지능시스템

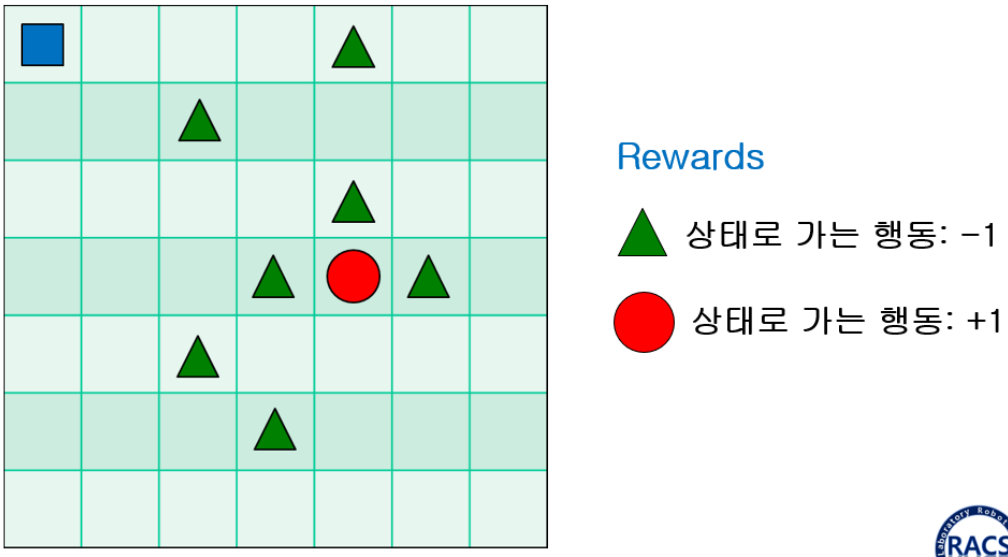
과제5

2019305059

이현수

숙제 5 (First Visit Monte Carlo Method)

- 다음의 7 × 7 Grid Map에서 출발점, 목표지점, 장애물이 각각 표시된 위치에 있을 때, 목표지점을 찾아가는 행동을 **First Visit Monte Carlo 방법**으로 구하고 학습 진행(epoch)에 따를 결과(학습된 각 상태의 상태함수값 및 정책에 의한 행동 확률을 격자 위에 표시)를 제시하시오. (제출: 5월 29일 6시까지, Google Classroom)



그리드 월드에서 좌표값은 위 사진과 같다.

● **environment.py 수정코드**

```
6   np.random.seed(1)
7   PhotoImage = ImageTk.PhotoImage
8   UNIT = 100  # 픽셀 수
9   HEIGHT = 7  # 그리드 월드 세로
10  WIDTH = 7  # 그리드 월드 가로
```

그리드 월드는 5*5에서 7*7로 수정한다.

```
36      # 캔버스에 이미지 추가
37      self.rectangle = canvas.create_image(50, 50, image=self.shapes[0])
38      self.triangle1 = canvas.create_image(450, 50, image=self.shapes[1])
39      self.triangle2 = canvas.create_image(250, 150, image=self.shapes[1])
40      self.triangle3 = canvas.create_image(450, 250, image=self.shapes[1])
41      self.triangle4 = canvas.create_image(350, 350, image=self.shapes[1])
42      self.triangle5 = canvas.create_image(550, 350, image=self.shapes[1])
43      self.triangle6 = canvas.create_image(250, 450, image=self.shapes[1])
44      self.triangle7 = canvas.create_image(350, 550, image=self.shapes[1])
45      self.circle = canvas.create_image(450, 350, image=self.shapes[2])
```

캔버스에 문제와 같이 추가한다.

```
98      # 보상 함수
99      if next_state == self.canvas.coords(self.circle):
100         reward = 100
101         done = True
102     elif next_state in [self.canvas.coords(self.triangle1),
103                         self.canvas.coords(self.triangle2),
104                         self.canvas.coords(self.triangle3),
105                         self.canvas.coords(self.triangle4),
106                         self.canvas.coords(self.triangle5),
107                         self.canvas.coords(self.triangle6),
108                         self.canvas.coords(self.triangle7)]:
109         reward = -100
110         done = True
111     else:
112         reward = 0
113         done = False
```

102~108줄 코드와 같이 triangle을 추가시킨다.

```python
    def text_value(self, row, col, contents, action, font='Helvetica', size=10,
                   style='normal', anchor="nw"):
        if action == 0:
            origin_x, origin_y = 7, 42
        elif action == 1:
            origin_x, origin_y = 85, 42
        elif action == 2:
            origin_x, origin_y = 42, 5
        elif action ==3:
            origin_x, origin_y = 42, 77
        else:
            origin_x, origin_y = 42, 42

        x, y = origin_y + (UNIT * col), origin_x + (UNIT * row)
        font = (font, str(size), style)
        text = self.canvas.create_text(x, y, fill="black", text=contents,
                                       font=font, anchor=anchor)
        return self.texts.append(text)

```

```python
    def print_value_q_all(self, q_table, value_table):
        for i in self.texts:
            self.canvas.delete(i)
        self.texts.clear()
        for x in range(HEIGHT):
            for y in range(WIDTH):
                state=[x,y]
                if str(state) in value_table.keys():
                    temp=value_table[str(state)]
                    self.text_value(y,x,round(temp,2),4)
                for action in range(0,4):
                    if str(state) in q_table.keys():
                        temp=q_table[str(state)][action]
                        self.text_value(y,x,round(temp,2),action)
```

각 칸에 상태함수값과 행동 확률을 적기위한 함수들이다.

● **mc_agent.py 수정코드**

```python
# 몬테카를로 에이전트 (모든 에피소드 각각의 샘플로 부터 학습)
class MCAgent:
    def __init__(self, actions):
        self.width = 7
        self.height = 7
        self.actions = actions
        self.learning_rate = 0.01
        self.discount_factor = 0.9
        self.epsilon = 0.1
        self.samples = []
        self.value_table = defaultdict(float)
        self.percentage_table = defaultdict(lambda: [0.0,0.0,0.0,0.0])
        self.reward_table = defaultdict(lambda: [0.0,0.0,0.0,0.0])
```

Width=7, height=7로 업데이트 한다.

```python
    # 모든 에피소드에서 에이전트가 방문한 상태의 큐 함수를 업데이트
    def update(self):
        visit_state=[]
        sample_size=len(self.samples)
        for i, forward in enumerate(self.samples):
            state, reward, done = forward
            state = str(state)
            if state not in visit_state:
                visit_state.append(state)
                G_t = 0
                for j, reverse in enumerate(reversed(self.samples)):
                    if sample_size-j-1<i:
                        break
                    rev_state, rev_reward, rev_done = reverse
                    rev_state=str(rev_state)
                    G_t=rev_reward+self.discount_factor*G_t
                value = self.value_table[state]
                self.value_table[state]=(value+self.learning_rate*(G_t-value))
```

```python
    # 가능한 다음 모든 상태들을 반환
    def possible_next_state(self, state):
        col, row = state
        next_state = [0.0] * 4

        if row!=0:
            next_state[0]=self.reward_table[str(state)][0]+self.discount_factor+self.value_table[str([col,row-1])]
        else:
            next_state[0] = self.reward_table[str(state)][0] + self.discount_factor + self.value_table[str(state)]
        if row!=self.height-1:
            next_state[1] = self.reward_table[str(state)][1] + self.discount_factor + self.value_table[str([col, row + 1])]
        else:
            next_state[0] = self.reward_table[str(state)][1] + self.discount_factor + self.value_table[str(state)]
        if col!=0:
            next_state[2]=self.reward_table[str(state)][2]+self.discount_factor+self.value_table[str([col-1,row])]
        else:
            next_state[2]=self.reward_table[str(state)][2]+self.discount_factor+self.value_table[str(state)]
        if col!=self.width-1:
            next_state[3]=self.reward_table[str(state)][3]+self.discount_factor+self.value_table[str([col+1,row])]
        else:
            next_state[3]=self.reward_table[str(state)][3]+self.discount_factor+self.value_table[str(state)]

```

```python
    def reward_state_action(self, state, action,reward):
        state = str(state)
        self.reward_table[state][action]=reward

```

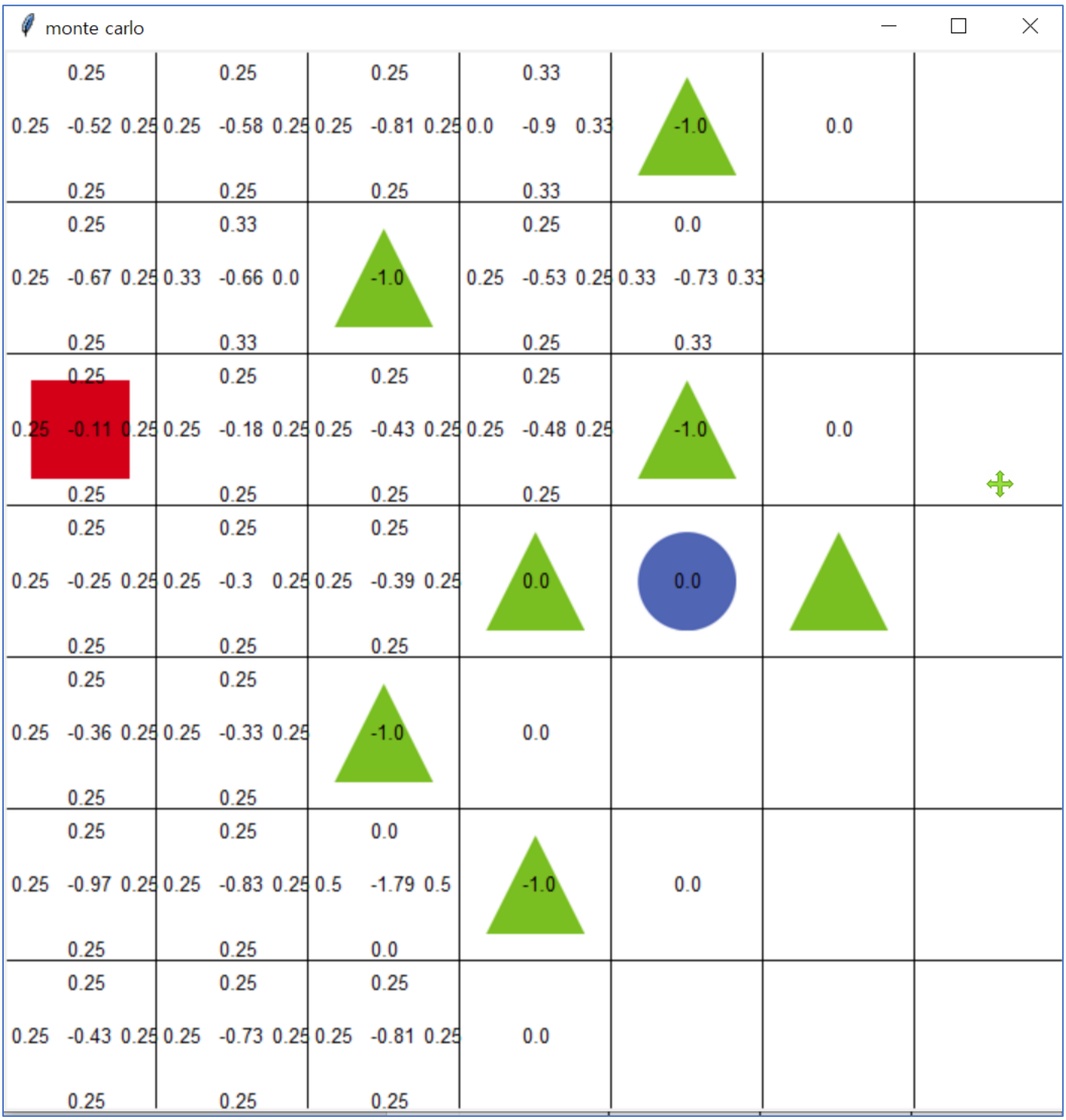
```python
    def update_percentage_table(self,state,next_state):
        value = -99999
        max_index = []

        result = [0.0,0.0,0.0,0.0]

        # 모든 행동에 대해서 [보상 + (감가율 * 다음 상태 가치함수)] 계산
        for index, action in enumerate([0, 1, 2, 3]):

            reward = self.reward_table[str(state)][action]
            next_value = self.value_table[str(next_state)]
            temp = reward + self.discount_factor * next_value

            # 받을 보상이 최대인 행동의 index(최대가 복수라면 모두)를 추출
            if temp == value:
                max_index.append(index)
            elif temp > value:
                value = temp
                max_index.clear()
                max_index.append(index)

        # 행동의 확률 계산
        prob = 1 / len(max_index)
        for index in max_index:
            result[index] = prob

        self.percentage_table[str(state)][0] = result[0]
        self.percentage_table[str(state)][1] = result[1]
        self.percentage_table[str(state)][2] = result[2]
        self.percentage_table[str(state)][3] = result[3]

```

행동에 대한 확률을 나타내기 위한 코드이다.

```python
# 메인 함수
if __name__ == "__main__":
    env = Env()
    agent = MCAgent(actions=list(range(env.n_actions)))

    for episode in range(10000):
        state = env.reset()
        action = agent.get_action(state)

        while True:
            env.render()

            # 다음 상태로 이동
            # 보상은 숫자이고, 완료 여부는 boolean
            now_state, next_state, reward, done = env.step(action)
            agent.save_sample(next_state, reward, done)
            #agent.reward_state_action(next_state, action, reward)
            # 다음 행동 받아옴
            action = agent.get_action(next_state)

            agent.reward_state_action(now_state, action, reward)
            agent.update_percentage_table(now_state, next_state)

            env.print_value_q_all(agent.percentage_table, agent.value_table)
            # 에피소드가 완료됐을 때, 큐 함수 업데이트
            if done:
                agent.update()
                agent.samples.clear()
                break
```

메인함수 코드이다.

●실행결과



원하는 결과가 나오지 못함. 완벽하게 못함.