

# Artificial-Intelligence

## homework3

0816023 張紳濡

## 1 Introduction

使用 minimax, Expectimax, Qlearning, DQN 進行 pacman Agent 訓練

## 2 Minimax

```
def Minimax(agent, depth, gameState):
    if gameState.isLose() or gameState.isWin() or depth == self.depth:
        return self.evaluationFunction(gameState)
    if agent == 0:
        tmp = float(-1e9)
        for State in gameState.getLegalActions(agent):
            tmp = max(Minimax(1, depth, gameState.getNextState(agent, State)), tmp)
        return tmp
    else:
        nextagent = agent + 1
        if gameState.getNumAgents() == nextagent:
            nextagent = 0
            depth += 1
        tmp = float(1e9)
        for State in gameState.getLegalActions(agent):
            tmp = min(Minimax(nextagent, depth, gameState.getNextState(agent, State)), tmp)
        return tmp
val = float(-1e9)
for State in gameState.getLegalActions(0):
    tmp = Minimax(1, 0, gameState.getNextState(0, State))
    if tmp > val:
        val = tmp
        next = State
return next
```

先判斷是否為中止狀態，若不是判斷 agent number 是否為 0(為 pacman) 是的話取 max，不是的話 (ghost) 取 min

## 3 Expectimax

```

def Expectimax(agent, depth, gameState):
    if gameState.isLose() or gameState.isWin() or depth == self.depth:
        return self.evaluationFunction(gameState)
    if agent == 0:
        tmp = float(-1e9)
        for State in gameState.getLegalActions(agent):
            tmp = max(Expectimax(1, depth, gameState.getNextState(agent, State)), tmp)
        return tmp
    else:
        nextagent = agent + 1
        if gameState.getNumAgents() == nextagent:
            nextagent = 0
            depth += 1
        tmp = 0
        for State in gameState.getLegalActions(agent):
            tmp += Expectimax(nextagent, depth, gameState.getNextState(agent, State))
        return tmp/float(len(gameState.getLegalActions(agent)))
val = float(-1e9)
for State in gameState.getLegalActions(0):
    tmp = Expectimax(1, 0, gameState.getNextState(0, State))
    if tmp > val:
        val = tmp
        next = State
return next

```

與 minmax 大致上相通，但在 ghost 行動那邊因為 ghost 行動是 random 不是 optimal，所以在決定 min 時改成以 child node 的平均值代替

## 4 Q-learning

### 4.1 Value Iteration

```

def computeQValueFromValues(self, state, action):
    """
    Compute the Q-value of action in state from the
    value function stored in self.values.
    """
    """
    *** YOUR CODE HERE ***
    # Begin your code
    val = 0
    for trans_prob in self.mdp.getTransitionStatesAndProbs(state, action):
        t,p = trans_prob
        reward = self.mdp.getReward(state, action, t)
        gamma = self.discount
        val += p*(reward+gamma * self.getValue(t))
    return val

```

計算 value 的公式為  $\sum p * (\text{reward} + \text{gamma} * \text{value})$

```
    if (self.mdp.isTerminal(state)):
        return None
    val = util.Counter()
    for i in self.mdp.getPossibleActions(state):
        val[i] = self.computeQValueFromValues(state, i)
    return val.argmax()
```

在決定行動時，則計算整個矩陣內最大值得行動

## 4.2 Q-learning

```
def computeValueFromQValues(self, state):
    """
    Returns max_action Q(state,action)
    where the max is over legal actions. Note that if
    there are no legal actions, which is the case at the
    terminal state, you should return a value of 0.0.
    """
    """ YOUR CODE HERE """
    # Begin your code
    values = []
    for action in self.getLegalActions(state):
        values.append(self.getQValue(state, action))
    if len(values) != 0:
        tmp = float(-1e9)
        for val in values:
            tmp = max(val, tmp)
        return tmp
    else:
        return 0.0
```

一樣計算最大值

## 4.3 epsilon-greedy

```
# Begin your code
if (util.flipCoin(self.epsilon)):
    action = random.choice(legalActions)
else:
    action = self.computeActionFromQValues(state)
return action
# End your code
```

在決定行動時加入隨機值

## 4.4 Approximate Q-learning

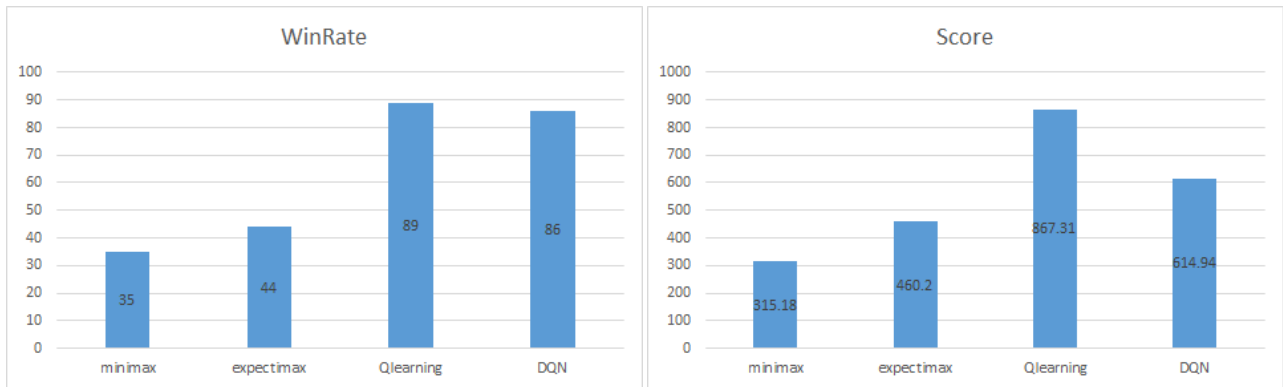
```
def update(self, state, action, nextState, reward):
    """
    Should update your weights based on transition
    """
    """
    *** YOUR CODE HERE ***
    """
    # Begin your code
    correction = (reward + self.discount * self.computeValueFromQValues(nextState)) - self.getQValue(state, action)
    features = self.featExtractor.getFeatures(state, action)

    for feature in features:
        self.weights[feature] += self.alpha * correction * features[feature]
    # End your code
```

更新的公式為  $w + \alpha[\text{correction}] * \text{feature}$ ,  $\text{correction} = \text{reward} + \gamma * \text{value} - \text{Qvalue}$

## 5 Compare

這邊都使用 smallClassic 進行測試，DQN 與 Q-Learnig 的 train 用 10000, test 的數量用 100, minimax, expectimax depth 用 3



Average Score: 315.18  
 Scores: 179.0, 1210.0, 276.0, -69.0, -105.0, -321.0, -446.0, -170.0, 1658.0, 74.0, 54.0, 1239.0, 79.0, 1486.0, 19  
 9.0, -112.0, 1029.0, 20.0, 1073.0, -44.0, -98.0, -827.0, 1422.0, -192.0, 1073.0, 245.0, -290.0, -268.0, -176.0, -154.0,  
 41.0, 160.0, 285.0, 190.0, -272.0, 1278.0, -60.0, -111.0, 1210.0, -384.0, -83.0, -153.0, 1151.0, -225.0, 840.0, 1212.0,  
 260.0, 836.0, -199.0, 985.0, -73.0, 608.0, 903.0, -215.0, -218.0, 1385.0, 917.0, 1067.0, -25.0, 73.0, 1217.0, -396.0, 91  
 7.0, 220.0, 93.0, 1195.0, 470.0, 194.0, 1242.0, -22.0, 946.0, 237.0, -89.0, -101.0, -226.0, -397.0, 14.0, -388.0, 34.0,  
 52.0, -597.0, 1551.0, 1214.0, -65.0, -102.0, 1576.0, 569.0, 929.0, 1179.0, 301.0, -136.0, -206.0, -167.0, 28.0, 1153.0,  
 966.0, 789.0, -343.0, -1207.0, -324.0  
 Win Rate: 35/100 (0.35)  
 Record: Loss, Win, Loss, Loss, Loss, Loss, Loss, Loss, Win, Loss, Loss, Win, Loss, Win, Loss, Loss, Win, Loss, Win,  
 a, Loss, Loss, Loss, Win, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Win, Loss, Loss, Win, L  
 oss, Loss, Loss, Win, Loss, Win, Win, Loss, Win, Loss, Win, Win, Loss, Loss, Win, Win, Win, Loss, Loss, Win,  
 Loss, Win, Loss, Loss, Win, Win, Loss, Win, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Loss, Win,  
 Win, Loss, Loss, Win, Win, Win, Win, Loss, Loss, Loss, Loss, Loss, Win, Win, Win, Loss, Loss, Loss

Figure 1: minimax

Average Score: 460.2  
 Scores: 1235.0, -106.0, 1273.0, -255.0, -297.0, -37.0, 1517.0, -359.0, 61.0, 234.0, 264.0, 1234.0, 10.0, -585.0,  
 -360.0, 1664.0, 360.0, 185.0, 1501.0, -430.0, 24.0, -951.0, 820.0, 1278.0, -30.0, 959.0, -289.0, 1175.0, 1223.0, 1098.0,  
 -123.0, 869.0, 1314.0, 193.0, 1356.0, 202.0, -21.0, 141.0, 1591.0, 822.0, 679.0, -100.0, 1331.0, 1461.0, 1229.0, 1696.0,  
 1340.0, -586.0, 1618.0, -46.0, -154.0, -903.0, 254.0, -338.0, -686.0, -373.0, -228.0, 1215.0, -144.0, 172.0, -239.0, 6  
 1.0, 910.0, 751.0, 163.0, -166.0, 1444.0, -192.0, 145.0, 668.0, -83.0, 1277.0, 94.0, 1228.0, -399.0, -21.0, 268.0, -725.  
 0, 1322.0, 513.0, 1412.0, 1280.0, 1035.0, 246.0, 21.0, -283.0, 1274.0, 1336.0, 365.0, 1454.0, 933.0, 324.0, -227.0, -389  
 0.0, -100.0, -371.0, 671.0, 1495.0, 481.0, 1236.0  
 Win Rate: 44/100 (0.44)  
 Record: Win, Loss, Win, Loss, Loss, Loss, Win, Loss, Loss, Loss, Loss, Win, Loss, Loss, Loss, Win, Loss, Loss, Win,  
 n, Loss, Loss, Loss, Win, Win, Loss, Win, Loss, Win, Win, Win, Loss, Win, Win, Loss, Loss, Loss, Win, Win, V  
 n, Loss, Win, Win, Win, Win, Loss, Win, Loss, Loss, Loss, Loss, Loss, Loss, Win, Loss, Loss, Loss, Win, V  
 n, Win, Loss, Loss, Win, Loss, Loss, Loss, Win, Loss, Win, Loss, Loss, Loss, Loss, Win, Win, Win, Win, Win, Lo  
 s, Loss, Loss, Win, Win, Loss, Win, Win, Loss, Loss, Loss, Loss, Loss, Win, Win, Win, Win, Win, Win

Figure 2: expectimax

Figure 3: Q-learning

Figure 4: DQN-1

Figure 5: DQN-2

- 理論上 DQN 應該要比 Qlearning 的結果好，但實驗結果並非如此，可能需要嘗試更多訓練參數
- 要把數學公式實作，要找到對應參數有點麻煩
- minimax 與 expectimax 跑起來有點花時間，原本想做 1000 筆的比較但跑到一半電腦盪了 QAQ

## 5