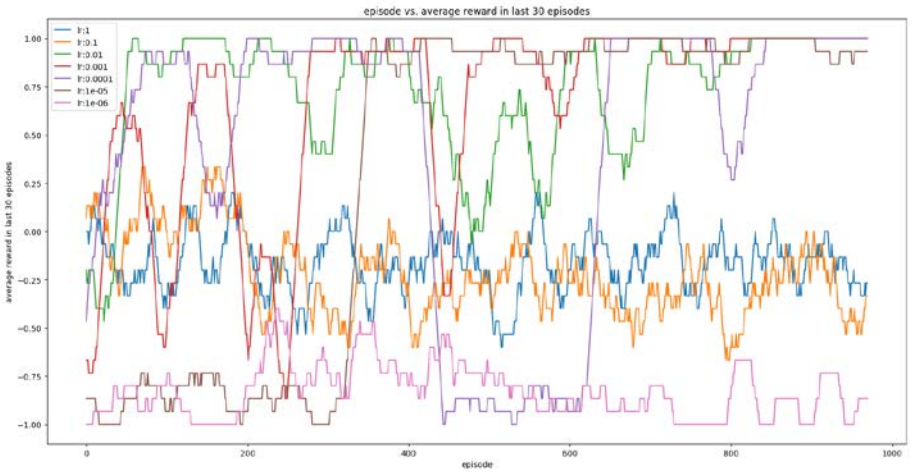


Figure 2–3

2.2 Explain why you choose this hyperparameter and how it effects the results

选择 `weight_mean` 的原因是在學習和使用 DQN 的時候，我碰到最大的問題在於：如果把 DQN 看作某種程度上的 supervised learning，當我們想要把例如$Q(S_1, a_1)$的值往$Q^*(S_1, a_1)$去調整（Q^*假設是正確的值，實際上我們用 TD-error 逼近 0 來替代），在 q-

learning 的時候我們只是去 update 那個 table 而不會影響到表中其他的值，但是在 DQN 中我們做這件事勢必會影響到其他的值（例如我們通過調整權重讓$Q(S_1, a_1)$變大一些，由於權重改變了，也許$Q(S_2, a_2)$變小了，然而正確的$Q^*(S_2, a_2)$的方向應該是讓$Q(S_2, a_2)$變大），這個是非常棘手的問題。

所以限制每次調整的方向和幅度就顯得很重要，這有兩個方法可以做到，其一就是很顯然調整 lr，其二就是調整 weight 的大小。

此外在做作業的過程中，在理解說 DQN 希望用 **NN 來當作很大的 q-table 的估計**這一個表述我覺得也有些讓人困惑。因為一般來說用 NN 來逼近函數（做 regression 之類的）的過程中，一個很重要的議題是 overfitting（限制 weight 小大小是其中一個方法），但是這一邊似乎並沒有要這樣（包括助教的 code 以及我網上找到的 code 中似乎都沒有提及類似 dropout 等技巧），難道這裡 NN 的作用真的就是背下來這個 table 嗎，那為什麼不直接使用 table 呢（使用 table 還可以避免調整權重帶來相關性的問題）？