

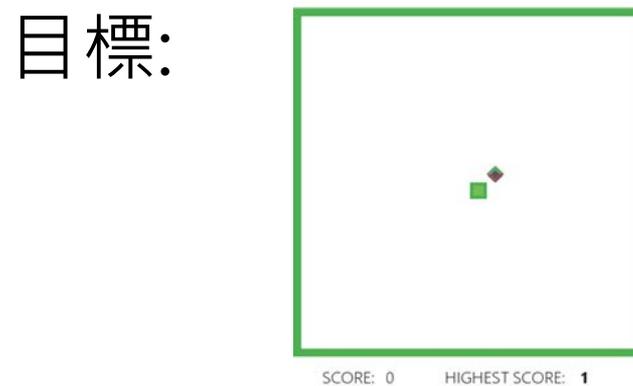
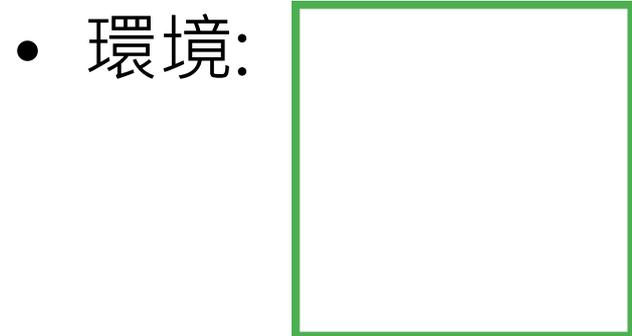
IIE Final Project

Q-Learning of Snake-Game

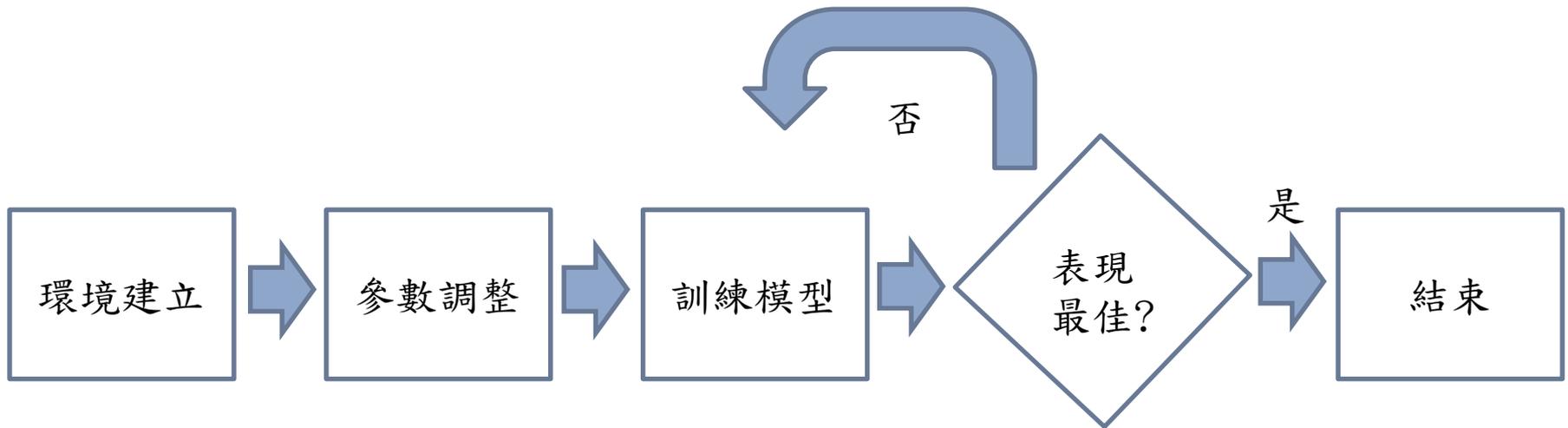
Presenter : 溫致淵

目標

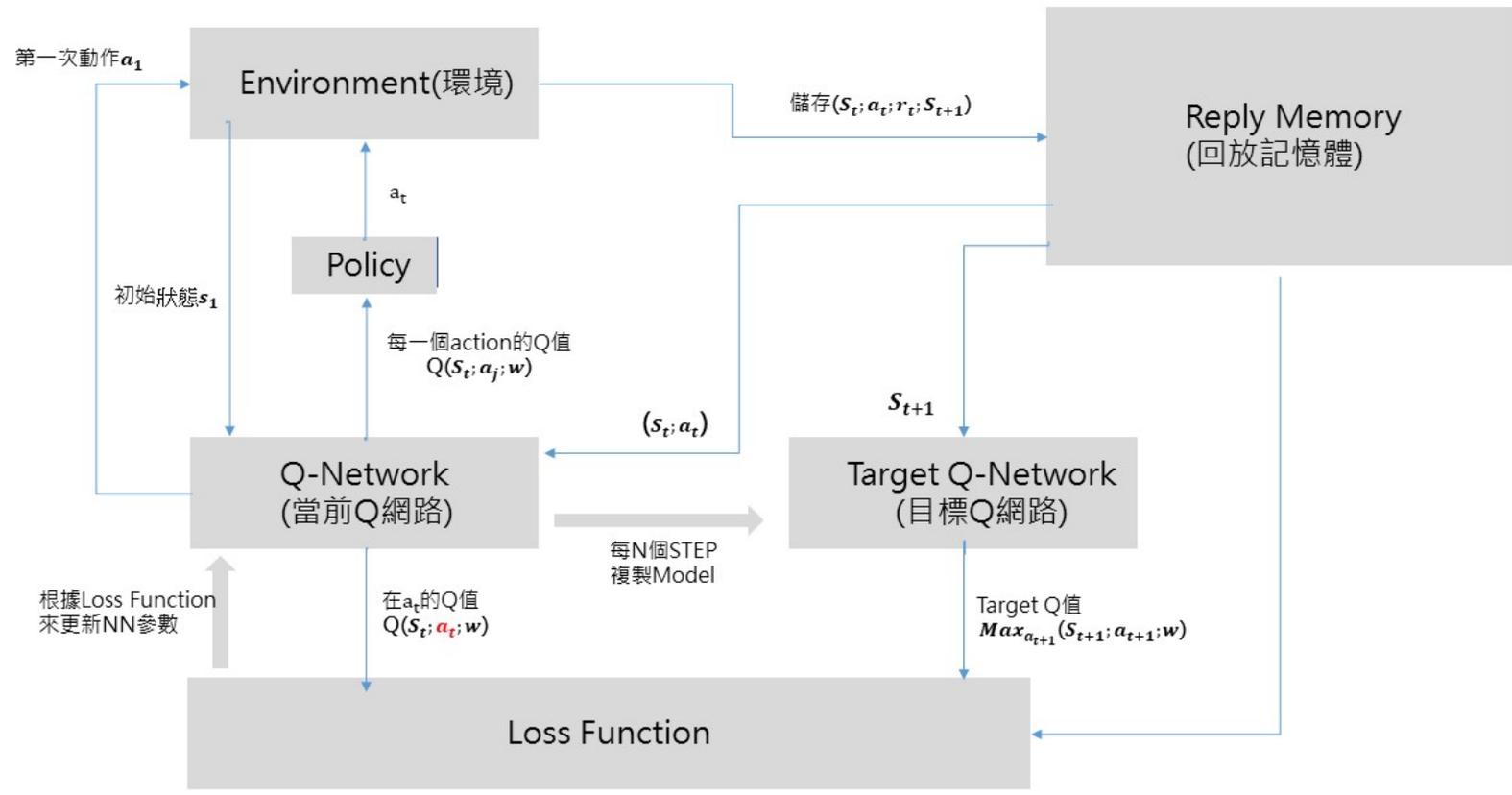
- 透過建立Q-Learning環境模型，利用非監督式增強學習技術建立貪吃蛇遊戲的環境數值，目標使得貪吃蛇分數最大化。



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代大小，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.agent_reward = 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

- 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
```

前進

```
(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
```

左轉

```
(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
```

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: $Q(\text{狀態}, \text{操作}) = R(\text{狀態}, \text{操作}) + \text{Gamma} * \text{Max}[Q(\text{下一個狀態}, \text{具有最高Q值的操作})]$

```
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:0.001

```

1 from keras.optimizers import Adam
2 from keras.models import Sequential
3 from keras.layers.core import Dense, Dropout
4 import random
5 import numpy as np
6 import pandas as pd
7 from operator import add
8
9
10 class DQNAgent(object):
11
12     def __init__(self):
13         self.reward = 0
14         self.gamma = 0.9
15         self.dataframe = pd.DataFrame()
16         self.short_memory = np.array([])
17         self.agent_target = 1
18         self.agent_predict = 0
19         self.learning_rate = 0.001
20         self.model = self.network()
21         #self.model = self.network("weight")
22         self.epsilon = 0
23         self.actual = []
24         self.memory = []
25
26     def get_state(self, game, player, food):
27
28         state = [
29             (player.x_change == 20 and player.position[-1][0] + 20 >= game.game_width) or
30             (player.x_change == -20 and (player.position[-1][0] - 20 <= 0)) or
31             (player.y_change == 20 and (player.position[-1][1] - 20 <= 0)) or
32             (player.y_change == -20 and (player.position[-1][1] + 20 >= game.game_height)) or
33             (player.x_change == 0 and player.position[-1][0] + 20 >= game.game_width) or
34             (-20,0)) in player.position) or
35             (player.x_change == -20 and (player.position[-1][0] - 20 <= 0)) or
36             (player.y_change == 20 and (player.position[-1][1] - 20 <= 0)) or
37             (player.y_change == -20 and (player.position[-1][1] + 20 >= game.game_height)) or
38             (player.x_change == 0 and player.position[-1][0] + 20 >= game.game_width) or
39             (-20,0)) in player.position) or
40             (player.x_change == -20 and (player.position[-1][0] - 20 <= 0)) or
41             (player.y_change == 20 and (player.position[-1][1] - 20 <= 0)) or
42             (player.y_change == -20 and (player.position[-1][1] + 20 >= game.game_height)) or
43             (player.x_change == 0 and player.position[-1][0] + 20 >= game.game_width) or
44             (-20,0)) in player.position) or
45             (player.x_change == -20 and (player.position[-1][0] - 20 <= 0)) or
46             (player.y_change == 20 and (player.position[-1][1] - 20 <= 0)) or
47             (player.y_change == -20 and (player.position[-1][1] + 20 >= game.game_height)) or
48             (player.x_change == 0 and player.position[-1][0] + 20 >= game.game_width) or
49             (-20,0)) in player.position) or
50             (player.x_change == -20 and (player.position[-1][0] - 20 <= 0)) or
51             (player.y_change == 20 and (player.position[-1][1] - 20 <= 0)) or
52             (player.y_change == -20 and (player.position[-1][1] + 20 >= game.game_height)) or

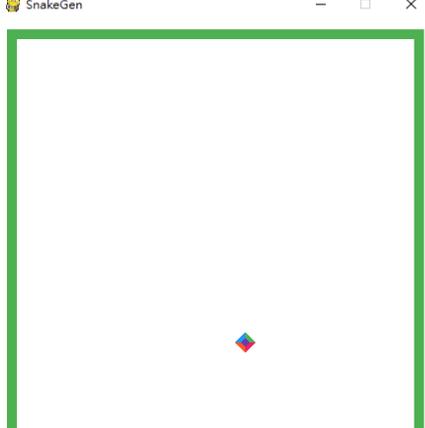
```

```

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

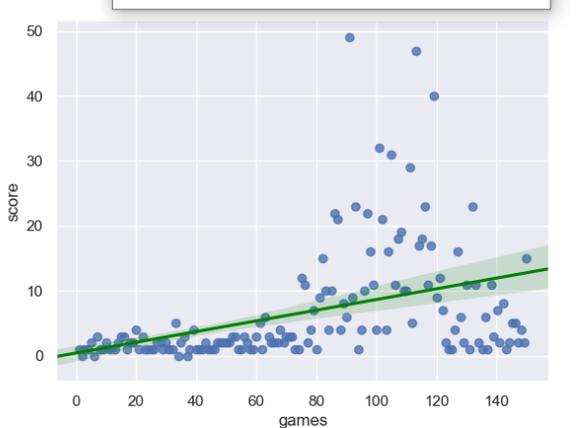
```

SnakeGen



SCORE: 15 HIGHEST SCORE: 49

Figure 1



位置: sktop/python project/Q-Learning snake-ga-master
 大小: 7.98 KB (8,180 位元組)
 磁碟大小: 8.00 KB (8,192 位元組)
 建立日期: 2019年11月28日, 下午 05:18:15
 修改日期: 2019年12月12日, 下午 04:04:22
 存取日期: 2019年12月12日, 下午 04:04:22
 屬性: 唯讀(R) 隱藏(H) 進階(D)...

LR:0.0005

```

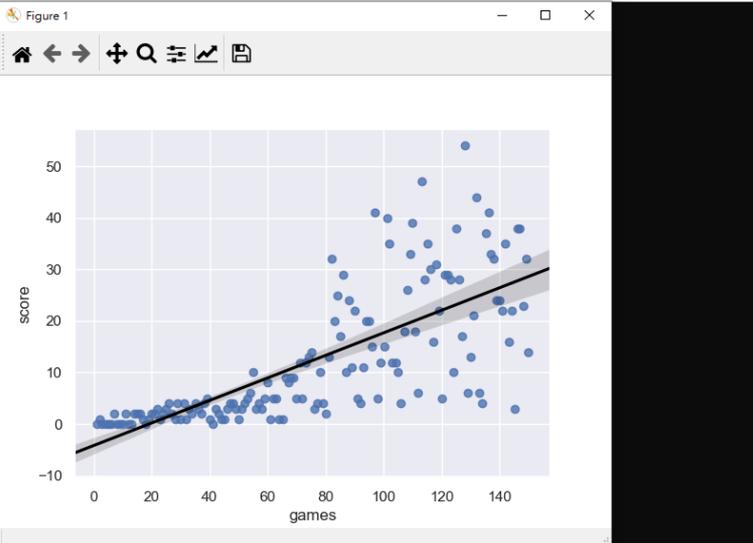
1 from keras.optimizers import Adam
2 from keras.models import Sequential
3 from keras.layers.core import Dense, Dropout
4 import random
5 import numpy as np
6 import pandas as pd
7 from operator import add
8
9
10 class DQNAgent(object):
11
12     def __init__(self):
13         self.reward = 0
14         self.gamma = 0.9
15         self.dataframe = pd.DataFrame()
16         self.short_memory = np.array([])
17         self.agent_target = 1
18         self.agent_predict = 0
19         self.learning_rate = 0.0005
20         self.model = self.network()
21         #self.model = self.network("weights.hdf5")
22         self.epsilon = 0
23         self.actual = []
24         self.memory = []
25
26     def get_state(self, game, player, food):
27
28         state = [
29             (player.x_change == 20 and player.y_chan
30             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],
31             player.position[-1][0] - 20 < 20)) or (player.x_change == -20 and player.y_change == 0 and ((list(map(
32             player.position[-1][-1] - 20 < 20)) or (player.x_change == 20 and player.y_change == 0 and (
33             player.position[-1][-1] + 20 >= (game.game_height-20))), # danger straight
34
35             (player.x_change == 0 and player.y_change == -20 and ((list(map(add,player.position[-1],[20, 0])) in player.position) or
36             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],
37             [-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(
38             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 (
39             (list(map(add,player.position[-1],[0,20])) in player.position) or player.position[-1][-1] - 20 < 20)) or (
40             [-1] + 20 >= (game.game_height-20))), # danger right
41
42             (player.x_change == 0 and player.y_change == 20 and ((list(map(add,player.position[-1],[20,0])) in player.position) or
43             player.position[-1][0] + 20 > (game.game_width-20)) or (player.x_change == 0 and player.y_change == -20 and ((list(map(
44             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 (
45             (list(map(add,player.position[-1],[0,-20])) in player.position) or player.position[-1][-1] - 20 < 20)) or (
46             player.x_change == -20 and player.y_change == 0 and ((list(map(add,player.position[-1],[0,20])) in player.position) or
47             player.position[-1][-1] + 20 >= (game.game_height-20))), #danger left
48
49
50             player.x_change == -20, # move left
51             player.x_change == 20, # move right
52             player.y_change == -20, # move up

```

```

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

```



Python file | length: 5,500 lines: 115 | Ln: 24 Col: 16 Sel: 0|0 | Unix (LF) | UTF-8 | INS

下午 04:03
2019/12/12

LR:0.0001

```

1 from keras.optimizers import Adam
2 from keras.models import Sequential
3 from keras.layers.core import Dense, Dropout
4 import random
5 import numpy as np
6 import pandas as pd
7 from operator import add
8
9
10 class DQNAgent(object):
11
12     def __init__(self):
13         self.reward = 0 #0
14         self.gamma = 0.9 #0.9
15         self.dataframe = pd.DataFrame()
16         self.short_memory = np.array([])
17         self.agent_target = 1
18         self.agent_predict = 0
19         self.learning_rate = 0.0001 #0.0005
20         self.model = self.network()
21         #self.model = self.network("weights.hdf5")
22         self.epsilon = 0
23         self.actual = []
24         self.memory = []
25
26     def get_state(self, game, player, food):
27
28         state = [
29             (player.x_change == 20 and player.y_change == 20 > (game.game_width-20)) or (player.position[-1][0] + 20 >= (game.game_width-20)) or (player.position[-1][0] - 20 < 20))
30             (player.position[-1][0] - 20 < 20))
31             (player.position[-1][1] - 20 < 20))
32             (player.position[-1][1] + 20 >= (game.game_height-20))), # danger straight
33
34             (player.x_change == 0 and player.y_change == -20 and ((list(map(add,player.position[-1],[20,0])) in player.position) or
35             player.position[-1][0] + 20 > (game.game_width-20)) or (player.x_change == 0 and player.y_change == 20 and ((list(map(add,player
36             [-20,0])) in player.position) or player.position[-1][0] - 20 < 20)) or (player.x_change == -20 and player.y_change == 0 and ((list
37             add,player.position[-1],[0,-20])) in player.position) or player.position[-1][1] - 20 < 20)) or (player.x_change == 20 and player
38             (list(map(add,player.position[-1],[0,20])) in player.position) or player.position[-1][1] + 20 >= (game.game_height-20))), # danger right
39
40             (player.x_change == 0 and player.y_change == 20 and ((list(map(add,player.position[-1],[20,0])) in player.position) or
41             player.position[-1][0] + 20 > (game.game_width-20)) or (player.x_change == 0 and player.y_change == -20 and ((list(map(
42             add, player.position[-1],[0,-20])) in player.position) or player.position[-1][0] - 20 < 20)) or (player.x_change == 20 and player
43             (list(map(add,player.position[-1],[0,-20])) in player.position) or player.position[-1][1] - 20 < 20)) or (
44             player.x_change == -20 and player.y_change == 0 and ((list(map(add,player.position[-1],[0,20])) in player.position) or
45             player.position[-1][1] + 20 >= (game.game_height-20))), #danger left
46
47
48
49
50         player.x_change == -20, # move left #-20
51         player.x_change == 20, # move right #20
52         player.y_change == -20, # move up #-20

```

Game	Score
122	28
123	19
124	14
125	10
126	29
127	20
128	40
129	18
130	41
131	22
132	10
133	43
134	32
135	39
136	20
137	21
138	18
139	16
140	21
141	15
142	30
143	29
144	16
145	27
146	16
147	14
148	19
149	41
150	48

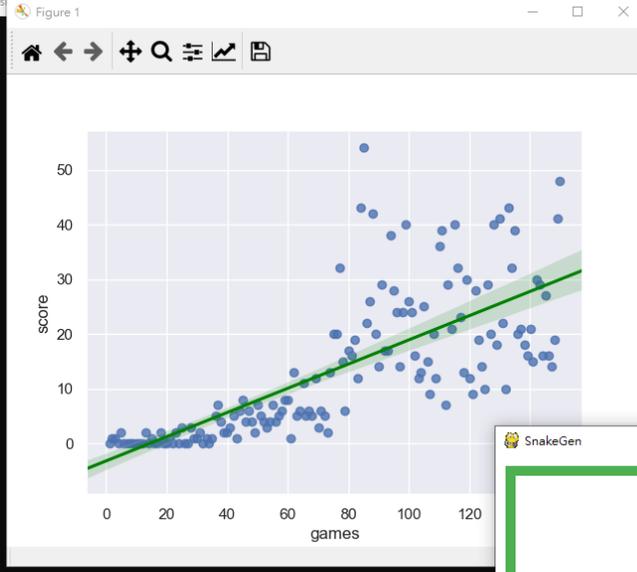
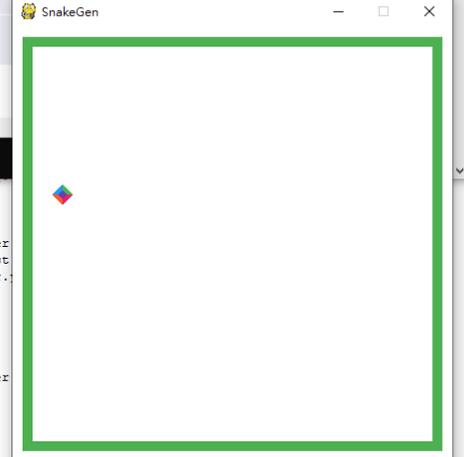


Figure 1



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

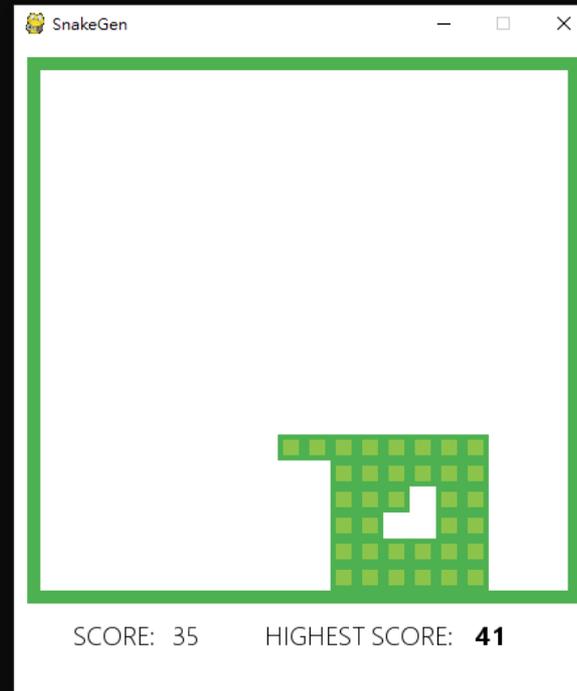
參數調整(3/6)

- 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

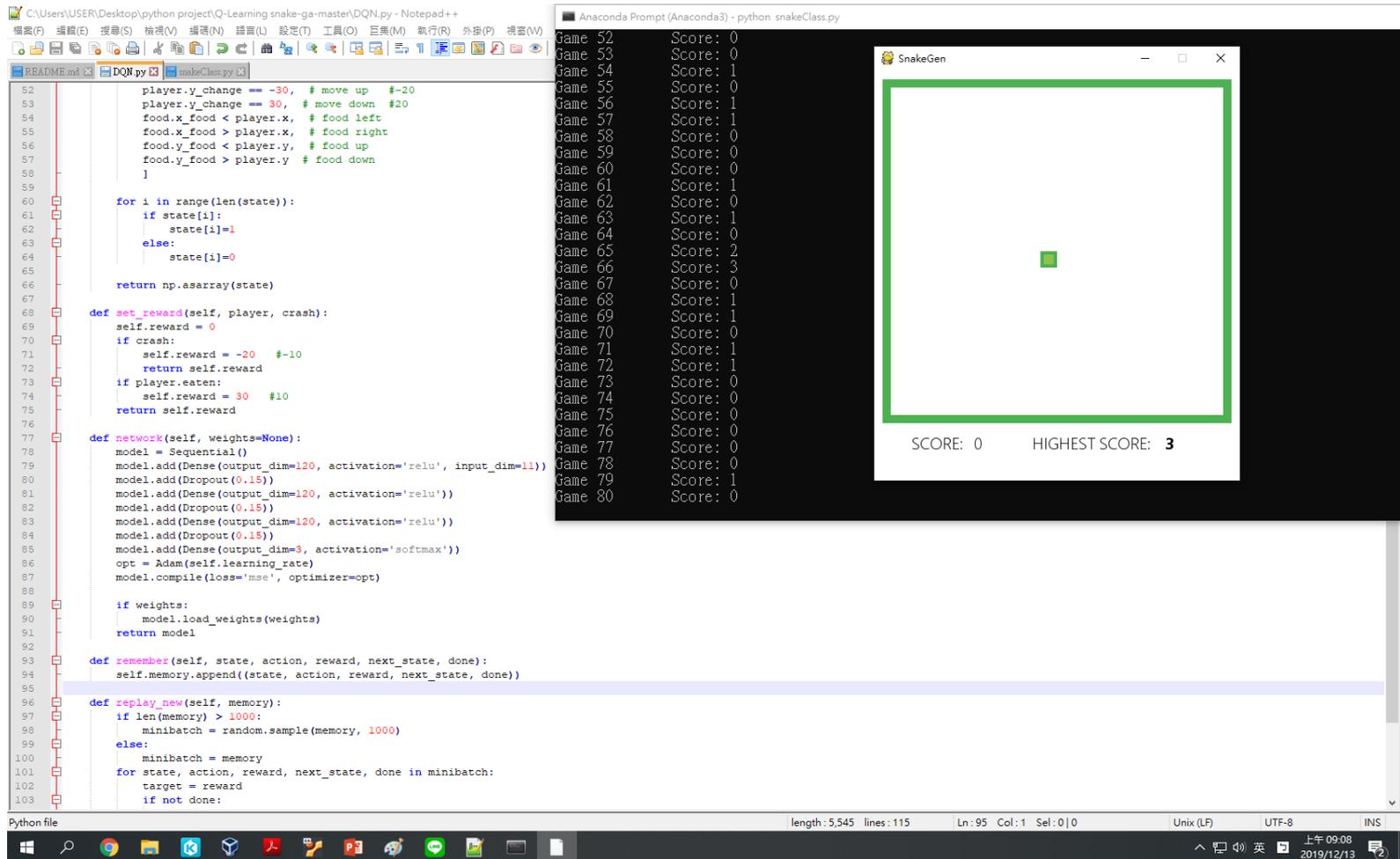
參數調整(4/6)

Anaconda Prompt (Anaconda3) - python snakeClass.py

```
Game 73      Score: 12  
Game 74      Score: 13  
Game 75      Score: 14  
Game 76      Score: 3  
Game 77      Score: 4  
Game 78      Score: 10  
Game 79      Score: 4  
Game 80      Score: 2  
Game 81      Score: 13  
Game 82      Score: 32  
Game 83      Score: 20  
Game 84      Score: 25  
Game 85      Score: 17  
Game 86      Score: 29  
Game 87      Score: 10  
Game 88      Score: 24  
Game 89      Score: 11  
Game 90      Score: 22  
Game 91      Score: 5  
Game 92      Score: 4  
Game 93      Score: 11  
Game 94      Score: 20  
Game 95      Score: 20  
Game 96      Score: 15  
Game 97      Score: 41  
Game 98      Score: 5  
Game 99      Score: 12  
Game 100     Score: 15  
Game 101     Score: 40
```



參數調整(5/6)



The screenshot displays a Python development environment with two main windows:

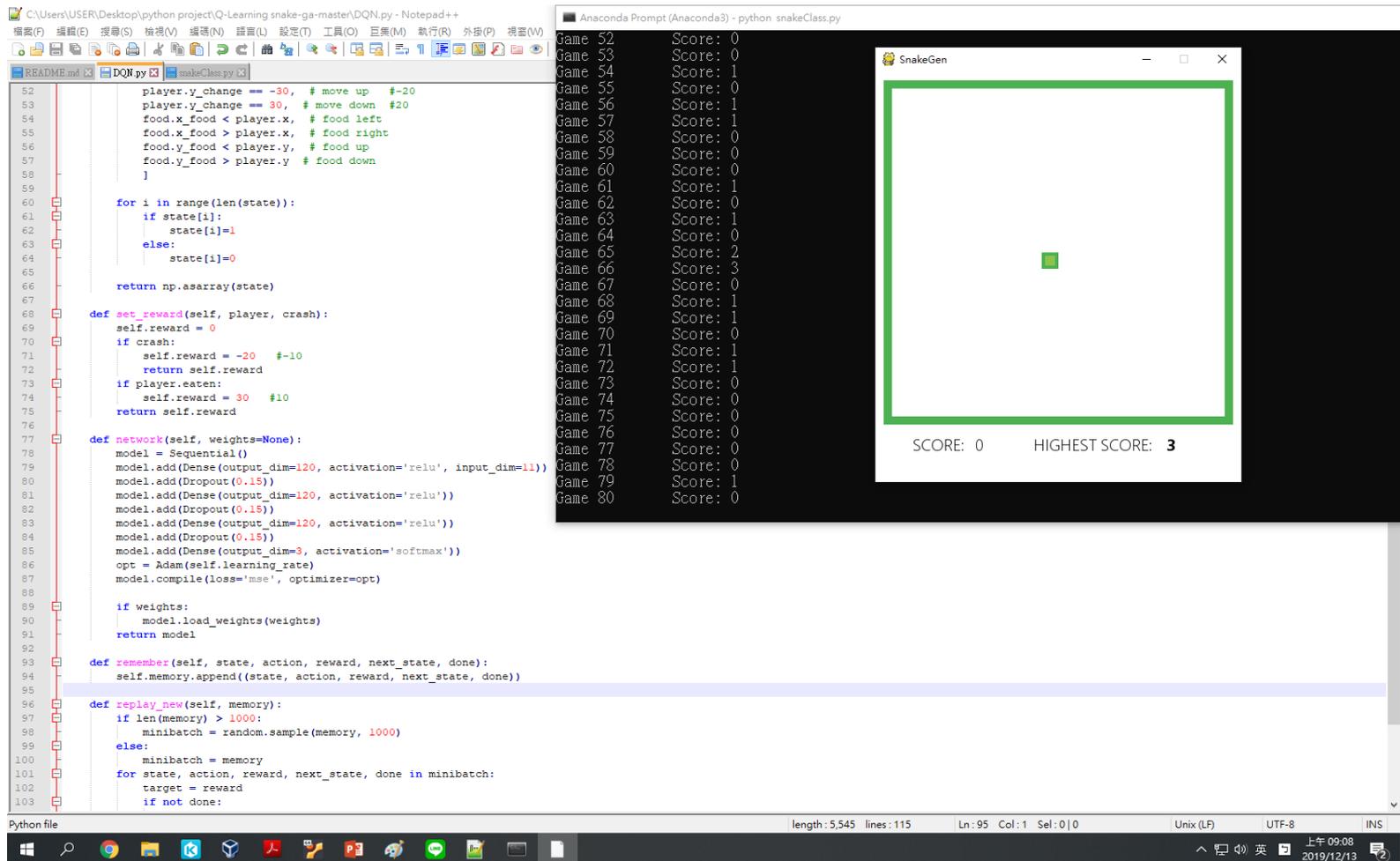
- Code Editor (Left):** Shows the implementation of a Snake game class. Key methods include:
 - `set_reward`: Updates the reward based on game events (eaten, crash, or no action).
 - `network`: Defines a neural network architecture with three hidden layers and a softmax output layer.
 - `replay_new`: Implements a replay buffer mechanism for training.
- Terminal (Right):** Shows the execution output of the game, listing 80 games and their corresponding scores. The highest score achieved is 3.
- SnakeGen Window (Right):** A graphical window showing a snake on a grid. The current score is 0, and the highest score is 3.

The terminal output is as follows:

```

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
  
```

參數調整(6/6)



The screenshot displays a Python IDE with two windows. The left window shows the source code for a Snake game, and the right window shows the execution output and a game window.

Code Snippets:

```

52     player.y_change -= 30, # move up # -20
53     player.y_change = 30, # move down #20
54     food.x_food < player.x, # food left
55     food.x_food > player.x, # food right
56     food.y_food < player.y, # food up
57     food.y_food > player.y # food down
58 ]
59
60 for i in range(len(state)):
61     if state[i]:
62         state[i]=1
63     else:
64         state[i]=0
65
66     return np.asarray(state)
67
68 def set_reward(self, player, crash):
69     self.reward = 0
70     if crash:
71         self.reward = -20 # -10
72         return self.reward
73     if player.eaten:
74         self.reward = 30 #10
75         return self.reward
76
77 def network(self, weights=None):
78     model = Sequential()
79     model.add(Dense(output_dim=120, activation='relu', input_dim=11))
80     model.add(Dropout(0.15))
81     model.add(Dense(output_dim=120, activation='relu'))
82     model.add(Dropout(0.15))
83     model.add(Dense(output_dim=120, activation='relu'))
84     model.add(Dropout(0.15))
85     model.add(Dense(output_dim=3, activation='softmax'))
86     opt = Adam(self.learning_rate)
87     model.compile(loss='mse', optimizer=opt)
88
89     if weights:
90         model.load_weights(weights)
91     return model
92
93 def remember(self, state, action, reward, next_state, done):
94     self.memory.append((state, action, reward, next_state, done))
95
96 def replay_new(self, memory):
97     if len(memory) > 1000:
98         minibatch = random.sample(memory, 1000)
99     else:
100         minibatch = memory
101     for state, action, reward, next_state, done in minibatch:
102         target = reward
103         if not done:

```

Execution Output:

```

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 Window:

SCORE: 0 HIGHEST SCORE: 3

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

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的最佳比例
- 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/>

Thank You for Your Listening