AI Lab 8 Report

113368545 陳昀鴻
113568519 吳孟軒

TAIPEI TECH

Open AI gym for Lab 8 and 9

OpenAI Gym is a tool that provides many test environments, so that everyone can have a common environment to test their own RL algorithms, instead of spending time to build their own test environment.

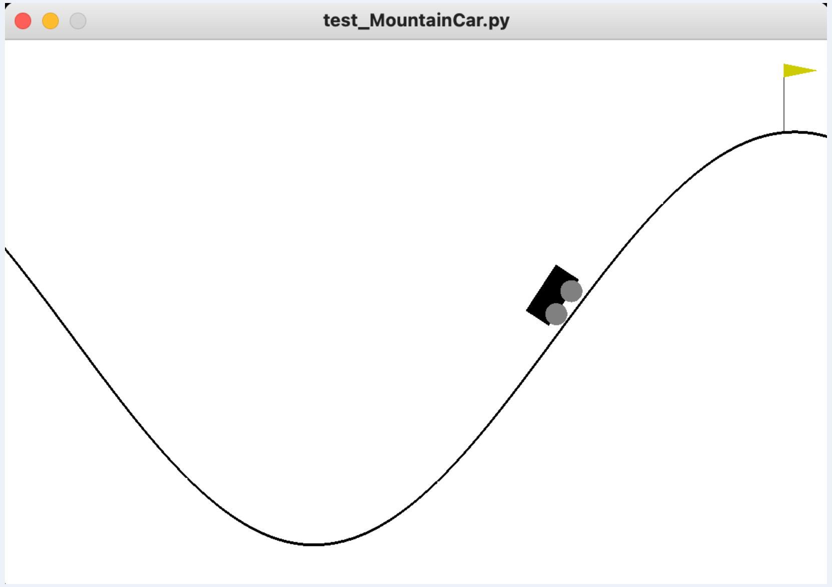
Run Lab08

```
 ~/Desktop/Yun/NTUT_ML/LAB_code/NTUT_ML_LAB08  on  main  ll
 rwxr-xr-x  11  chenyunhong  staff   352 B    Fri Nov  8 15:26:33 2024  after/
 rwxr-xr-x   6  chenyunhong  staff   192 B    Wed Nov  6 14:30:09 2024  before/
 rw-r--r--   1  chenyunhong  staff     2 KiB  Fri Nov  8 18:27:22 2024  README.md
```

```
 ~/De/Y/N/LAB_code/NTUT_ML_LAB08/after  on  main  conda activate py36
 ~/De/Y/N/LAB_c/NTUT_ML_LAB08/after  on  main  python3 test_MountainCar.py
```

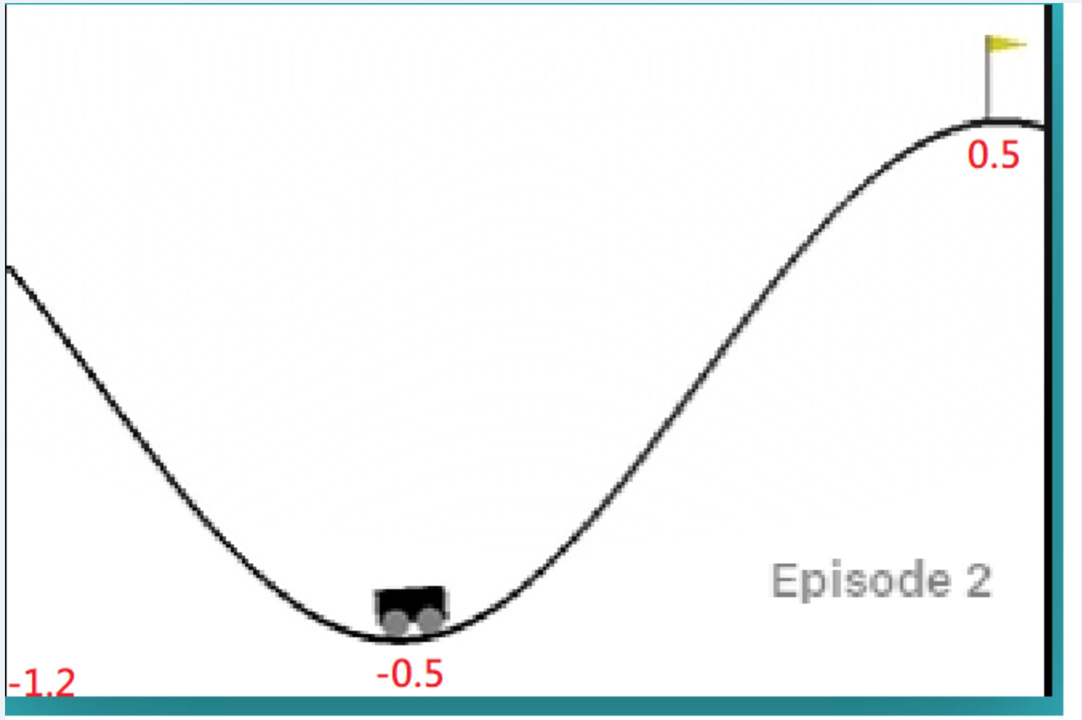
```
 ~/De/Y/N/LAB_c/NTUT_ML_LAB08/after  on  main  ll
 rwxr-xr-x   4  chenyunhong  staff   128 B    Fri Nov  8 18:06:40 2024  __pycache__/
 rwxr-xr-x  43  chenyunhong  staff     1 KiB  Fri Nov  8 18:24:49 2024  logs/
 rwxr-xr-x   6  chenyunhong  staff   192 B    Fri Nov  8 18:25:15 2024  saved_model/
 rw-r--r--   1  chenyunhong  staff   103 KiB  Fri Nov  8 18:25:38 2024  Figure_1.png
 rw-rw-r--   1  chenyunhong  staff    11 KiB  Fri Nov  8 15:17:00 2024  RL_brain.py
 rw-r--r--   1  chenyunhong  staff   717 B    Wed Nov  6 18:01:56 2024  requirements.txt
 rw-r--r--   1  chenyunhong  staff  1007 B    Fri Nov  8 18:06:23 2024  reward_functions.py
 rw-rw-r--   1  chenyunhong  staff     3 KiB  Fri Nov  8 18:24:46 2024  run_MountainCar.py
 rw-r--r--   1  chenyunhong  staff     2 KiB  Fri Nov  8 18:33:16 2024  test_MountainCar.py
```

Folder Structure



RL_brain.py：DQN 代理實作
run_MountainCar.py：主要訓練腳本
test_MountainCar.py：測試腳本
reward_functions.py：reward function 獨立出來
saved_model/：訓練過後的模型存放在內
logs/：log 資訊

Mountain Car

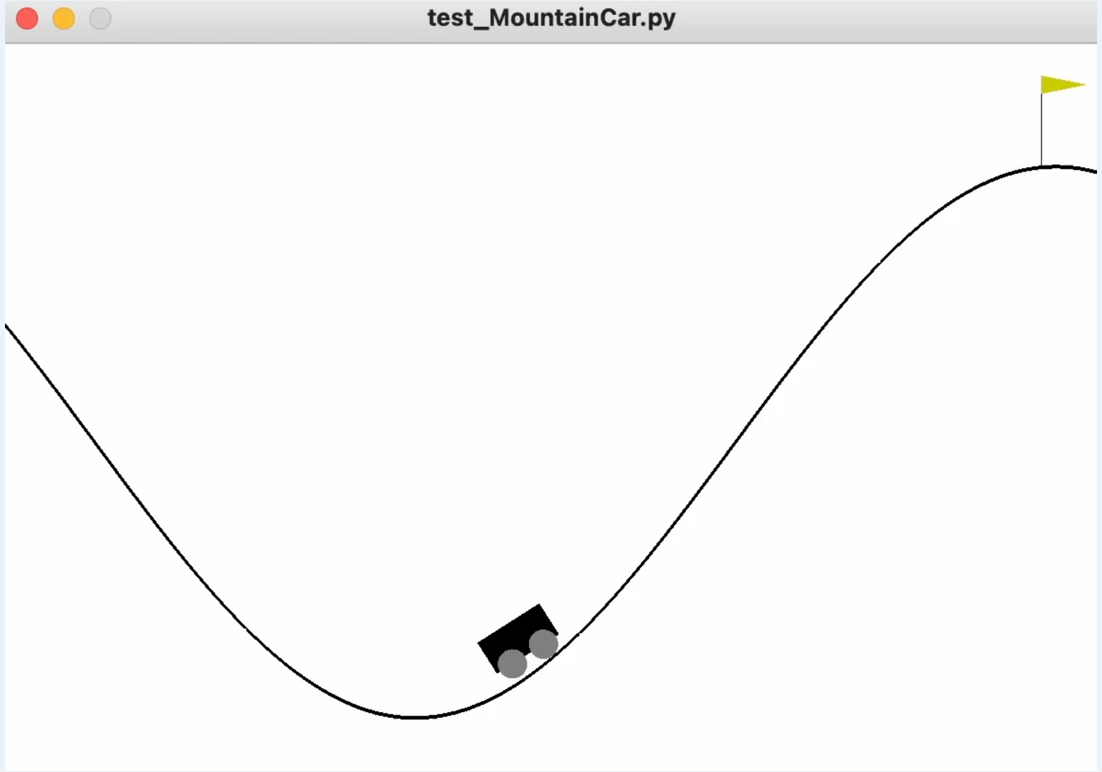


- Action: There are 3 actions: forward 2, motionless 1 and backward 0.
- States: 2, position and velocity. The value of position is about -0.5 at the lowest point, the top of the slope on the left is -1.2, the height position corresponding to it on the right is 0, and the position of the small yellow flag = 0.5.
- Reward: Every time you move, you will get a reward of -1 until the car reaches the yellow flag position 0.5

實驗影片

```
[2024-11-12 19:05:36,398] Making new env: MountainCar-v0
模型已從 ./saved_model/model.ckpt 加載
成功載入模型
回合 1: 步數 = 115, 最終位置 = 0.527, 最終速度 = 0.049, 總獎勵 = 455.3, 是否成功 = 是
回合 2: 步數 = 114, 最終位置 = 0.517, 最終速度 = 0.047, 總獎勵 = 450.5, 是否成功 = 是
回合 3: 步數 = 144, 最終位置 = 0.537, 最終速度 = 0.050, 總獎勵 = 691.4, 是否成功 = 是
回合 4: 步數 = 111, 最終位置 = 0.527, 最終速度 = 0.039, 總獎勵 = 448.2, 是否成功 = 是
回合 5: 步數 = 115, 最終位置 = 0.526, 最終速度 = 0.049, 總獎勵 = 455.5, 是否成功 = 是
回合 6: 步數 = 86, 最終位置 = 0.508, 最終速度 = 0.022, 總獎勵 = 412.8, 是否成功 = 是
回合 7: 步數 = 113, 最終位置 = 0.526, 最終速度 = 0.043, 總獎勵 = 451.3, 是否成功 = 是
回合 8: 步數 = 111, 最終位置 = 0.537, 最終速度 = 0.050, 總獎勵 = 482.2, 是否成功 = 是
回合 9: 步數 = 86, 最終位置 = 0.515, 最終速度 = 0.022, 總獎勵 = 414.0, 是否成功 = 是
回合 10: 步數 = 112, 最終位置 = 0.537, 最終速度 = 0.050, 總獎勵 = 461.8, 是否成功 = 是

測試結果統計：
成功率: 100.0%
平均步數: 110.7
最少步數: 86
最多步數: 144
```

Log Info



Program RL_brain.py

DQN 使用三層神經網絡：

> **這種三層結構特別適合 MountainCar 問題，因為：**

- 需要理解狀態空間中的複雜模式
- 需要學習長期規劃（爬山需要先往後退）
- 需要整合位置和速度信息來做出決策

> **而且實驗結果表明，這種結構確實能帶來：**

- 更快的學習速度
- 更穩定的訓練過程
- 更好的最終性能

- 輸入層：狀態特徵（位置和速度）
- 隱藏層 1：64 個節點，使用 ELU 激活函數
- 隱藏層 2：32 個節點，使用 ELU 激活函數
- 輸出層：每個動作的 Q 值

> **調整重點**

- 梯度裁剪以提高穩定性
- 將 ReLU 改為 ELU 激活函數
- 原始 ReLU 版本可能在訓練初期表現不穩定
- ELU 版本通常能提供：
- 更快的收斂速度
- 更穩定的學習曲線
- 更好的最終性能
- 模型保存/加載功能
- 增強的視覺化工具(plot 優化)

Program RL_brain.py

```python
# Deep Q Network off-policy
class DeepQNetwork:
    def __init__(
            self,
            n_actions,          # 動作空間大小
            n_features,         # 狀態特徵數量
            learning_rate=0.01,         # 學習率
            reward_decay=0.9,           # 獎勵衰減率
            e_greedy=0.9,               # epsilon-貪婪策略中的 epsilon 最大值
            replace_target_iter=100,    # 目標網絡更新間隔
            memory_size=5000,           # 記憶體大小
            batch_size=32,              # 批次大小
            e_greedy_increment=0.002,   # 探索率增長
            output_graph=False,         # 是否輸出計算圖
    ):
```

Program RL_brain.py

```python
# ------------------ build evaluate_net ------------------
self.s = tf.placeholder(
    tf.float32, [None, self.n_features], name='s')  # 輸入
self.q_target = tf.placeholder(
    tf.float32, [None, self.n_actions], name='Q_target')  # 用於計算損失

with tf.variable_scope('eval_net'):
    # c_names 是用於存儲變量的集合
    c_names, n_l1, n_l2, w_initializer, b_initializer = \
        ['eval_net_params', tf.GraphKeys.GLOBAL_VARIABLES], 64, 32, \
        tf.random_normal_initializer(0., 0.3), tf.constant_initializer(0.1)

    # 第一層使用 ELU 激活函數
    with tf.variable_scope('l1'):
        w1 = tf.get_variable('w1', [self.n_features, n_l1], initializer=w_initializer, collections=c_names
        b1 = tf.get_variable('b1', [1, n_l1], initializer=b_initializer, collections=c_names)
        l1 = tf.nn.elu(tf.matmul(self.s, w1) + b1)

    # 新增一層
    with tf.variable_scope('l2'):
        w2 = tf.get_variable('w2', [n_l1, n_l2], initializer=w_initializer, collections=c_names)
        b2 = tf.get_variable('b2', [1, n_l2], initializer=b_initializer, collections=c_names)
        l2 = tf.nn.elu(tf.matmul(l1, w2) + b2)

    # 輸出層
    with tf.variable_scope('l3'):
        w3 = tf.get_variable('w3', [n_l2, self.n_actions], initializer=w_initializer, collections=c_names)
        b3 = tf.get_variable('b3', [1, self.n_actions], initializer=b_initializer, collections=c_names)
        self.q_eval = tf.matmul(l2, w3) + b3

with tf.variable_scope('loss'):
    self.loss = tf.reduce_mean(
        tf.squared_difference(self.q_target, self.q_eval))
```

Program RL_brain.py

```
with tf.variable_scope('train'):
    # 添加梯度裁剪
    optimizer = tf.train.RMSPropOptimizer(self.lr)
    gradients = optimizer.compute_gradients(self.loss)
    capped_gradients = [(tf.clip_by_value(grad, -1., 1.), var) for grad, var in gradients if grad is not N
    self._train_op = optimizer.apply_gradients(capped_gradients)

# ------------------ build target_net ------------------
self.s_ = tf.placeholder(
    tf.float32, [None, self.n_features], name='s_')    # input
with tf.variable_scope('target_net'):
    # c_names(collections_names) are the collections to store variables
    c_names = ['target_net_params', tf.GraphKeys.GLOBAL_VARIABLES]

    # 第一層
    with tf.variable_scope('l1'):
        w1 = tf.get_variable('w1', [self.n_features, n_l1], initializer=w_initializer, collections=c_names
        b1 = tf.get_variable('b1', [1, n_l1], initializer=b_initializer, collections=c_names)
        l1 = tf.nn.elu(tf.matmul(self.s_, w1) + b1)

    # 新增中間層，與 eval_net 保持一致
    with tf.variable_scope('l2'):
        w2 = tf.get_variable('w2', [n_l1, n_l2], initializer=w_initializer, collections=c_names)
        b2 = tf.get_variable('b2', [1, n_l2], initializer=b_initializer, collections=c_names)
        l2 = tf.nn.elu(tf.matmul(l1, w2) + b2)

    # 輸出層
    with tf.variable_scope('l3'):
        w3 = tf.get_variable('w3', [n_l2, self.n_actions], initializer=w_initializer, collections=c_names)
        b3 = tf.get_variable('b3', [1, self.n_actions], initializer=b_initializer, collections=c_names)
        self.q_next = tf.matmul(l2, w3) + b3
```

Program reward_functions.py

對於狀態 $s_t = (x_t, \dot{x}t)$，其中：

- x_t 是位置，範圍 $[-1.2, 0.6]$
- $\dot{x}t$ 是速度，範圍 $[-0.07, 0.07]$

獎勵函數 $R(s_t)$ 由以下幾個部分組成：

- 位置獎勵 $R_{position}$：

$$R_{position} = 5.0 \cdot \frac{x_t - (-1.2)}{0.6 - (-1.2)}$$

- 速度獎勵 $R_{velocity}$：

$$R_{velocity} = 3.0 \cdot \frac{|\dot{x}t|}{0.07} \cdot \begin{cases} 2.0 & \text{if } \dot{x}t > 0 \text{ and } x_t > -0.4 \\ 1.0 & \text{otherwise} \end{cases}$$

- 接近頂部獎勵 R_{top}：

$$R_{top} = \begin{cases} 1.5 \cdot (R_{position} + R_{velocity}) & \text{if } x_t \geq -0.2 \\ R_{position} + R_{velocity} & \text{otherwise} \end{cases}$$

- 目標獎勵 R_{goal}：

$$R_{goal} = \begin{cases} 20.0 & \text{if } x_t \geq 0.5 \\ R_{top} & \text{otherwise} \end{cases}$$

- 低位置懲罰 $R_{penalty}$：

$$R_{penalty} = \begin{cases} -1.0 & \text{if } |\dot{x}t| < 0.001 \text{ and } x_t < -0.8 \\ 0 & \text{otherwise} \end{cases}$$

最後，總獎勵被限制在 [-3, 20] 範圍內：

$$R_{final}(s_t) = \text{clip}(R_{goal} + R_{penalty}, -3, 20)$$

> **調整重點**

這個優化後的獎勵函數特點：

- 提供更強的位置獎勵（權重為 5.0）
- 在上坡時給予雙倍速度獎勵
- 接近頂部時（x ≥ -0.2）提供 1.5 倍獎勵
- 達到目標時給予較大獎勵（20.0）
- 懲罰在低位置停滯不前的情況
- 更大的獎勵範圍（-3 到 20）以提供更強的學習信號

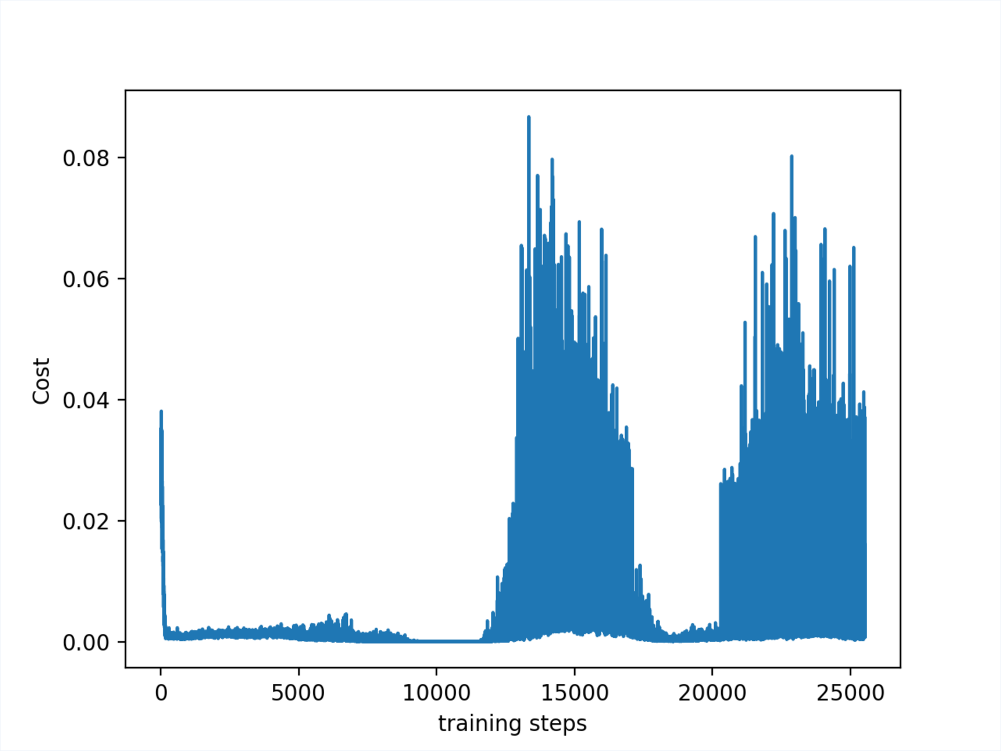
Program reward_functions.py

```python
def calculate_mountain_car_reward(position, velocity, goal_position):
    """
    優化後的獎勵函數，提供更強的位置和速度獎勵
    """
    reward = 0

    # 增加基於位置的獎勵
    position_reward = (position - (-1.2)) / (0.6 - (-1.2))
    reward += 5.0 * position_reward  # 增加位置獎勵的權重

    # 優化速度獎勵
    velocity_reward = abs(velocity) / 0.07
    if velocity > 0 and position > -0.4:
        velocity_reward *= 2.0  # 在上坡時給予更多速度獎勵
    reward += 3.0 * velocity_reward

    # 特殊位置的額外獎勵
    if position >= -0.2:
        reward *= 1.5  # 接近頂部時給予額外獎勵

    # 達到目標的獎勵
    if position >= goal_position:
        reward = 20.0  # 增加目標獎勵

    # 懲罰在低位置停留
    if abs(velocity) < 0.001 and position < -0.8:
        reward -= 1.0

    # 限制獎勵範圍
    reward = np.clip(reward, -3, 20)

    return reward
```

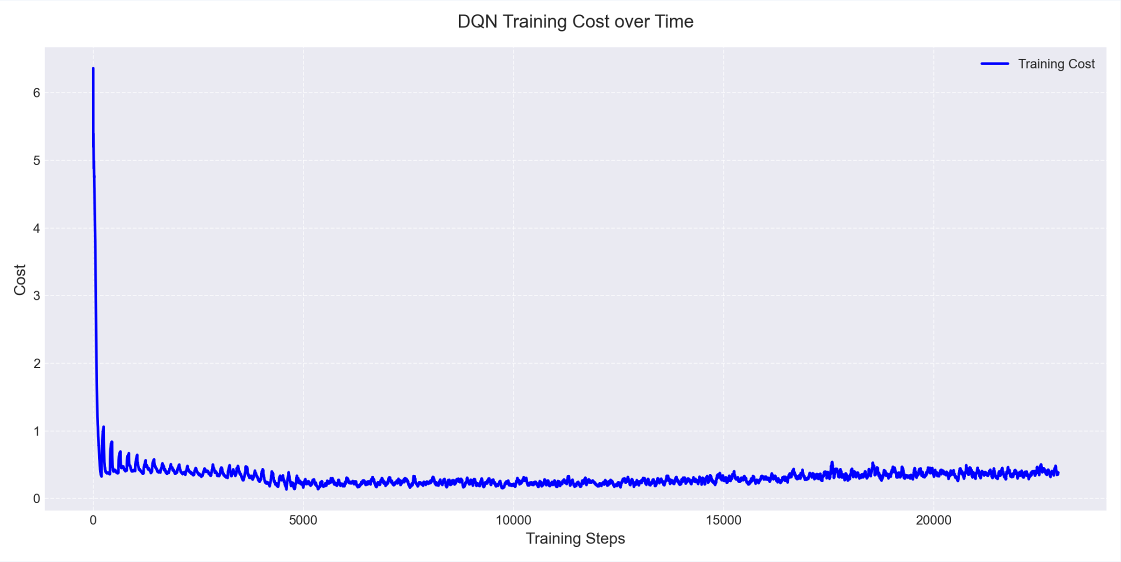
Program: run_MountainCar.py

```python
# 主要訓練循環
for i_episode in range(100):  # 增加訓練回合數
    observation = env.reset()     # 重置環境
    ep_r = 0                      # 初始化每集的總獎勵
    steps_in_episode = 0         # 新增：記錄每回合的步數
    max_steps = 1000  # 增加最大步數限制

    while True:  # 使用步數限制
        # env.render()  # 註釋掉訓練時的渲染
        action = RL.choose_action(observation)  # 根據當前狀態選擇動作

        # 執行動作並獲取下一狀態、獎勵和結束標記
        observation_, reward, done, info = env.step(action)

        position, velocity = observation_

        # 使用導入的獎勵函數
        reward = calculate_mountain_car_reward(position, velocity, env.unwrapped.goal_position)

        if position >= env.unwrapped.goal_position:
            done = True

        # 儲存當前的轉換 (狀態, 動作, 獎勵, 新狀態)
        RL.store_transition(observation, action, reward, observation_)

        if total_steps > 1000:  # 更頻繁地學習
            RL.learn()

        ep_r += reward
        steps_in_episode += 1     # 新增：步數計數

        # FIXME: here has some problem
        if done:
            print(f'回合: {i_episode}, 步數: {steps_in_episode}, '
                  f'總獎勵: {ep_r:.1f}, 最終位置: {position:.3f}, '
                  f'最終速度: {velocity:.3f}, Epsilon: {RL.epsilon:.3f}')
            break

        # 更新狀態
        observation = observation_
        total_steps += 1
```

Loss function



Before



After

Ref

1. https://github.com/nitish-kalan/MountainCar-v0-Deep-Q-Learning-DQN-Keras
2. https://ithelp.ithome.com.tw/articles/10247712
3. https://exp-blog.com/ai/gym-bi-ji-04-mountaincar/
4. https://blog.csdn.net/weixin_42454034/article/details/111194389
5. https://steemit.com/cn-stem/@hongtao/q-learning-mountaincar

Project Github:
https://github.com/roger28200901/NTUT_ML_LAB08