R06725048 資管碩一 陳信豪

MLDS HW3 Report

- **Describe your Policy Gradient & DQN model**

  [Policy Gradient]

  我的 deep learning model 如下，與助教大致相同

  2 CNN + 1 DNN 最後再 output

  Activation 全使用 relu (除最後一層是 softmax)

  optimizer 為 Adam (learning rate = 1e-4)，loss 為 categorical_crossentropy

  kernel_initializer 為 he_uniform

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape_1 (Reshape)          (None, 80, 80, 1)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 20, 20, 16)        1040
_________________________________________________________________
activation_1 (Activation)    (None, 20, 20, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 32)        8224
_________________________________________________________________
activation_2 (Activation)    (None, 10, 10, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               409728
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
_________________________________________________________________
activation_4 (Activation)    (None, 3)                 0
=================================================================
Total params: 419,379
Trainable params: 419,379
Non-trainable params: 0
_________________________________________________________________
```

  我們餵 action 給 env 後會拿到 observation、reward、done 等資訊，而我餵給 DL model 的是兩個 env_step 回傳的 observation 的差值 (state)，輸出為動作（我有將重複的六個 action 簡化成三個，所以 DL model 最後輸出只有三維）

  train 時，要 make action 時，會先取出 model prediction 的機率分佈，然後再根據這個機率分佈 sample 出一個動作出來

  （test 時，則是直接取機率最大的）

然後為了使可以獲得比較好的 reward 的 action，下次被選中的機率更高，我們從真正選擇的 action 與 model predict action 的機率分佈算出 gradient，並將這個 gradient 乘上該次的 discounted-normalized reward，再將之乘上一個 learning rate 後加回原本的 model predict action 機率分佈當作 model 要吃的 label

最後我的 model 會在每次 done 時，將此次 episode 視為一個 mini-batch，將其中 step 的 state （observation 差）當作 X，以及上面所說的 label 當作 y 來 train

[DQN]
我的 DL model 如下，大致與助教相同
3 CNN + 1DNN 最後 output
CNN Activation 使用 relu，DNN Activation 使用 LeakyReLU
optimizer 為 Adam (learning rate = 1e-4)，loss 為 mse （for Q 值）

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 1, 21, 32)         172064
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 1, 11, 64)         32832
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 1, 11, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 704)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               360960
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 2052
=================================================================
Total params: 604,836
Trainable params: 604,836
Non-trainable params: 0
_________________________________________________________________
```

首先我們會有兩個架構相同的 model，onlineQ 和 target Q
targetQ 是代表隨著 step 不斷在更新的 Q-table，而我們會每隔一段時間用 targetQ 去 update onlineQ

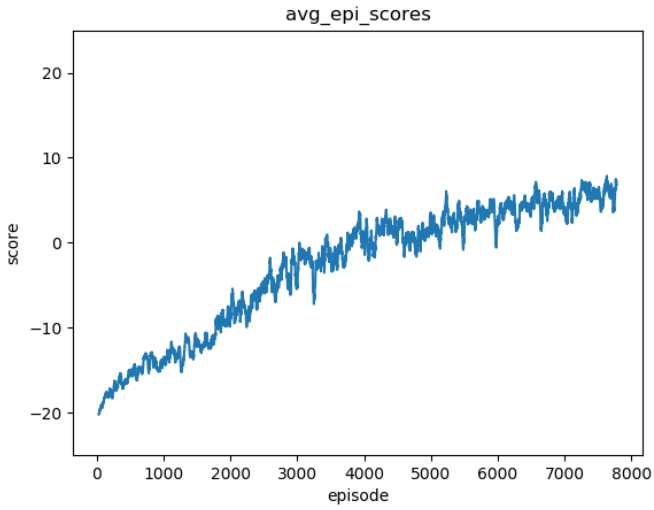
而為了實現 Experience Replay，會有一 memory，來記錄要 training 的資料，而當該 memory 滿了便會要開始做 training，並從 memory 中 sample 出 batch_size 個來做，這樣就可以斷開時間關聯性對 model 造成的影響

而這些要 training 的資料會先經過 targetQ 去做 prediction 得到目標 Q 值

表，再將這組資料與目標 Q 值（reward + (1-terminal)*gamma*max(Q)）表餵給 onlineQ 去做 training，以使得 onlineQ 逼近 target

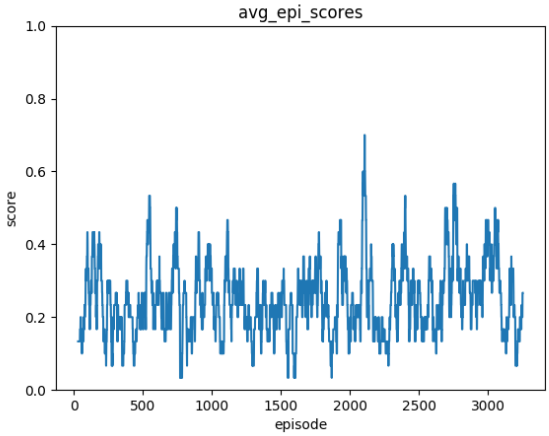
另外，我們會有一個 exploration 的前期，在這個時期的動作會根據一個隨 step 下降的 exploration_factor 來決定他要是 random 的 action 還是 online Q 決定出來的 action
(沒 train 出來 …)

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



（用 macbook train 了 三天半 …）
(橫軸為 episode，縱軸為前 30 個 episode 的 total reward average)

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



(橫軸為 episode，縱軸為前 30 個 episode 的 total reward average)

- **Experience with DQN hyperparameters**