**인공지능 과제 2: Q learning 보고서**

2019055078 신채영

* Q table 기반의 Q learning 구현
* **코드설명(+주석):**

input.txt를 읽어 미로 지도를 읽고 states배열에 저장:

f = open("input.txt", 'r')

lines = f.readlines()

N = sum(1 for line in open('input.txt')) #입력된 줄 수 = 배열 크기

states = [[0]\*N ]\*N

#input

i=0

for line in lines:

states[i]=list(line.rstrip())

i=i+1

f.close()

미로 게임에 필요한 기본 정보 (rewards, 이동 방향, gamma, 시작점):

# rewards

rewards = {

'S': 0, #start point

'G': +100, #goal point

'T': +1, #bonus point #보너스 점수: 0, 10, 20

#'T': +10, #bonus point

#'T': +20, #bonus point

'B': -100, #bomb

'P': 0 #normal path

}

# possible actions

actions = {

'UP': (-1, 0), #4방향

'DOWN': (+1, 0),

'RIGHT': (0, +1),

'LEFT': (0, -1),

}

#gamma

gamma = 0.9

# initial

for i in range (N):

for j in range (N):

if (states[i][j]=='S'): #시작점: S

start\_position\_i = i

start\_position\_j = j

break

q를 초기화하고 현재 위치를 시작점으로 설정:

q = {}

current\_state\_i, current\_state\_j = start\_position\_i, start\_position\_j

goal=0 #골에 도착했는지 체크하는 함수

현재 자리에서 이동할 수 있는 방향을 담은 배열을 출력하는 함수:

def possible\_actions(i, j):

act=[]

for action in actions.items():

\_i, \_j = i + action[1][0], j + action[1][1]

if 0 <= \_i < len(states) and 0 <= \_j < len(states[i]):

act.append(action[0])

return act

Q표 초기화:

# Q표 초기화

def initialize\_q():

for i in range(0, len(states)):

for j in range(0, len(states[i])):

for action in possible\_actions(i,j):

q[i,j,action]=0

states[i,j]칸의 reward return 하는 함수:

def compute\_reward(i, j): #next i,j

return rewards[states[i][j]]

현재 위치에서 ‘action’ 방향으로 이동시키는 함수:

def take\_action(i, j, action):

return i+actions[action][0],j+actions[action][1]

현재 위치에서 q가 가장 큰 방향(policy)를 찾는 함수 (방향 return):

# Q값이 가장 큰 방향 선정

def best\_action\_policy(ii, jj):

return max( possible\_actions(ii,jj), key = lambda possible\_action: q[(ii, jj, possible\_action)])

공식을 사용해 q표를 업데이트해주는 함수:

def q\_update(i, j, action, next\_i, next\_j):

if (states[next\_i][next\_j]=='G'): #다음칸이 골이면 100

return 100

if (states[next\_i][next\_j]=='B'):

return -100

reward = compute\_reward(next\_i, next\_j)

return reward + gamma \* q[(next\_i, next\_j, best\_action\_policy(next\_i, next\_j))]

q\_update로 현재 위치의 표를 갱신하고 possible actions를 통해 이동할 수 있는 방향을 확인한 후 랜덤하게 선택해 이동하는 함수:

# q-learning

def q\_learning(i, j):

for act in possible\_actions(i, j): #이동가능한 방향에 대한 q값 모두 q\_update

ii, jj = take\_action(i, j, act)

q[(i, j, act)] = q\_update(i, j, act, ii, jj)

random\_act = random.choice(possible\_actions(i,j)) #random하게 방향 선택해서 이동

ii, jj = take\_action(i, j, random\_act)

return ii, jj

현재위치에서 q\_learning을 진행하는 함수

def game():

global current\_state\_i, current\_state\_j, q

current\_state\_i, current\_state\_j = q\_learning(current\_state\_i, current\_state\_j)

q learning을 마치고 표를 보고 이동경로를 출력하는 함수: (q\_learning 코드에서 update하는 부분을 지우고 q가 가장 큰 부분을 찾아 (best\_action\_policy 사용) 이동하는 코드를 추가함)

#경로찾기

def print\_answer(i, j):

###

Print\_answer를 call할 함수 : (현재위치에 대해 print\_answer)

def start():

###

시작:

# initialize Q function

initialize\_q()

cnt=100000

fw = open("output.txt","w") #output 파일

while(cnt>0): #100000(cnt)만큼 q-learning 실시

cnt=cnt-1

game()

current\_state\_i=0

current\_state\_j=0

while (goal==0): #골에 도착할 때 까지 완성된 표를 보면서 최적의 경로 탐색

start()

fw.write('\n')

fw.write(str(max(q[0,0,'RIGHT'],q[0,0,'DOWN']))) #첫번째 칸의 max Q를 출력

fw.close()

**실행결과:**

* 텍스트이(가) 표시된 사진

  자동 생성된 설명**input.txt:**
* **output.txt:**

Bonus=+1

텍스트이(가) 표시된 사진

자동 생성된 설명

Bonus=+10

텍스트이(가) 표시된 사진

자동 생성된 설명

Bonus=+20

* 답을 출력하지 못함

분석한 Q table을 출력해보면: (하이라이트=이동, 밑줄=값이 같음)

0 0 DOWN -100

0 0 RIGHT 85.2631578947368

0 1 DOWN 94.73684210526312

0 1 RIGHT 94.73684210526312

0 1 LEFT 76.73684210526312

0 2 DOWN 105.2631578947368

….

1 0 UP 76.73684210526312

1 0 DOWN 76.73684210526312

1 0 RIGHT 94.73684210526312

1 1 UP 85.2631578947368

1 1 DOWN 85.2631578947368

1 1 RIGHT 105.2631578947368

…..

이 테이블을 가지고 경로를 탐색하면:

* (0,0)에서 시작 -> RIGHT선택 -> (0,1)
* (0,1)에서 DOWN, RIGHT 가 같다. -> 구현한 코드상 max에서 앞에 있는 DOWN 선택

->(1,1)

* (1,1)에서 UP, DOWN이 같다. -> 구현한 코드상 max에서 앞에 있는 UP선택 -> 다시 (0,1)
* 이런식으로 2,7,2,7 ..로 무한 루프를 돌아 최적 경로를 찾지 못하고 output.txt가 close되지 못해서 프로그램을 강제종료 하지 않으면 output.txt에 아무것도 적히지 않는다

+) 경로탐색을 할 때 지나온 경로를 다시 가지 않게 해당 방향 q값은 0으로 처리해주고

+) 2,7,12와 같이 잘못 된 경로를 찾을 것을 방지하기 위해 max()함수에 들어간 두 Q값이 같을 경우 random하게 선택하게 한 후 restart를 여러 번 하게하면 성공적으로 최적의 경로를 출력할 수 있을 것으로 보인다.