Seoul National University
College of Engineering
Department of Naval Architecture and Ocean Engineering
1, Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea

Fall 2020

딥러닝

PA # 5

Instructor name	김태완 교수님
Student name	이용준
Department	조선해양공학과
Student ID	2015-19595
Submission date	2020.12.15

Contents

1. Problem Definition

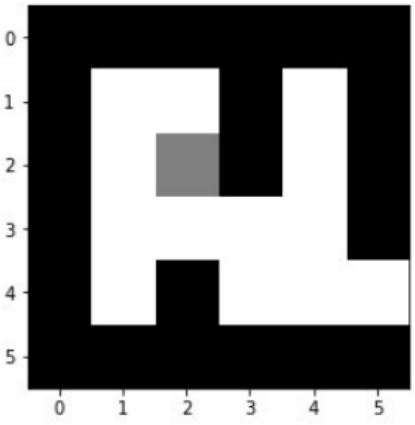


Figure 1 Simple Maze

- **강화 학습 코드를 구현하여 미로를 빠져나오는 학습하도록 한다.**
- **DQN 방식을 사용하여 회색 grid가 5행 6열의 출구로 빠져나올 수 있도록 한다.**
- **주어진 skeleton code 내의 To do 부분을 적절히 채워 넣어 코드를 완성시킨다.**

2. Problem Analysis

Q- learning : 예측되는 행동의 보상의 총합을 최대화할 수 있는 방향으로 행동한다.

Q-learning finds a policy that is optimal in the sense that it maximizes the expected value of the total reward over any and all successive steps, starting from the current state.[1]

"Q" names the function that returns the reward used to provide the reinforcement and can be said to stand for the "quality" of an action taken in a given state.[2]

$$Q(s,a) = r(s,a) + \gamma \max_a Q(s',a)$$

ε – greedy의 필요성: 최대의 Q값만을 따라가도록 하면, 매번 같은 곳에만 도달하기 때문에 최적의 해를 찾지 못한다. 따라서 가지 않은 길을 찾아 가야 더 좋은 해를 찾아 갈수 있다.

3. Code Explanation

이번 과제는 주어진 skeleton code의 To do 부분을 채우는 것이 목적이었다.

작성해야 할 To do는 모두 다섯 군데였다.

<Agent class>

action_and_next_state :

```
############### To do ##############
memory_temp[:8] = np.copy(self.state) # Current state
############### To do ##############
```

임시 메모리에 Current state를 복사한다.

```
############### To do ##############
memory_temp[8] = np.copy(action) # action
memory_temp[9] = np.float32(done) # done or not
memory_temp[10] = np.copy(reward) # reward
memory_temp[11:] = np.copy(next_state) # next state
############### To do ##############
```

임시 메모리에 각각의 항목을 복사한다.

```
if(len(self.replay_memory) >= self.memory_size):
    ############### To do ##############
    self.replay_memory[np.random.randint(len(self.replay_memory))] = np.copy(memory_temp)

else:
    self.replay_memory.append(np.copy(memory_temp))
    ############### To do ##############
```

리플레이 메모리의 크기가 메모리 사이즈보다 커졌을 경우 리플레이 메모리 속 일부분에 임시 메모리를 덧붙여 복사하고 그렇지 않으면 임시 메모리를 리플레이 메모리에 추가한다.

train_estimator :

```
############### To do ##############
next_state = np.copy(replay_memory[:, 11:])
Q_sa_next = np.copy(self.estimator.predict(next_state))
############### To do ##############
```

리플레이 메모리를 이용하여 next_state를 초기화하고, 이를 이용하여 Q_sa_next를 설정한다.

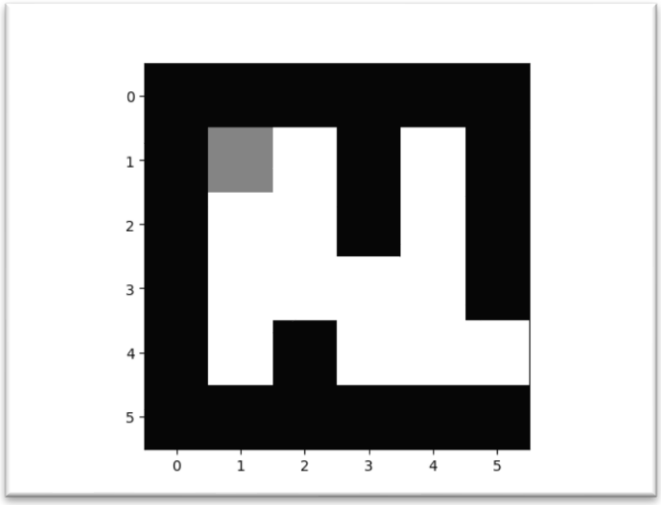
```
############## To do ##############

if replay_memory[i, 10] == 1:
    Q_sa[i][np.int32(replay_memory[i, 8])] = np.copy(replay_memory[i, 10])
else:
    Q_sa[i][np.int32(replay_memory[i, 8])] = np.copy(replay_memory[i, 10] + self.discount_reward*np.max(Q_sa_next[i,:]))

############## To do ##############
```

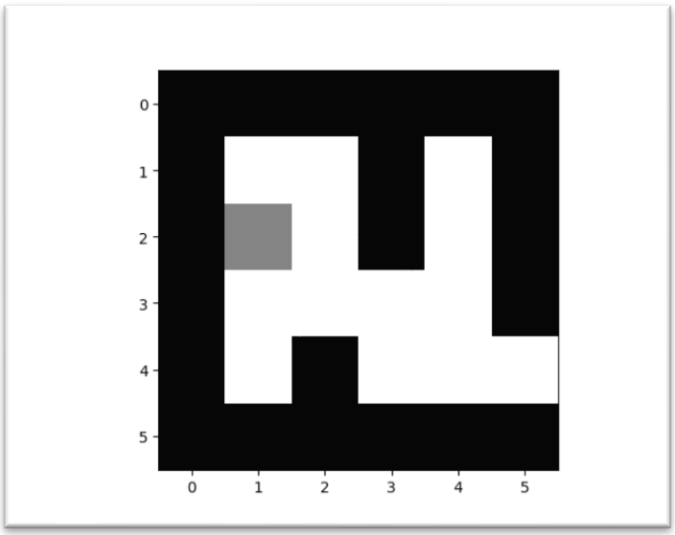
리워드 값이 1인지 아닌 지에 따라서 Q_sa의 해당 값을 위와 같이 설정한다.

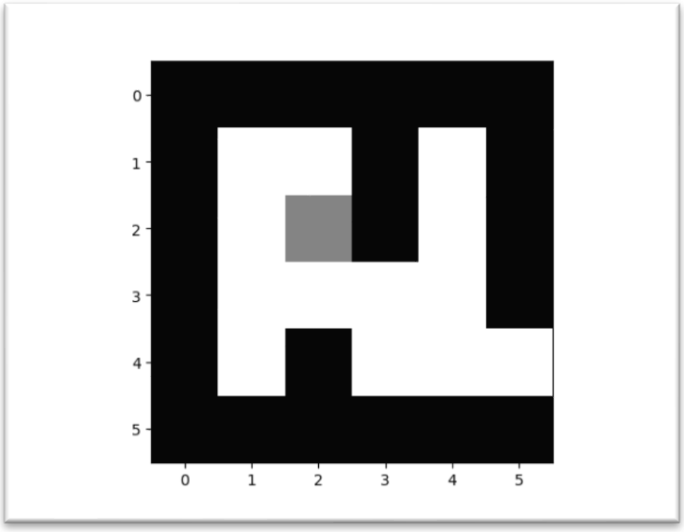
4. Result & Conclusion

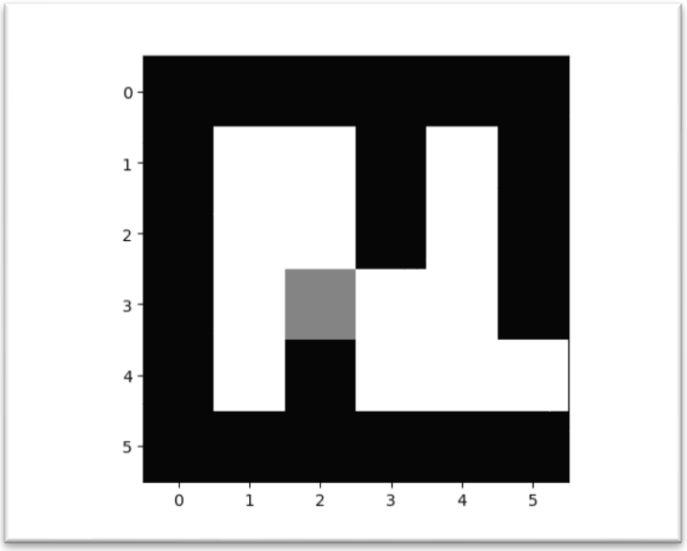
4.1 Result

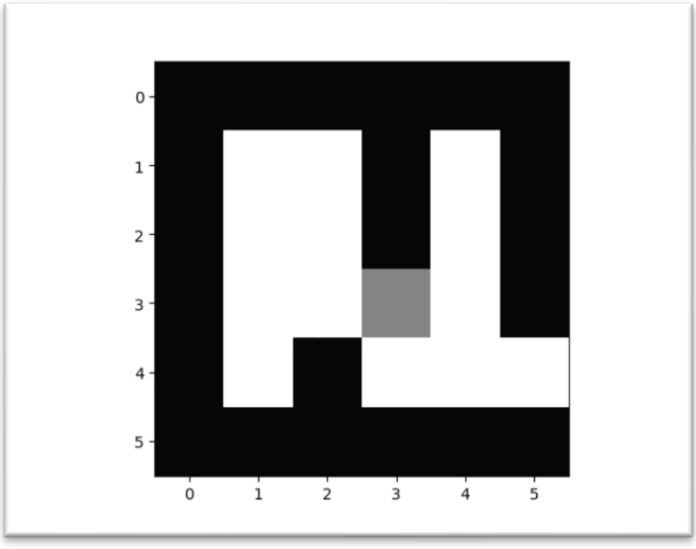
코드 작성을 완료하고 실행시켜 보았다. 에이전트의 시작 위치는 (1,1)이었다. 점차 출구를 향해 이동하는 모습을 확인할 수 있었다.

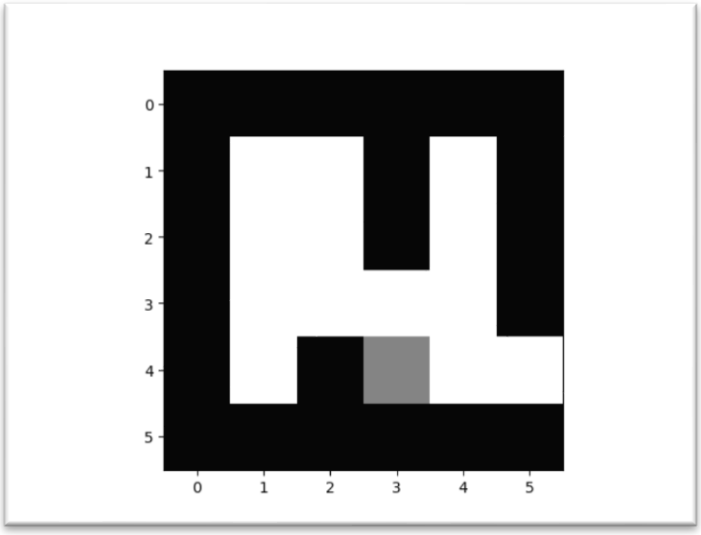


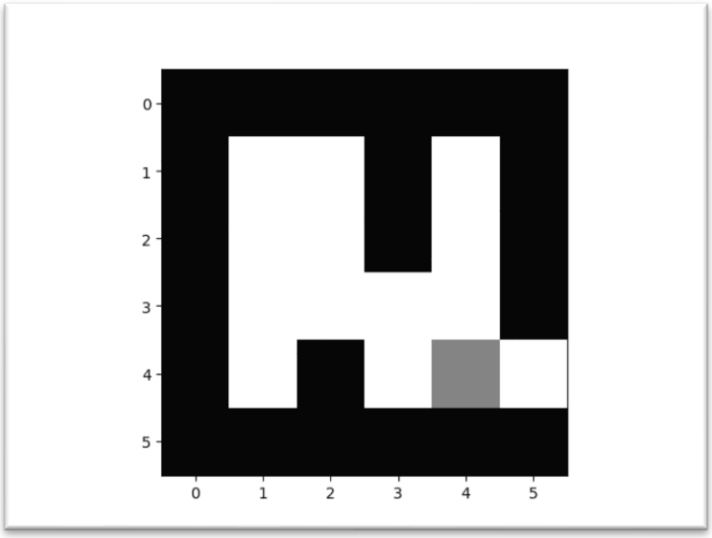



0
1
2
3
4
5
0 1 2 3 4 5


0
1
2
3
4
5
0 1 2 3 4 5


0
1
2
3
4
5
0 1 2 3 4 5





4.2 Conclusion

이번 과제를 통하여 비록 코드의 일부분을 작성하는 것이었지만, 강화학습의 한 종류인 Q-learning 에 대한 실습을 해볼 수 있었다. 주어진 코드를 완벽히 이해하는 것에는 어려움이 있어서, 시간이 된다면 좀 더 자세히 공부를 해보고 싶은 마음이 생겼다.