國立清華大學
National Tsing Hua University

INNOVATION ENGINEERING LAB

IIE Final Project

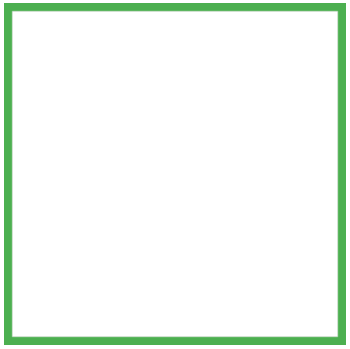
Q-Learning of Snake-Game

Presenter：溫致淵

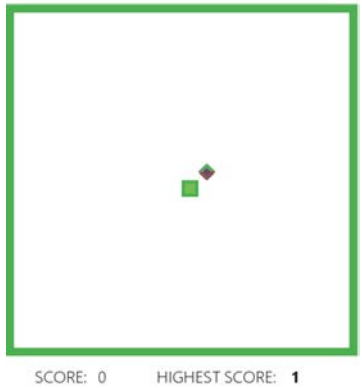
目標

- 透過建立Q-Learning環境模型，利用非監督式增強學習技術建立貪吃蛇遊戲的環境數值，目標使得貪吃蛇分數最大化。

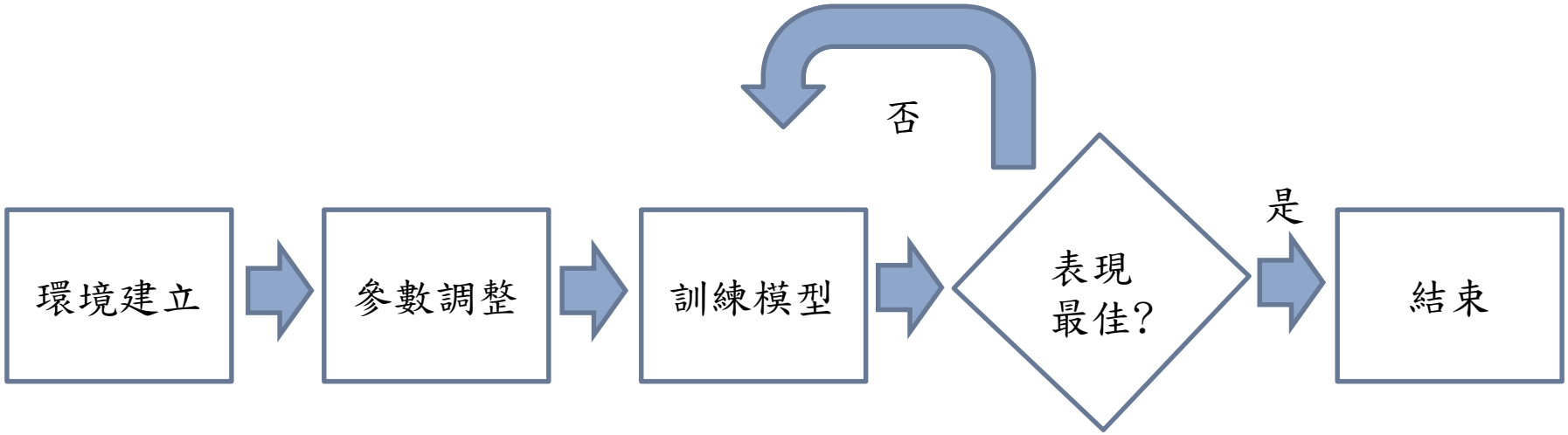
- 環境:



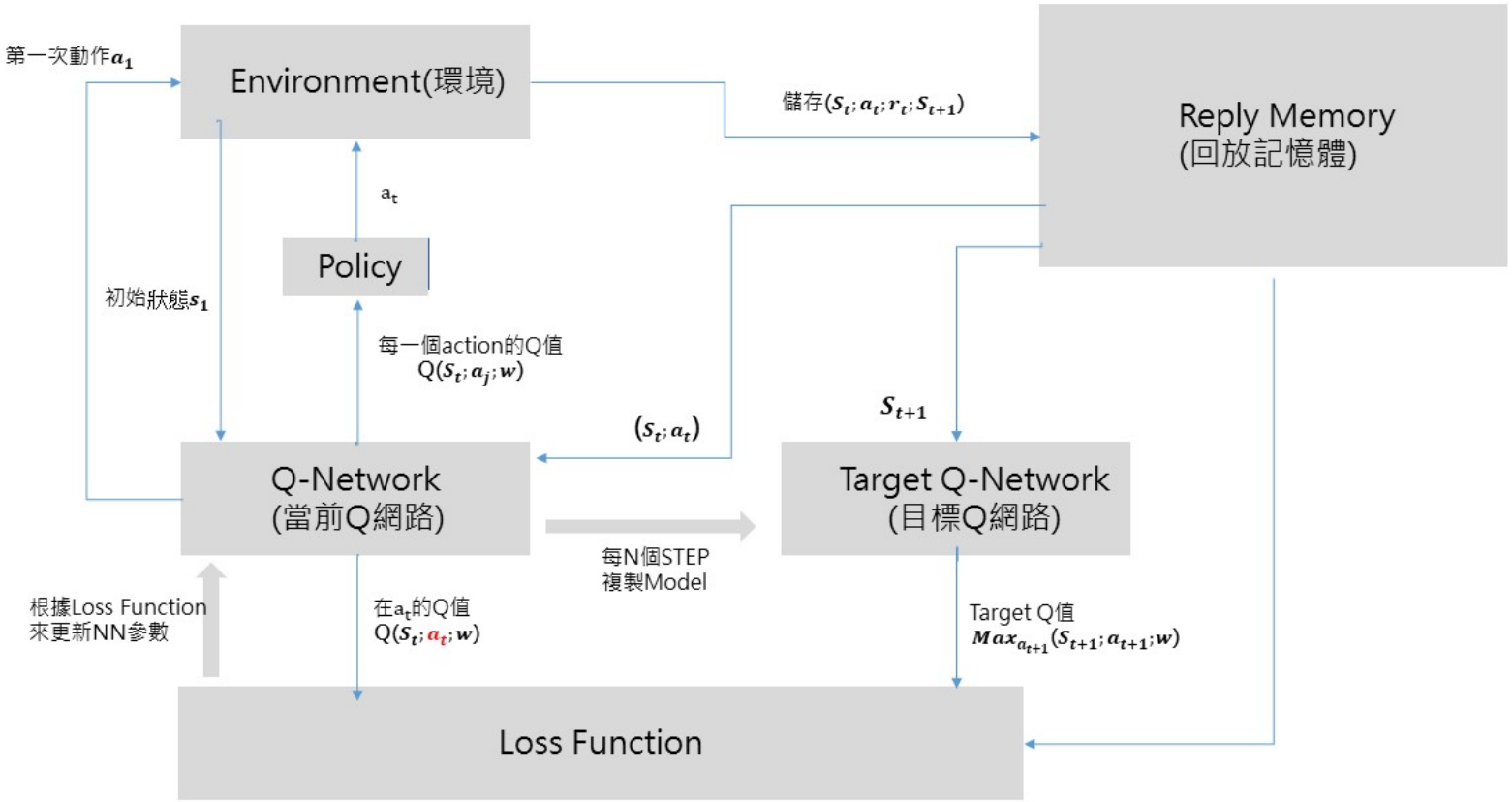
目標:



流程



Q-Learning 架構



Q-Learning 架構

- 設置gamma參數，負責貪吃蛇要在多長時間內獲得獎勵，範圍在0到1之間。(愈接近1愈考慮未來獎勵)

```
class DQNAgent(object):

    def __init__(self):
        self.reward = 0  #0
        self.gamma = 0.9    #0.9
        self.dataframe = pd.DataFrame()
        self.short_memory = np.array([])
        self.agent_target = 1
        self.agent_predict = 0
        self.learning_rate = 0.0005
        self.model = self.network()
        #self.model = self.network("weights.hdf5")
        self.epsilon = 0
        self.actual = []
        self.memory = []
```

Q-Learning 架構

- Self.reward參數，負責貪吃蛇的自身獎懲，範圍在吃到獎勵與自毀懲罰之間。

```
class DQNAgent(object):

    def __init__(self):
        self.reward = 0  #0
        self.gamma = 0.9   #0.9
        self.dataframe = pd.DataFrame()
        self.short_memory = np.array([])
        self.agent_target = 1
        self.agent_predict = 0
        self.learning_rate = 0.0005
        self.model = self.network()
        #self.model = self.network("weights.hdf5")
        self.epsilon = 0
        self.actual = []
        self.memory = []
```

```
def set_reward(self, player, crash):
    self.reward = 0  #0
    if crash:
        self.reward = -10    #-10
        return self.reward
    if player.eaten:
        self.reward = 10    #10
    return self.reward
```

Q-Learning 架構

- Learning Rate參數，決定找出全域最低點的步伐大小，LR設置太大或太小皆會影響最後的結果

```
class DQNAgent(object):

    def __init__(self):
        self.reward = 0  #0
        self.gamma = 0.9   #0.9
        self.dataframe = pd.DataFrame()
        self.short_memory = np.array([])
        self.agent_target = 1
        self.learning_rate = 0.0005
        #self.model = self.network("weights.hdf5")
        self.epsilon = 0
        self.actual = []
        self.memory = []
```

Q-Learning 架構

- Q-Table , 設定貪吃蛇向前/左/右(沒有後)時圖形與環境的改變，20為蛇身體的像素單位。

```
def get_state(self, game, player, food):

    state = [
        (player.x_change == 20 and player.y_change == 0 and ((list(map(add, player.position[-1], [20, 0])) in player.position) or
        player.position[-1][0] + 20 >= (game.game_width - 20))) or (player.x_change == -20 and player.y_change == 0 and ((list(map(add, player.position[-1], [-20, 0])) in player
        player.position[-1][0] - 20 < 20)) or (player.x_change == 0 and player.y_change == -20 and ((list(map(add, player.position[-1], [0, -20])) in player.position) or
        player.position[-1][-1] - 20 < 20)) or (player.x_change == 0 and player.y_change == 20 and ((list(map(add, player.position[-1], [0, 20])) in player.position) or
        player.position[-1][-1] + 20 >= (game.game_height-20))),  # danger straight

        (player.x_change == 0 and player.y_change == -20 and ((list(map(add,player.position[-1],[20, 0])) in player.position) or
        player.position[ -1][0] + 20 > (game.game_width-20))) or (player.x_change == 0 and player.y_change == 20 and ((list(map(add,player.position[-1],
        [-20,0])) in player.position) or player.position[-1][0] - 20 < 20)) or (player.x_change == -20 and player.y_change == 0 and ((list(map(
        add,player.position[-1],[0,-20])) in player.position) or player.position[-1][-1] - 20 < 20)) or (player.x_change == 20 and player.y_change == 0 and (
        (list(map(add,player.position[-1],[0,20])) in player.position) or player.position[-1][
         -1] + 20 >= (game.game_height-20))),  # danger right

         (player.x_change == 0 and player.y_change == 20 and ((list(map(add,player.position[-1],[20,0])) in player.position) or
         player.position[-1][0] + 20 > (game.game_width-20))) or (player.x_change == 0 and player.y_change == -20 and ((list(map(
         add, player.position[-1],[-20,0])) in player.position) or player.position[-1][0] - 20 < 20)) or (player.x_change == 20 and player.y_change == 0 and (
        (list(map(add,player.position[-1],[0,-20])) in player.position) or player.position[-1][-1] - 20 < 20)) or (
        player.x_change == -20 and player.y_change == 0 and ((list(map(add,player.position[-1],[0,20])) in player.position) or
        player.position[-1][-1] + 20 >= (game.game_height-20))), #danger left
```

前進

左轉

右轉

Q-Learning 架構

- NN-Model

```
def network(self, weights=None):
    model = Sequential()
    model.add(Dense(output_dim=120, activation='relu', input_dim=11))
    model.add(Dropout(0.15))
    model.add(Dense(output_dim=120, activation='relu'))
    model.add(Dropout(0.15))
    model.add(Dense(output_dim=120, activation='relu'))
    model.add(Dropout(0.15))
    model.add(Dense(output_dim=3, activation='softmax'))
    opt = Adam(self.learning_rate)
```

CNN parameter	Value
Number of convolution layers	3
Number of pooling layers	2 max-pooling
Number of fully connected layers	1
Activation function	ReLU
Regularization method	Dropout
Classification function of the output layer	Softmax

Q-Learning 架構

- Q-Learning記憶方式:Q（狀態，操作）= R（狀態，操作）+ Gamma * Max [Q（下一個狀態，具有最高Q值的操作）]

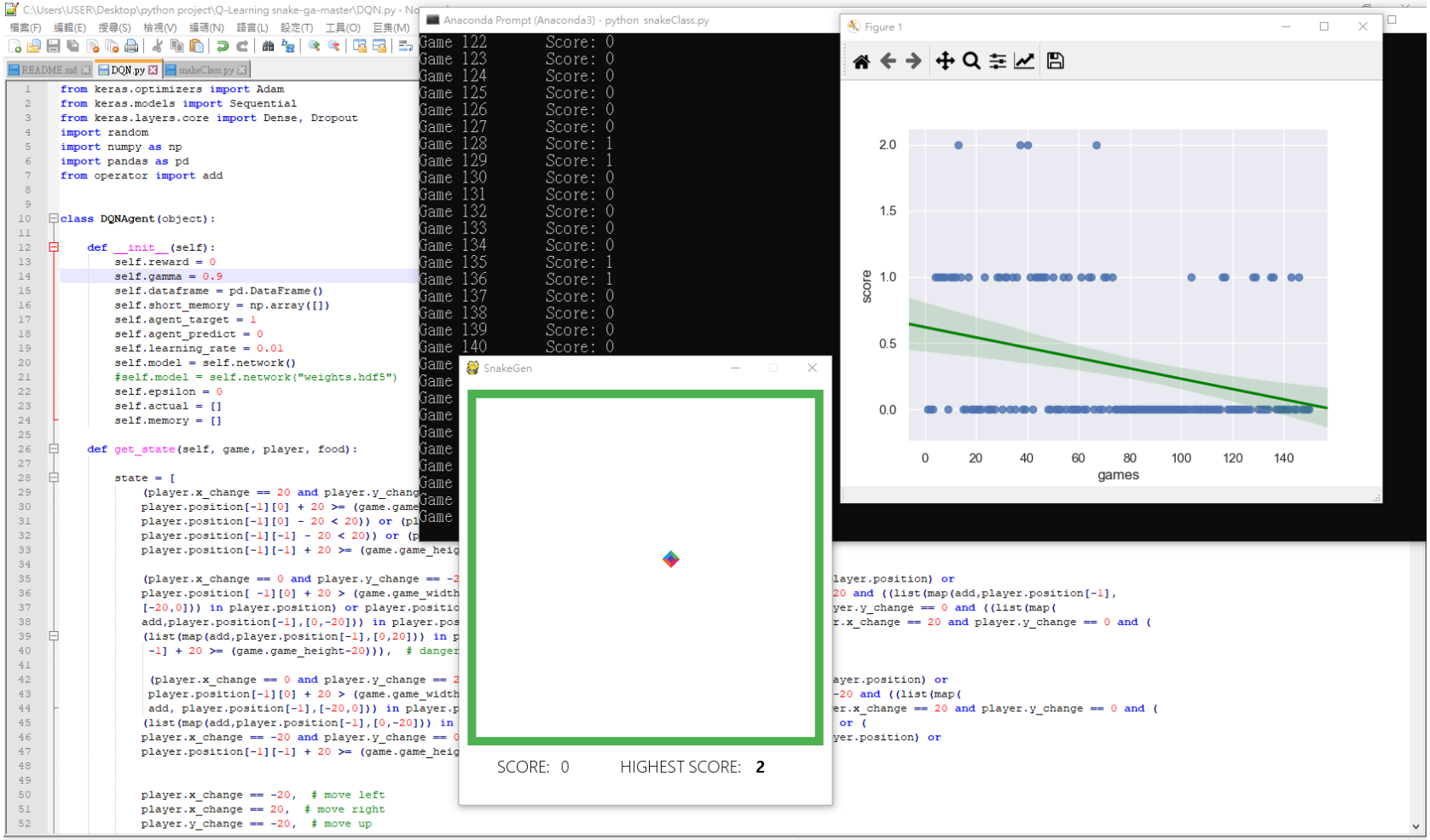
```
def remember(self, state, action, reward, next_state, done):
    self.memory.append((state, action, reward, next_state, done))

def replay_new(self, memory):
    if len(memory) > 1000:
        minibatch = random.sample(memory, 1000)
    else:
        minibatch = memory
    for state, action, reward, next_state, done in minibatch:
        target = reward
        if not done:
            target = reward + self.gamma * np.amax(self.model.predict(np.array([next_state]))[0])
        target_f = self.model.predict(np.array([state]))
        target_f[0][np.argmax(action)] = target
        self.model.fit(np.array([state]), target_f, epochs=1, verbose=0)
```

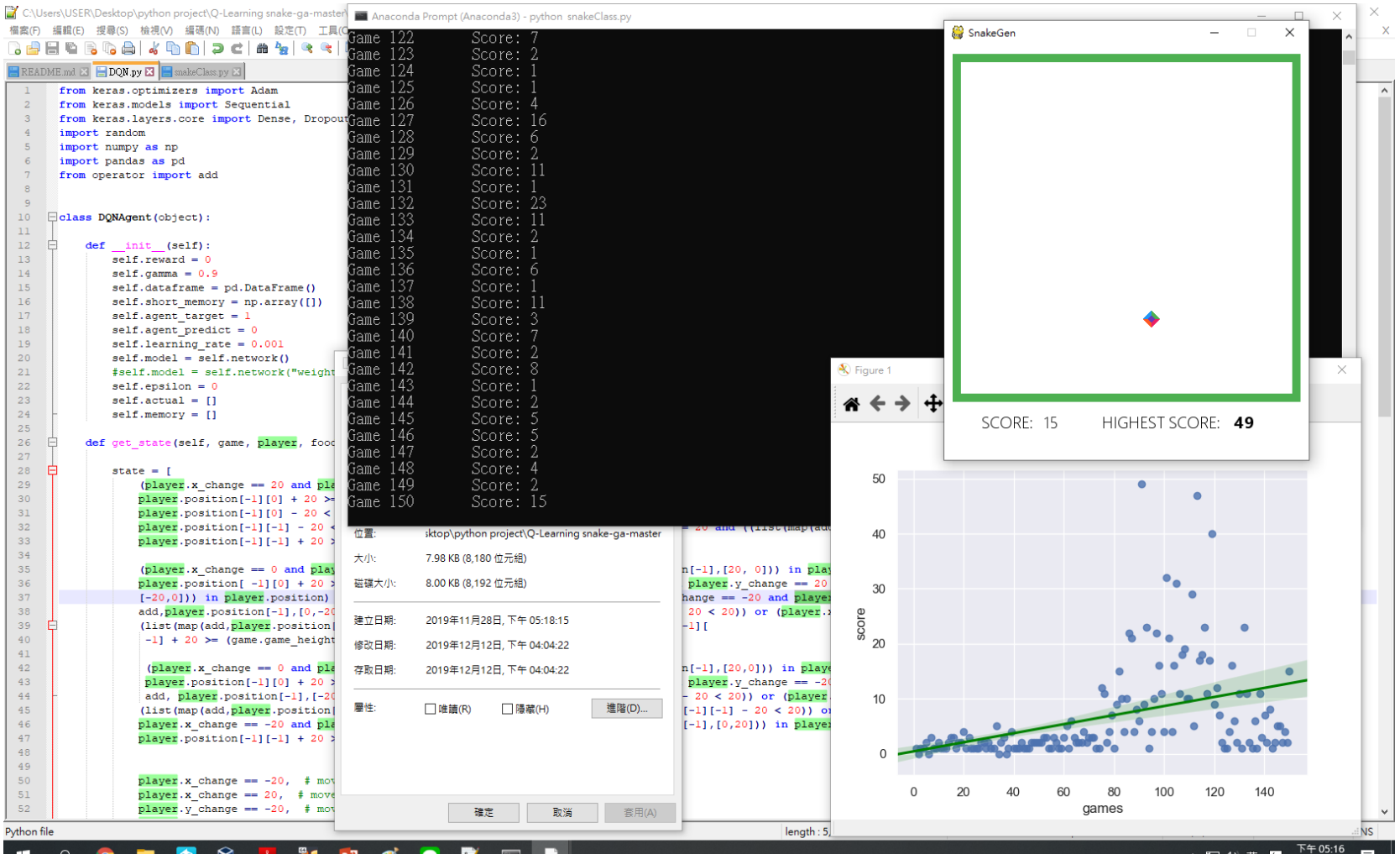
學習率

- 先調整學習率(LR)，調整學習率可以直接影響NN-Model最終output梯度下降演算法的極小值尋找跨度

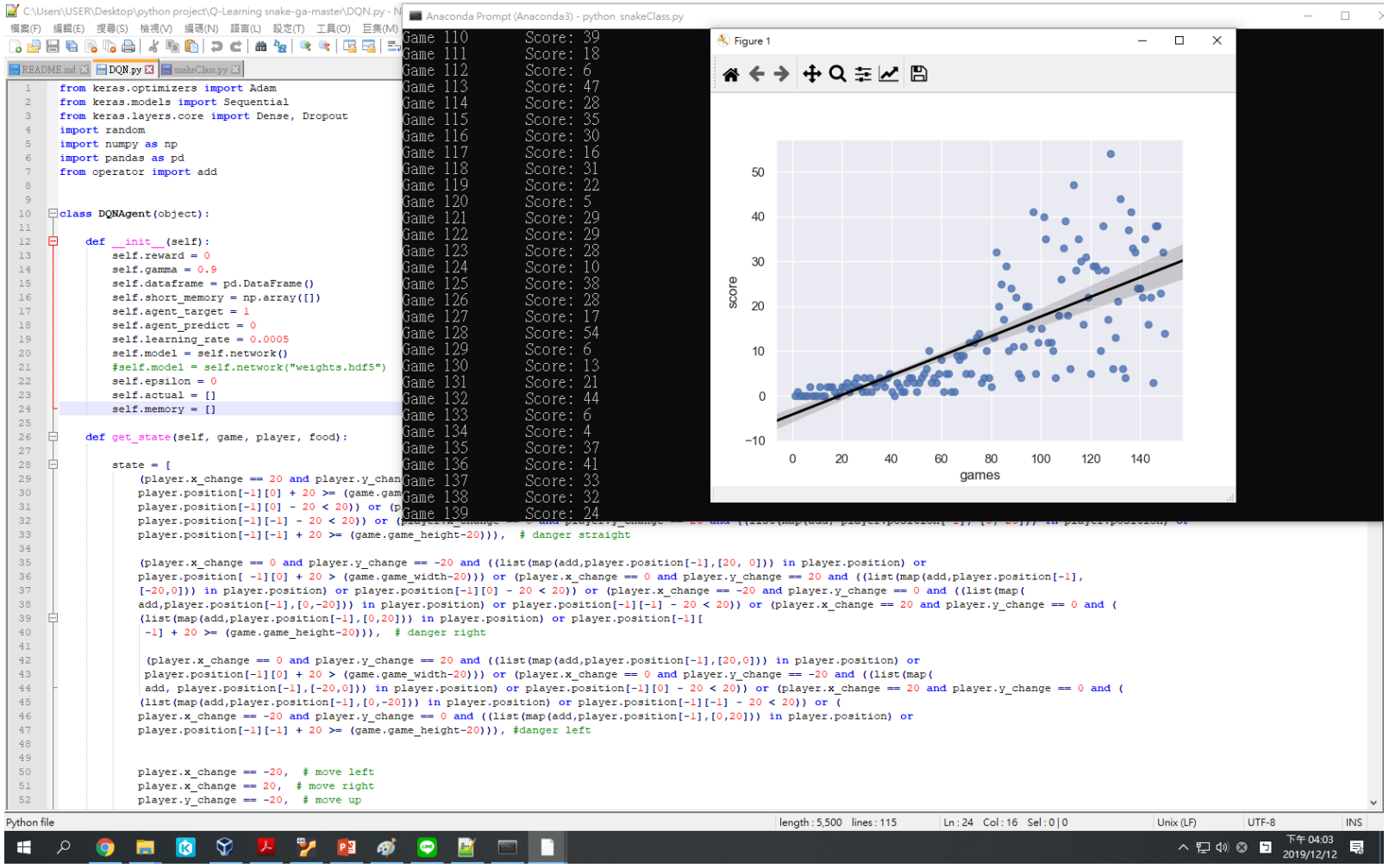
LR:0.01


Anaconda Prompt (Anaconda3) - python snakeClass.py
Game 122 Score: 0
Game 123 Score: 0
Game 124 Score: 0
Game 125 Score: 0
Game 126 Score: 0
Game 127 Score: 0
Game 128 Score: 1
Game 129 Score: 1
Game 130 Score: 0
Game 131 Score: 0
Game 132 Score: 0
Game 133 Score: 0
Game 134 Score: 0
Game 135 Score: 1
Game 136 Score: 1
Game 137 Score: 0
Game 138 Score: 0
Game 139 Score: 0
Game 140 Score: 0
Figure 1
score
games
SnakeGen
SCORE: 0
HIGHEST SCORE: 2
Python file

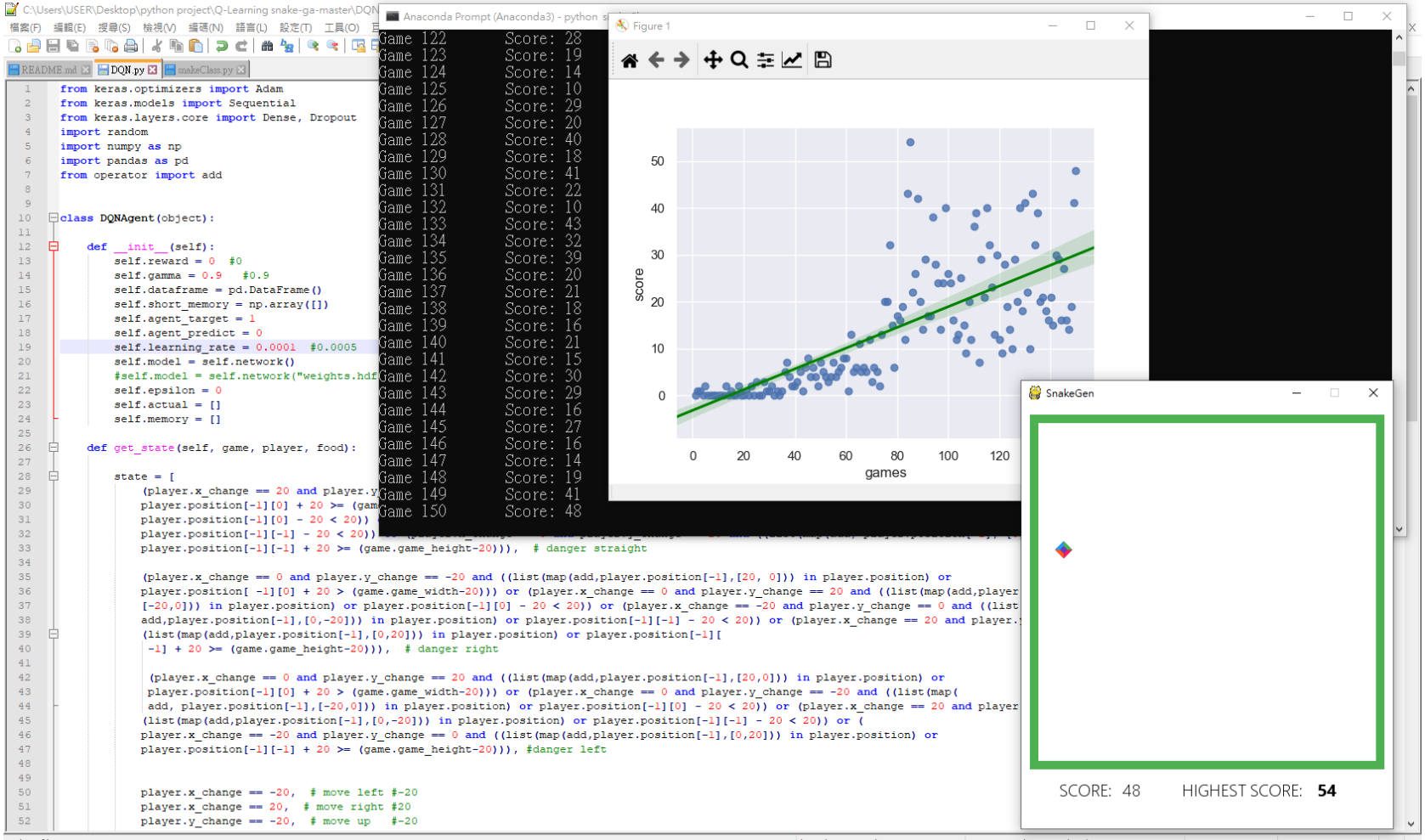
LR:0.001


Anaconda Prompt (Anaconda3) - python snakeClass.py
Game 122 Score: 7
Game 123 Score: 2
Game 124 Score: 1
Game 125 Score: 1
Game 126 Score: 4
Game 127 Score: 16
Game 128 Score: 6
Game 129 Score: 2
Game 130 Score: 11
Game 131 Score: 1
Game 132 Score: 23
Game 133 Score: 11
Game 134 Score: 2
Game 135 Score: 1
Game 136 Score: 6
Game 137 Score: 1
Game 138 Score: 11
Game 139 Score: 3
Game 140 Score: 7
Game 141 Score: 2
Game 142 Score: 8
Game 143 Score: 1
Game 144 Score: 2
Game 145 Score: 5
Game 146 Score: 5
Game 147 Score: 2
Game 148 Score: 4
Game 149 Score: 2
Game 150 Score: 15
SnakeGen
SCORE: 15 HIGHEST SCORE: 49
Figure 1
score
games
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers.core import Dense, Dropout
import random
import numpy as np
import pandas as pd
from operator import add
class DQNAgent(object):
def __init__(self):
self.reward = 0
self.gamma = 0.9
self.dataframe = pd.DataFrame()
self.short_memory = np.array([])
self.agent_target = 1
self.agent_predict = 0
self.learning_rate = 0.001
self.model = self.network()
self.epsilon = 0
self.actual = []
self.memory = []
def get_state(self, game, player, food):
state = [
Python file

LR:0.0005


Anaconda Prompt (Anaconda3) - python snakeClass.py
Game 110 Score: 39
Game 111 Score: 18
Game 112 Score: 6
Game 113 Score: 47
Game 114 Score: 28
Game 115 Score: 35
Game 116 Score: 30
Game 117 Score: 16
Game 118 Score: 31
Game 119 Score: 22
Game 120 Score: 5
Game 121 Score: 29
Game 122 Score: 29
Game 123 Score: 28
Game 124 Score: 10
Game 125 Score: 38
Game 126 Score: 28
Game 127 Score: 17
Game 128 Score: 54
Game 129 Score: 6
Game 130 Score: 13
Game 131 Score: 21
Game 132 Score: 44
Game 133 Score: 6
Game 134 Score: 4
Game 135 Score: 37
Game 136 Score: 41
Game 137 Score: 33
Game 138 Score: 32
Game 139 Score: 24
Figure 1
score
games
README.md
DQN.py
snakeClass.py
class DQNAgent(object):
def __init__(self):
self.reward = 0
self.gamma = 0.9
self.learning_rate = 0.0005
self.model = self.network()
def get_state(self, game, player, food):
state = [
Python file
length : 5,500 lines : 115
Ln : 24 Col : 16 Sel : 0 | 0
Unix (LF)
UTF-8
INS

LR:0.0001


C:\Users\USER\Desktop\python project\Q-Learning snake-ga-master\DQN
README.md
DQN.py
snakeClass.py
from keras.optimizers import Adam
from keras.models import Sequential
from keras.layers.core import Dense, Dropout
import random
import numpy as np
import pandas as pd
from operator import add
class DQNAgent(object):
def __init__(self):
self.reward = 0 #0
self.gamma = 0.9 #0.9
self.dataframe = pd.DataFrame()
self.short_memory = np.array([])
self.agent_target = 1
self.agent_predict = 0
self.learning_rate = 0.0001 #0.0005
self.model = self.network()
#self.model = self.network("weights.hdf
self.epsilon = 0
self.actual = []
self.memory = []
def get_state(self, game, player, food):
state = [
player.x_change == -20,
player.x_change == 20,
player.y_change == -20,
Anaconda Prompt (Anaconda3) - python
Game 122 Score: 28
Game 123 Score: 19
Game 124 Score: 14
Game 125 Score: 10
Game 126 Score: 29
Game 127 Score: 20
Game 128 Score: 40
Game 129 Score: 18
Game 130 Score: 41
Game 131 Score: 22
Game 132 Score: 10
Game 133 Score: 43
Game 134 Score: 32
Game 135 Score: 39
Game 136 Score: 20
Game 137 Score: 21
Game 138 Score: 18
Game 139 Score: 16
Game 140 Score: 21
Game 141 Score: 15
Game 142 Score: 30
Game 143 Score: 29
Game 144 Score: 16
Game 145 Score: 27
Game 146 Score: 16
Game 147 Score: 14
Game 148 Score: 19
Game 149 Score: 41
Game 150 Score: 48
Figure 1
score
games
SnakeGen
SCORE: 48
HIGHEST SCORE: 54
Python file
length : 5,558 lines : 115
Ln : 19 Col : 45 Sel : 0 | 0
Unix (LF)
UTF-8
INS

參數調整(1/6)

- 由先前調整確認了最佳學習率介於0.0005~0.0001間。後續在LR=0.0005條件下調整獎勵與懲罰參數

分數/學習率	0.01	0.001	0.0005	0.0001
最低分數	0	0	0	0
最高分數	2	49	54	52

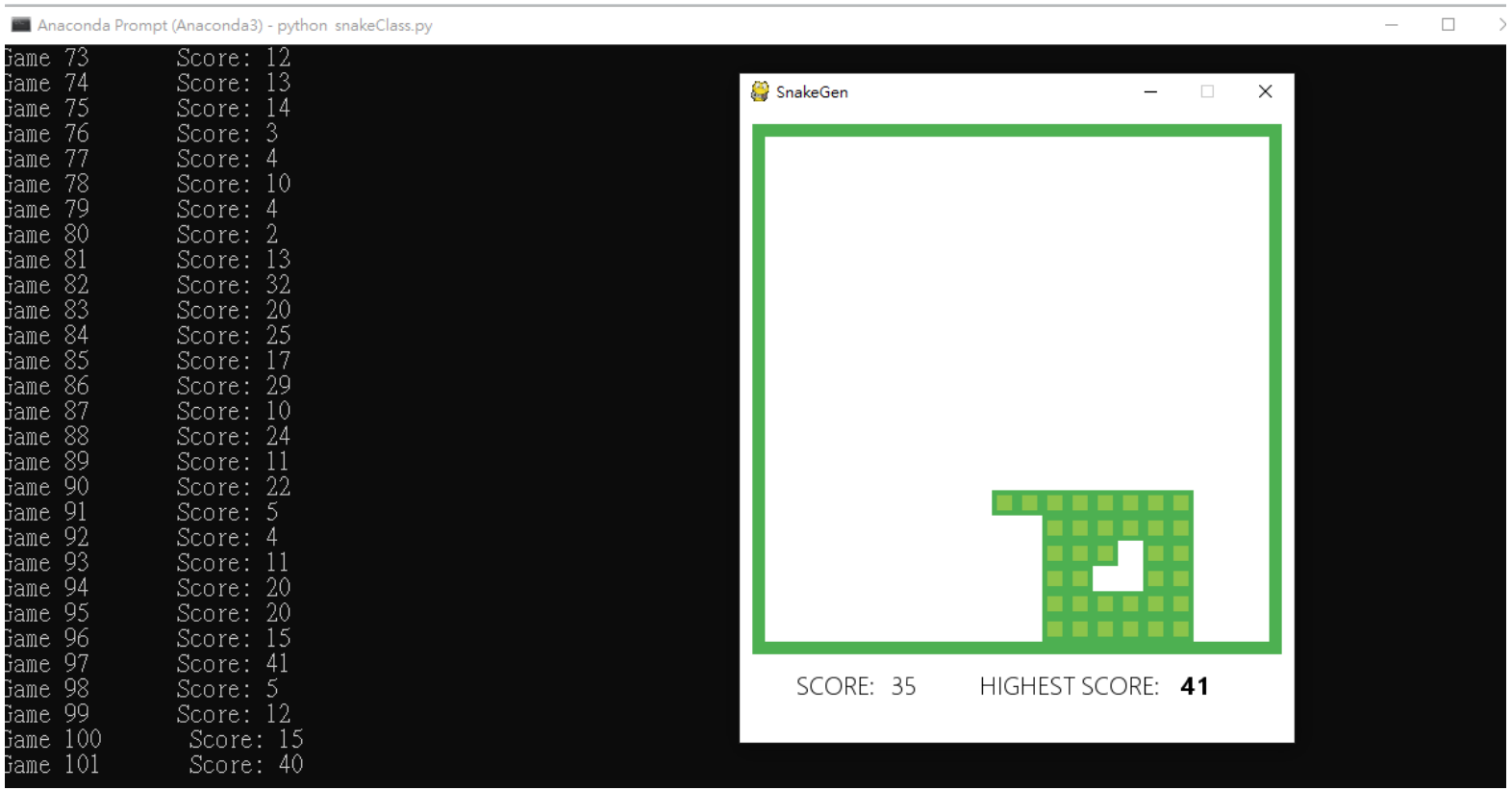
參數調整(2/6)

分數/獎懲權重	+30/0	+20/0	+10/-10	0/-20	0/-30
最低分數	0	0	0	0	0
最高分數	8	10	56 	2	0

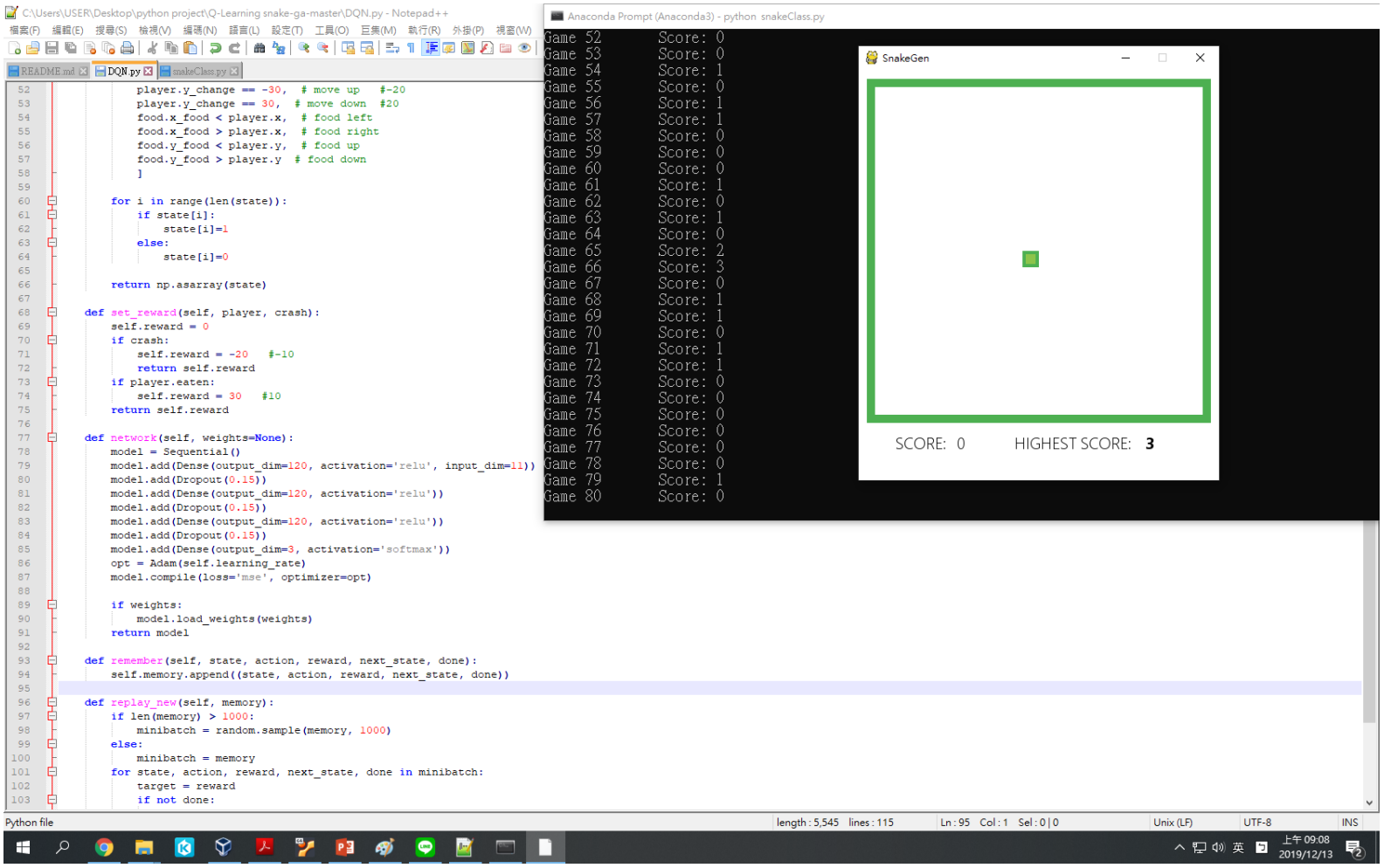
參數調整(3/6)

- 在獎勵/懲罰參數差距過大時貪吃蛇會有幾種有趣的現象，不僅無法達到最高分數的目標，還會造成機器進行迴圈無法破解的情況。
- 繞著最大範圍跑圈而無視獎勵
- 過度重視獎勵而碰觸自身結束遊戲
- 未來可以利用實驗設計找出最大影響因子

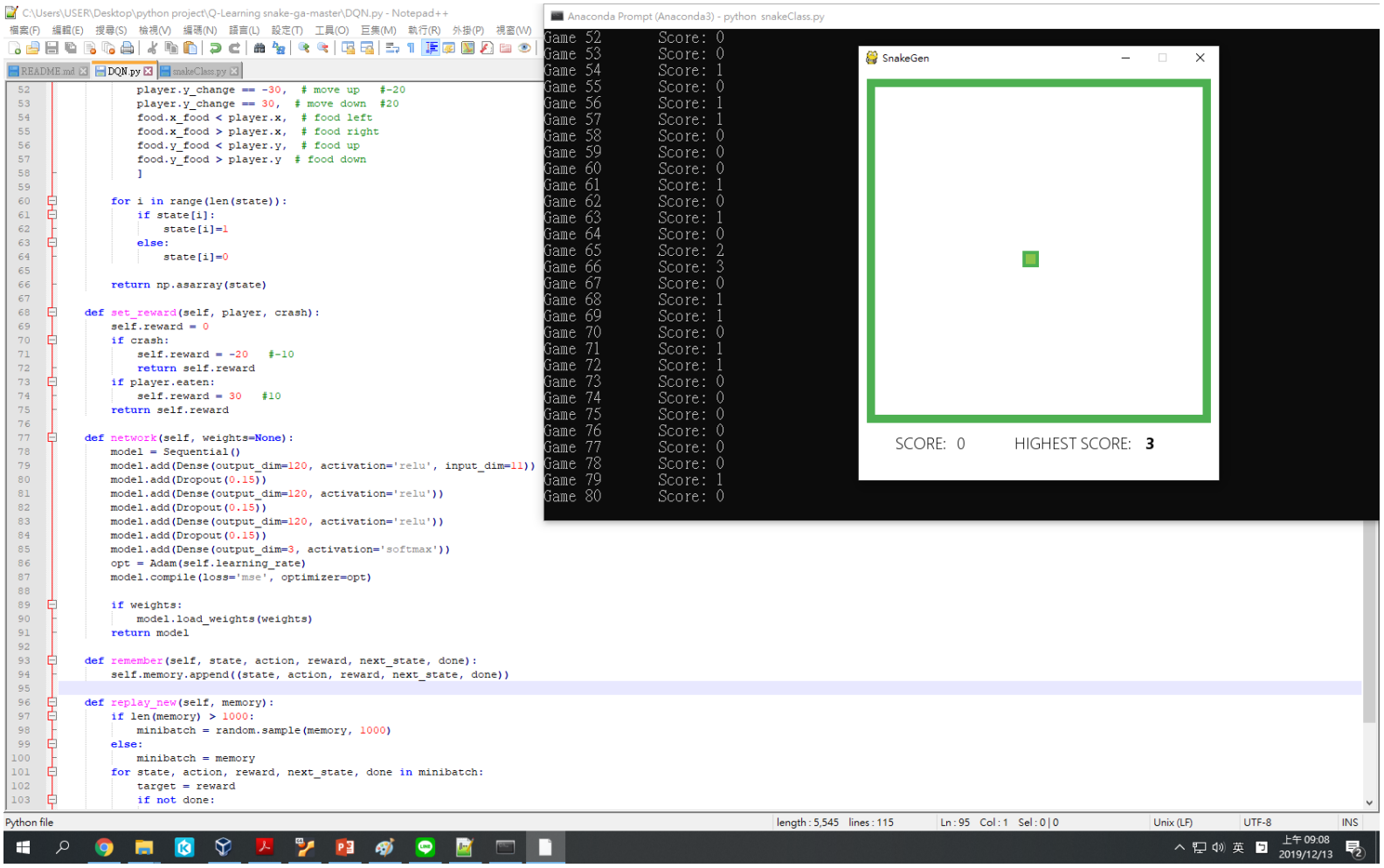
參數調整(4/6)


Anaconda Prompt (Anaconda3) - python snakeClass.py
ame 73 Score: 12
ame 74 Score: 13
ame 75 Score: 14
ame 76 Score: 3
ame 77 Score: 4
ame 78 Score: 10
ame 79 Score: 4
ame 80 Score: 2
ame 81 Score: 13
ame 82 Score: 32
ame 83 Score: 20
ame 84 Score: 25
ame 85 Score: 17
ame 86 Score: 29
ame 87 Score: 10
ame 88 Score: 24
ame 89 Score: 11
ame 90 Score: 22
ame 91 Score: 5
ame 92 Score: 4
ame 93 Score: 11
ame 94 Score: 20
ame 95 Score: 20
ame 96 Score: 15
ame 97 Score: 41
ame 98 Score: 5
ame 99 Score: 12
ame 100 Score: 15
ame 101 Score: 40
SnakeGen
SCORE: 35
HIGHEST SCORE: 41

參數調整(5/6)


C:\Users\USER\Desktop\python project\Q-Learning snake-ga-master\DQN.py - Notepad++
README.md
DQN.py
snakeClass.py
player.y_change == -30, # move up #-20
player.y_change == 30, # move down #20
food.x_food < player.x, # food left
food.x_food > player.x, # food right
food.y_food < player.y, # food up
food.y_food > player.y # food down
for i in range(len(state)):
if state[i]:
state[i]=1
else:
state[i]=0
return np.asarray(state)
def set_reward(self, player, crash):
self.reward = 0
if crash:
self.reward = -20 #-10
return self.reward
if player.eaten:
self.reward = 30 #10
return self.reward
def network(self, weights=None):
model = Sequential()
model.add(Dense(output_dim=120, activation='relu', input_dim=11))
model.add(Dropout(0.15))
model.add(Dense(output_dim=120, activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(output_dim=120, activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(output_dim=3, activation='softmax'))
opt = Adam(self.learning_rate)
model.compile(loss='mse', optimizer=opt)
if weights:
model.load_weights(weights)
return model
def remember(self, state, action, reward, next_state, done):
self.memory.append((state, action, reward, next_state, done))
def replay_new(self, memory):
if len(memory) > 1000:
minibatch = random.sample(memory, 1000)
else:
minibatch = memory
for state, action, reward, next_state, done in minibatch:
target = reward
if not done:
Anaconda Prompt (Anaconda3) - python snakeClass.py
Game 52 Score: 0
Game 53 Score: 0
Game 54 Score: 1
Game 55 Score: 0
Game 56 Score: 1
Game 57 Score: 1
Game 58 Score: 0
Game 59 Score: 0
Game 60 Score: 0
Game 61 Score: 1
Game 62 Score: 0
Game 63 Score: 1
Game 64 Score: 0
Game 65 Score: 2
Game 66 Score: 3
Game 67 Score: 0
Game 68 Score: 1
Game 69 Score: 1
Game 70 Score: 0
Game 71 Score: 1
Game 72 Score: 1
Game 73 Score: 0
Game 74 Score: 0
Game 75 Score: 0
Game 76 Score: 0
Game 77 Score: 0
Game 78 Score: 0
Game 79 Score: 1
Game 80 Score: 0
SnakeGen
SCORE: 0
HIGHEST SCORE: 3
Python file
length : 5,545 lines : 115
Ln : 95 Col : 1 Sel : 0 | 0
Unix (LF)
UTF-8
INS

參數調整(6/6)


C:\Users\USER\Desktop\python project\Q-Learning snake-ga-master\DQN.py - Notepad++
README.md
DQN.py
snakeClass.py
player.y_change == -30, # move up #-20
player.y_change == 30, # move down #20
food.x_food < player.x, # food left
food.x_food > player.x, # food right
food.y_food < player.y, # food up
food.y_food > player.y # food down
for i in range(len(state)):
if state[i]:
state[i]=1
else:
state[i]=0
return np.asarray(state)
def set_reward(self, player, crash):
self.reward = 0
if crash:
self.reward = -20 #-10
return self.reward
if player.eaten:
self.reward = 30 #10
return self.reward
def network(self, weights=None):
model = Sequential()
model.add(Dense(output_dim=120, activation='relu', input_dim=11))
model.add(Dropout(0.15))
model.add(Dense(output_dim=120, activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(output_dim=120, activation='relu'))
model.add(Dropout(0.15))
model.add(Dense(output_dim=3, activation='softmax'))
opt = Adam(self.learning_rate)
model.compile(loss='mse', optimizer=opt)
if weights:
model.load_weights(weights)
return model
def remember(self, state, action, reward, next_state, done):
self.memory.append((state, action, reward, next_state, done))
def replay_new(self, memory):
if len(memory) > 1000:
minibatch = random.sample(memory, 1000)
else:
minibatch = memory
for state, action, reward, next_state, done in minibatch:
target = reward
if not done:
Anaconda Prompt (Anaconda3) - python snakeClass.py
Game 52 Score: 0
Game 53 Score: 0
Game 54 Score: 1
Game 55 Score: 0
Game 56 Score: 1
Game 57 Score: 1
Game 58 Score: 0
Game 59 Score: 0
Game 60 Score: 0
Game 61 Score: 1
Game 62 Score: 0
Game 63 Score: 1
Game 64 Score: 0
Game 65 Score: 2
Game 66 Score: 3
Game 67 Score: 0
Game 68 Score: 1
Game 69 Score: 1
Game 70 Score: 0
Game 71 Score: 1
Game 72 Score: 1
Game 73 Score: 0
Game 74 Score: 0
Game 75 Score: 0
Game 76 Score: 0
Game 77 Score: 0
Game 78 Score: 0
Game 79 Score: 1
Game 80 Score: 0
SnakeGen
SCORE: 0
HIGHEST SCORE: 3
Python file
length : 5,545 lines : 115
Ln : 95 Col : 1 Sel : 0 | 0
Unix (LF)
UTF-8
INS

國立清華大學 National Tsing Hua University

INNOVATION ENGINEERING LAB

討論

- Q-Learning對於環境與參數的依賴性非常高，任何參數的調整都會對於最終結果造成巨大的影響。

討論

- 最高成績56分不僅受限於學習率、獎懲參數的影響，環境對於貪吃蛇的限制也非常大；若是環境建置的更大，貪吃蛇的活動範圍擴大可以讓最終成績提高

國立清華大學 National Tsing Hua University

INNOVATION ENGINEERING LAB

未來可改進處

- 測試原環境下的最佳參數與最高成績
- 調整環境參數(以擴大環境為目標)
- 動態調整環境與蛇身長度 (找出最佳比例)
- 利用實驗設計找出最佳參數組合

遇到的困難

- Q-Learning的了解程度不足
- 利用網路上的開源資料，藉由model成品的code來對照可以更快了解各參數的作用

Reference

- Q-Learning:
- https://pyliaorachel.github.io/blog/tech/python/2018/06/14/deep-q-learning.html

- https://morvanzhou.github.io/tutorials/machine-learning/torch/4-05-DQN/

國立清華大學
National Tsing Hua University

INNOVATION ENGINEERING LAB

Thank You for Your Listening