DSAI
Classic control in reinforcement learning
Classic control: Mountain Car

F44026149 呂伯駿

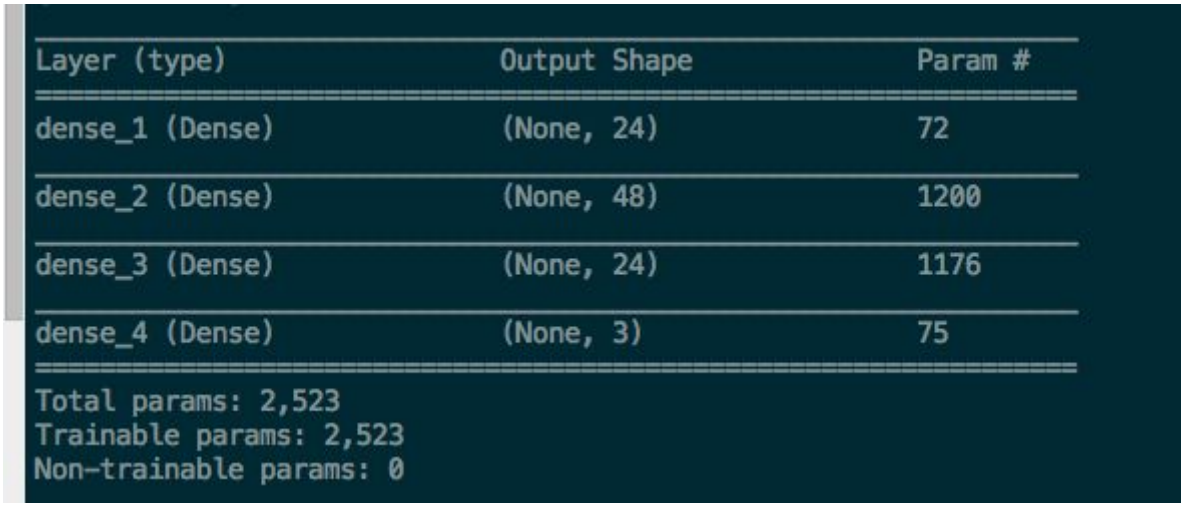
1. Report

A. Overview

方法

使用 Deep Q Learning 2015，包含 Target Network。

Evaluation Network 與 Target Network 各使用四層 Fully Connect 層
Activation function 除輸出層外皆使用Relu
Loss 採用 mean_squared_error



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 24)	72
dense_2 (Dense)	(None, 48)	1200
dense_3 (Dense)	(None, 24)	1176
dense_4 (Dense)	(None, 3)	75

Total params: 2,523
Trainable params: 2,523
Non-trainable params: 0

參數

```
MEMORY_SIZE = 2000
BATCH_SIZE = 32
GAMMA = 0.85
EPSILON = 1.0
EPSILON_MIN = 0.01
EPSILON_DECAY = 0.995
ALPHA = 0.005
ALPHA_DECAY = 0.001

N_EPISODE = 50
UPDATE_TARGET_PERIOD = 1
START_REPLAY_EPISODE = 0
```

UPDATE_TARGET_PERIOD
指的是每幾個Episode要更新Target Network的值，這裡設為1表示每一個Episode結束時都會更新一次。

START_REPLAY_EPISODE
指的是從第幾個Episode之後要開始Experience Replay，這裡設為0表示從一開始便Replay。

根據下方Reward設定後的成果來看，不需要特別設定上述兩個值即可跑出不錯的效果，即使設定了也因為現在Reward效果已經不錯對結果並沒有顯著的改善，這是上述兩個數值設為 0 與 1 的原因。

EPISLON_DECAY
這裡設定為每次 Step 都會 Decay一次。

```
if(i_episode > START_REPLAY_EPISODE):
    agent.decay_epsilon()
    agent.replay()
    if(i_episode % UPDATE_TARGET_PERIOD == 0):
        agent.train_target()
```

Reward 設計

若車子加速的方向與車子下一個時間點的速度方向相同則給予較多的Reward。
除此之外Reward 還會加上與中心點的距離。

Reward皆設為負數的原因是避免出現 “不斷擺盪不會到終點” 的情況，訓練其能夠已最少步數到達終點。

```
def set_reward(new_state, action_):
    position, velocity = new_state
    if (action_ == 2 and velocity > 0):
        reward = -1
    elif(action_ == 0 and velocity < 0):
        reward = -1
    else:
        reward = -5

    if (position - (-0.5) > 0):
        reward += abs(position - (-0.5))

    return reward
```

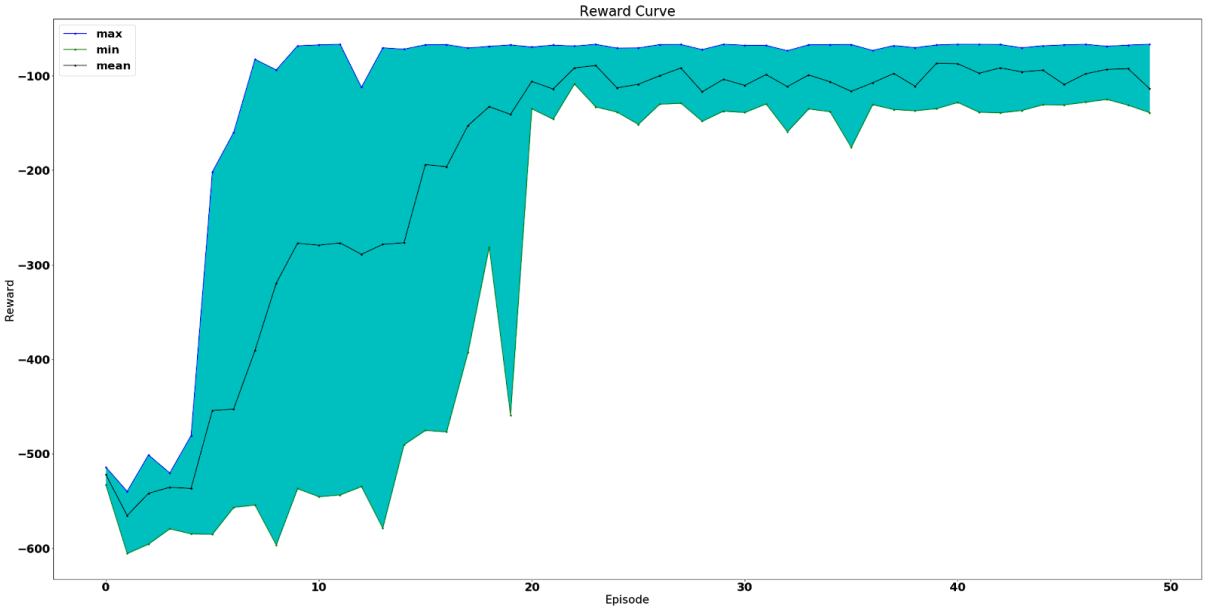
輸出動作調整

這裡為提升效果，調整其動作輸出使其不會輸出”不加速也不減速”的動作。

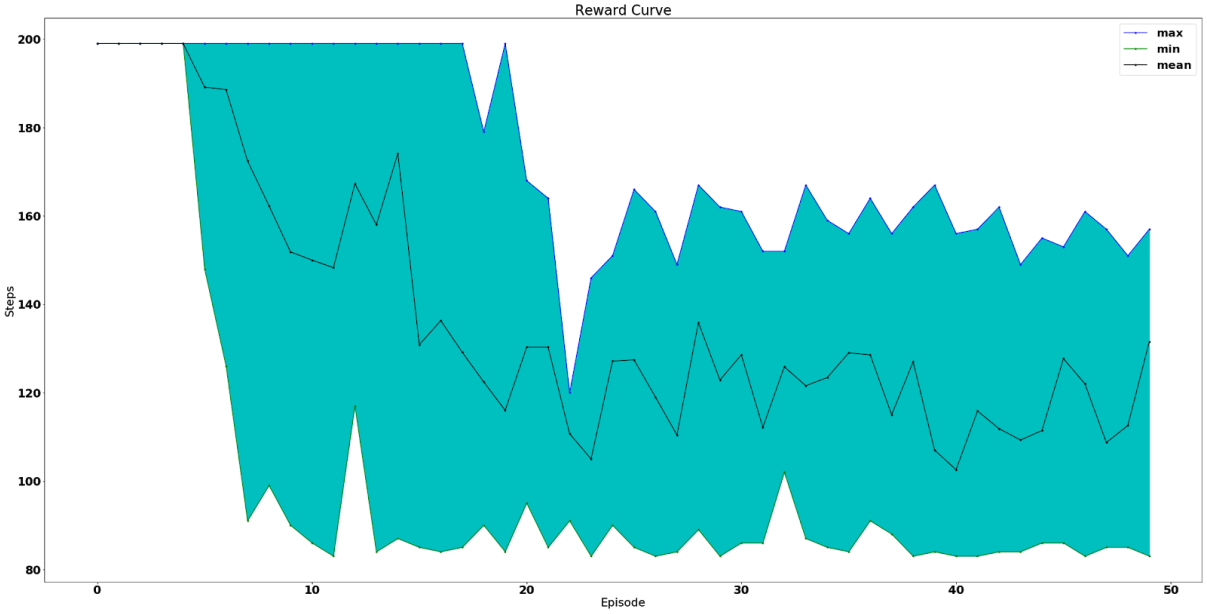
```
  def choose_action(self, state):
      if np.random.random() < self.epsilon:
          return np.random.choice([0, 1])
      else:
          # choose action by past model
          return np.argmax(self.model.predict(state)[0])
```

實驗結果

依據上述Reward設定，實驗結果如下(跑7次的結果)，平均 10 個Episode Reward 便可上升達穩定值。



而達成步數如下，即使不調整部署上限(199)仍可訓練完成，最終可維持在 83~150 左右的達成步數。 步數83目測(env.render)已十分接近此環境最優步數。



僅顯示單一次數據之Steps結果

B. 其他實驗

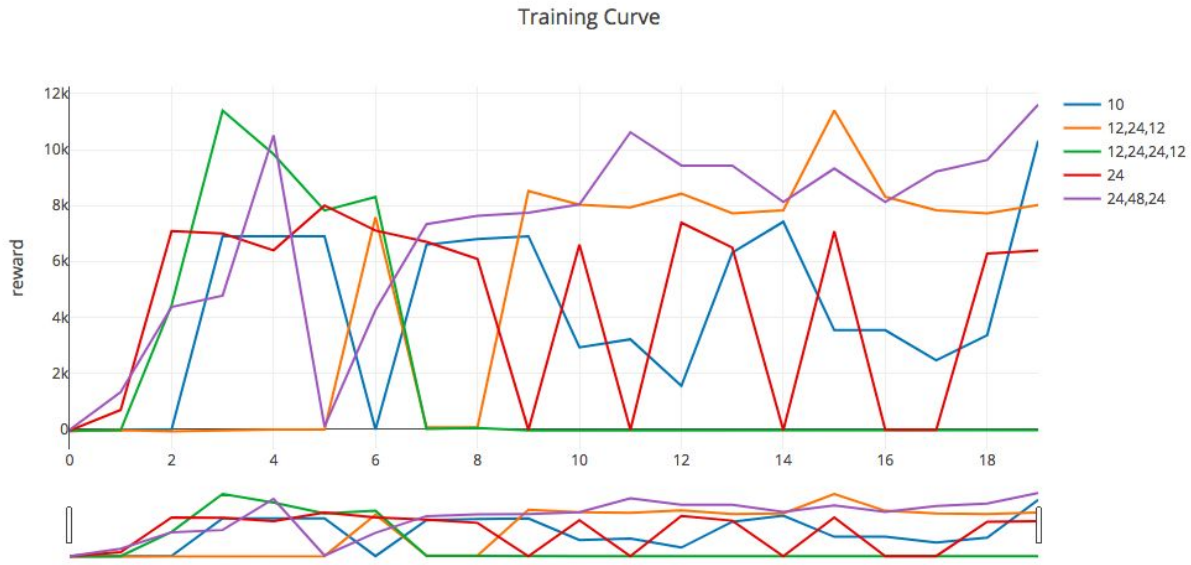
調整 Layer 設定

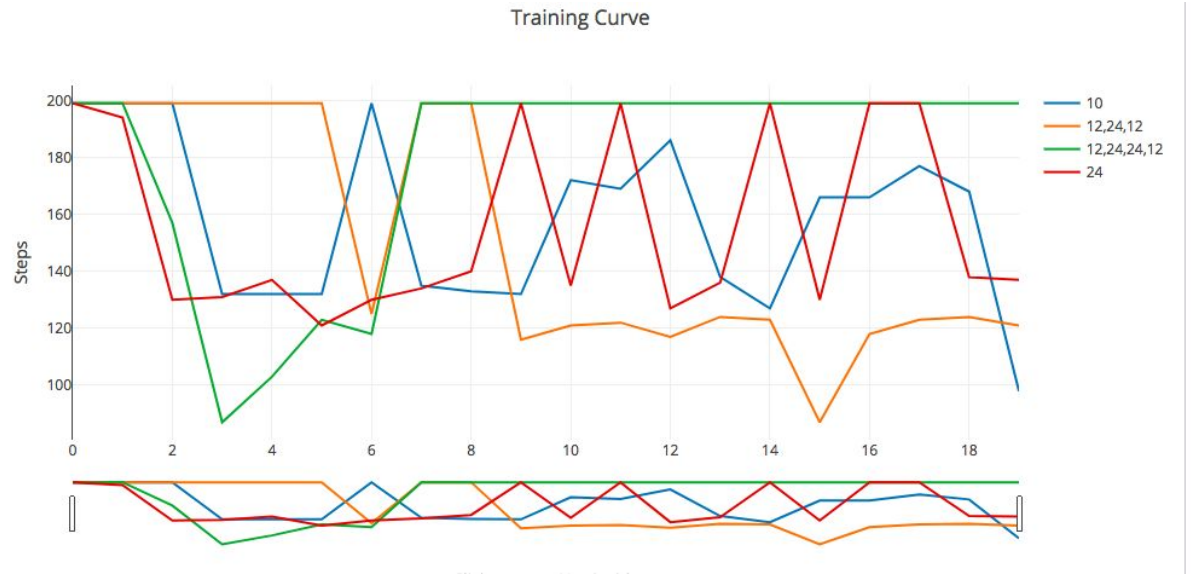
此區Reward設定如下

```
position, velocity = new_state
if (action_ == 2 and velocity > 0):
    reward = 1
elif(action_ == 0 and velocity < 0):
    reward = 1
else:
    reward = -2

if (position - (-0.5) > 0):
    reward += abs(position - (-0.5))

if (position > 0.5):
    reward = (200-steps) * 100
```

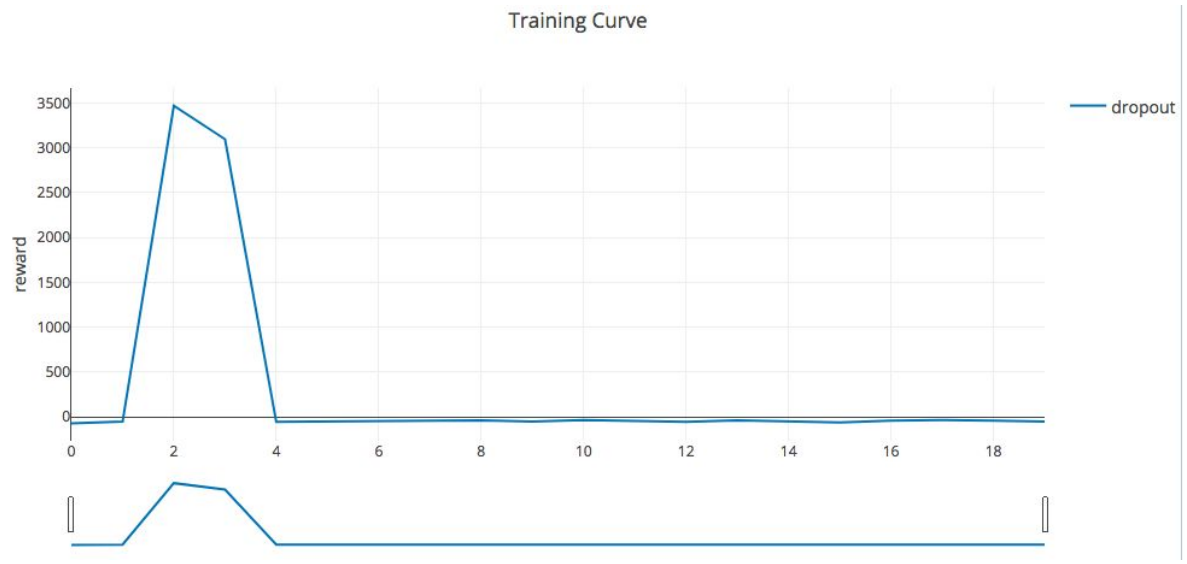
1) 2層，第一層Output 10 個 neuron，第二層為輸出層
2) 4層，每層分別有 12,24,12,3 neuron 輸出
3) 5層，每層分別有 12,24,24,12,3個 neuron
4) 2層，第一層輸出有 24個 neuron
5) 4 層，現行設定，每層分別有 24,48,24,3個neuron輸出

Reward 與 Step 結果如下，可以看到兩層的NN(藍線與紅線)的結果並不穩定，而五層NN在20Episode內並沒有Train成功(仍是199步)。12,24,12,3的NN效果則略低於現行設定。





此外也嘗試多新增一層Dropout (0.5)，或許是因為環境複雜度的緣故，可見模型效果明顯變差。



2. Discuss

1. What kind of RL algorithms did you use? value-based, policy-based, model-based? why?
A: Value-Based，輸出Q Value 專注於當下行動，而非輸出機率調整整體策略。

2.This algorithms is off-policy or on-policy? why?
A: off-policy，因為是基於Q Value Function 於每一步輸出Argmax的行動。 此外也可透過 Experience Replay 學習過去經驗。

3.How does your algorithm solve the correlation problem in the same MDP?
A: 可以，因為DQN的學習方式是先將學習結果放在 Memory 中，後續再隨機取出學習，因此可以避免 Correlation Problem。