석천-bot

보고서 (word) 양식

**I. 프로젝트 제목, 팀원 이름 및 소개**

**II. 프로젝트 요약: 중요한 내용을 한 눈에 볼 수 있도록 1단락으로 작성**

(프로젝트 제안서 내용 그대로임, 좀 더 요약하고 기술적 내용, 알고리즘 등 추가할 것)

주어진 공간에서 특정 지점들을 포함하면서도 많은 범위를 청소할 수 있는 최적 경로를 찾는 청소 로봇을 구현해보고자 한다. 정해진 Map에서 최적 경로를 탐색하는 것에 그치지 않고 실제 청소 로봇과 동일하게 임의로 주어지는 map을 탐색해서 작성하고, 이후에 청소가 필요한 부분 등의 학습의 보상으로 얻게 하여 전반적인 과정을 수행하고자 한다. Map이 변경될 경우 (Environment 변화) 이전 map에서 강화학습한 알고리즘이 유효한가 같이 살펴보고자 한다.

**III. 본문**

**1.서론: 동기 및 문제 설명, 참고문헌 및 관련 내용 기술**

1)동기

바쁜 일상에서 집안일을 대신해주면서도 어지럽혀진 방, 짧은 시간 내에 많은 범위를 청소할 수 있는 로봇에 대한 니즈가 커졌다. 또한 보편화 되어 쉽게 접할 수 있다. 이를 강화 학습을 통해 구현해보고자 한다.

로봇이 연속적인 작업을 이전의 특강을 통해서 접하였던 강의가 motivation이 되었다. 프로젝트주제를 정하면서 로보틱스는 고전적인 강화학습의 적용 분야인 것을 알게 되었다. 로봇의 경로 탐색을 위한 강화학습은 고전적인 활용 예시라고 하지만 많은 확장성을 갖고 있고 넓은 범위의 응용 분야를 가질 있다고 생각한다.

강화학습 알고리즘 중 Q-Learning을 사용하여 문제를 해결하고자 한다. Q-Learning은 Agent가 Environment와의 상호작용으로 특정 상태 s에서의 특정 행동 a에 대한 Q-Value를 계산하며 Q-Table을 Update 한다. 이때 Q-Value를 Update하는 수식은 다음과 같다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

새 Q-Value는 현재 상태에서 해당 행동의 Q-Value에 새로운 보상 정보와 현재 Q-Value의 차이에 학습율 alpha를 곱한 값을 더한다. 여기서 새로운 보상 정보는 현재 상태 s에서 행동 a를 수행했을 때 즉시 얻는 보상과 다음 상태 s’에서 가능한 행동 중 가장 높은 Q-Value에 할인율 gamma를 곱한 값을 더한 값이다.

alpha가 1에 가까울 수록 새로운 정보가 Q-Value에 크게 반영되어 빠르게 학습하나 Noise에 민감할 수 있으며 0에 가까울 수록 기존 정보가 유지되며 새로운 정보의 반영이 적어져 학습이 느려질 수 있다.

gamma가 1에 가까울수록 미래 보상을 중요하게 여기며, 장기 보상을 최대화하려고 하며 0에 가까울수록 즉시 보상에 집중한다.

더 자세한 알고리즘 개념와 진행 방식은 아래 본론에서 작성하였다.

**2.본론: 수집한 데이터와 소스, 문제 해결의 방법 기술**

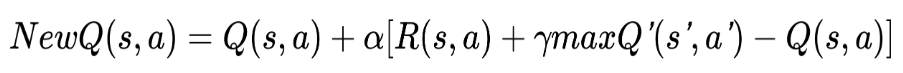
**1) 수집한 데이터**

강화학습을 진행하기 위해서는 Environment와 Agent, Policy, Reward Function, Value Function이 필요하다. 이 프로젝트의 Environment는 청소할 공간, 우리의 방이 될 것이고 Agent는 방을 청소할 로봇이다. Environment는 임의로 지정한 크기의 Grid 공간으로, Agent는 Grid 공간 내에서 up, down, left, right의 행동을 하는 객체로 설정했다. 강화학습의 알고리즘 중 Q-Learning을 사용했으며, Q-Learning의 Policy, Reward Function, Value Function은 다음과 같다.

- Policy: Policy는 Agent가 특정 상태 s에서 어떤 행동 a를 취할지 결정하는 전략으로, Q-Learning에서는 Q-Value(상태-행동 가치 함수)를 기반으로 결정된다. 가장 높은 Q-Value를 가진 행동을 선택하며, 일반적으로 epsilon-Greedy policy를 사용하여 Exploration과 Exploitation를 확률적으로 수행한다.

- Reward Function: Agent가 특정 상태 s에서 특정 행동 a를 취했을 때 즉각적인 Feedback을 제공하는 함수로, Q-Learning에서 이를 Q-Value Update에 사용한다. Grid에서 해당 좌표의 값을 Reward로 사용하도록 설계했다.

- Value Function: 특정 상태나 상태-행동 쌍의 Value를 평가하는 함수로, Q-Learning에서는 행동 가치 함수(Q-Value)를 사용한다. Q-Value는 특정 상태 s에서 특정 행동 a을 수행했을 때 기대 보상을 나타낸다. Agent는 Q-Value를 Update하여 최적의 Policy를 학습하며, Update Formula는 아래와 같다. [16]



- Q-Learning은 model-free and value-based 알고리즘으로 Agent가 Environment와의 상호작용을 통해 특정 상태 s에서의 행동 가치를 계산, 최적의 행동을 취할 수 있도록 학습하기 때문에 State Transition Probability를 명시적으로 사용하지 않는다. [12]

**2) 소스와 문제 해결 방법**

**Import Module**

필요한 Module들을 Import 한다. 여러 계산을 위한 numpy module과, 난수 생성을 위한 random, iteration 작업에 대한 progress를 확인할 수 있는 tqdm를 import 했다.



**Grid Class**

Agent의 학습이 이루어질 환경인 Grid Class를 정의했다.

[\_\_init\_\_] 객체를 생성할 때 Grid의 가로 세로 크기와 시작 위치, 목표 위치를 입력 받는다. 이후 해당 크기의 Zero Matrix를 만든다. 지금 상태를 시작 위치로 설정하고, 목표 위치도 설정한다. 이후 해당 목표 위치의 값을 Reward 값으로 설정하고, Grid 내에서 행해지는 행동들을 담은 List도 생성한다.

[set\_obstacles] 장애물 좌표들이 담긴 List를 입력 받고, 해당 좌표들에 Penalty를 설정한다.

[get\_map] Grid의 좌표들 간 이동 가능 여부를 확인할 수 있는 Map Matrix를 만든다. 이번 알고리즘을 진행하면서 해당 작업을 Grid를 Mapping 했다 고 표현했다. 해당 Matrix는 Grid의 좌표 개수를 각각 row, column으로 하는 n\*n Matrix다(n = w \* h). Grid의 2차원 좌표 (a, b)는 Matrix의 a \* w + b 1차원 값으로 변환되며, Matrix에서 (c, d)는 해당 c 좌표에서 d 좌표로의 이동 가능 여부를 나타낸다. Grid를 2중 for문으로 탐색하며 up, down, left, right 수행 시 해당 위치의 값이 Penalty인지 확인하며, Penalty면 이동 불가를 의미하도록 Matrix에 0으로 입력한다. 이후 생성된 Matrix를 반환한다.

[reset] 현재 상태를 시작 위치로 초기화하고 반환한다.

[step] 행동을 입력 받고 수행한다. 우선 현재 상태를 x, y 변수에 담아두고 현재 상태를 Matrix의 1차원 좌표로 변환한다. 이후 While 반복문에서 액션에 따라 좌표를 변경한다. min(), max() 함수를 통해 벽에 맞닿은 위치에서 벽을 향한 이동은 수행되더라도 좌표가 변경되지 않는다. 이후 변경된 좌표도 Matrix의 1차원 좌표로 변환한다. 이렇게 변환된 두 1차원 좌표를 이용해 Matrix에서 해당 좌표간 이동이 가능한지 확인한다. 이동이 가능하다면 While문을 탈출한다. 이전 좌표 값과 이후 좌표 값이 같다면 벽에 부딪혀 좌표 이동이 발생하지 않은 것이며, 그렇지 않다면 Matrix에서 이동 불가능으로 확인된 것으로 두 경우 다 행동을 재 선택한다. 이렇게 이동한 좌표를 현재 상태로 설정하고 Grid의 해당 좌표에서 Reward를 가져온다. 만약 이동한 좌표가 목표 위치와 같다면 완료 변수를 True로 설정하고, 이후 현재 상태, 행동, 보상, 완료여부를 반환한다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

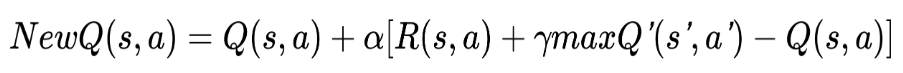
**Learning Agent Class**

Q Learning을 진행할 Agent Class를 선언한다.

[\_\_init\_\_] Agent 객체를 생성하면서 Grid와 가로 세로 크기, 그리고 세 Parameter alpha, gamma, epsilon을 입력 받는다. 이후 Grid와 같은 크기를 가진 2차원 Zero Matrix에 Action들의 개수만큼의 차원을 추가한 3차원 Q-Table을 생성한다.

[choose\_action] 현재 상태에서 행동을 선택한다. 난수를 발생하여 epsilon보다 작다면 Exploration을 수행하며, 행동을 무작위로 선택한다. epsilon보다 난수가 크다면 Exploitation을 수행한다. 현재 상태의 행동들의 Q-Value들 중 최대값을 구해 이를 행동으로 선택한다. 최대값을 가지는 행동이 여러 개라면 이 중 무작위로 선택한다.

[update\_q\_value] 현재 상태와 행동, 보상, 다음 상태를 입력 받아 Q-Value(Q-Table)를 Update 한다. 현재 상태의 행동들의 Q-Value들 중 최대값을 구한다. 이 값을 Q-Value를 Update하는 식에 사용한다. 해당 식은 아래와 같다.



Update된 Q-Value는 현재 상태의 선택된 액션의 Q-Value와 Reward, 다음 상태의 가장 좋은 행동의 Q-Value와 현재 Q-Value의 차이를 이용해 연산 된다. gamma와 alpha는 해당 연산에서 사용되며 Update되는 정도에 영향을 미친다.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

[train] 몇 차례 학습을 진행할지 입력 받고 학습을 진행한다. tqdm module을 이용하여 학습 진행의 진도를 파악한다. 현재 상태를 받아오고 종료 여부를 False로 설정한 뒤 While 반복문에 들어간다. 행동을 선택한 후 이에 따른 다음 위치, 보상, 종료 여부를 Grid의 step method를 통해 계산한다. 이 과정에서 수행 불가능한 행동일 경우 재 선택되므로 step method에서 행동도 반환 받는다. 이후 해당 정보들로 Q-Value를 Update하고 이동이 완료된 좌표를 현재 상태에 대입한다.

[get\_q\_table] 현재 Q-Table을 반환한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**장애물 생성 함수**

장애물의 개수와 시작 위치, 목표 위치, Grid의 가로 세로 크기를 입력 받고 장애물들을 생성한다. 장애물들의 좌표를 담을 빈 List를 생성한다. 해당 List의 크기가 사전에 입력 받은 장애물의 개수와 같아질 때까지 While 반복문을 수행한다. x, y 좌표를 Random module을 통해 생성하고, 이 좌표가 시작 위치나 목표 위치와 다르면 장애물 List에 추가한다. 추가한 다음 해당 List를 Set으로 변환했다가 다시 List로 변환하여 중복된 좌표가 있으면 제거한다. 이렇게 생성된 장애물 좌표 모음 List를 반환한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**환경 및 파라미터 설정**

Grid의 크기, 시작 위치와 목표 위치, 장애물의 개수, alpha, gamma, epsilon Parameter를 설정한다. 이후 Grid와 장애물들을 생성한다. 생성한 장애물 좌표를 이용해 Grid에 장애물을 추가하고, Grid 공간 내에서 좌표간 이동 가능 여부를 확인하는 Map Matrix를 생성한다. 이후 이 환경에서 Q Learning을 진행하는 Agent를 생성한다.

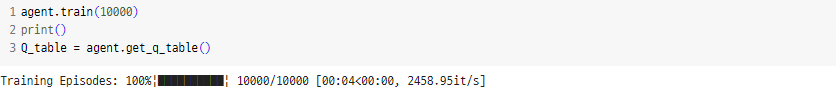
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

각 장애물들의 좌표와 Grid 환경을 확인하도록 출력하였다.

**Agent 학습**

Agent를 지정 횟수만큼 학습시키고, Q-Table을 반환하여 저장한다.



**Q-Table 변환**

생성된 Q-Table은 (가로\*세로\*행동) 차원의 배열로, Grid의 각 상태에 따른 행동 별 Q-Value를 확인하기 직관적이지 않다. 이를 (행동\*가로\*세로) 차원의 배열로 변환하였다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**Q-Table 확인**

차원을 변환한 Q\_table\_actions와 기존 Q\_table을 확인하였다.

텍스트, 폰트, 스크린샷, 문서이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

**3) 알고리즘 작성 중 발생한 문제들**

**- 최대 Q-Value를 가지는 행동 중복 시 선택 문제**

특정 상태 s’에서의 행동 각 a’의 Q-Value 중 같은 값이 있을 경우 가장 앞의 a’의 index를 가져오고 있었다. 그로 인해 Max Q-Value가 중복 발생할 시 up, down, left, right 순으로 우선 실행되게 되었고, 이는 우리가 설계한 알고리즘의 진행 방식과는 다르기 때문에 수정이 필요했다. 이는 numpy module의 **numpy.argmax** 함수를 사용했기 때문에 발생한 것으로, 구현 방식을 바꾸어 해결했다. 우선 a’의 최대값을 도출한 다음 그 값과 같은 값을 가지는 모든 index를 가져온다. 이후 그 index들 중 하나를 무작위로 선택하여 a’로 선택한다.

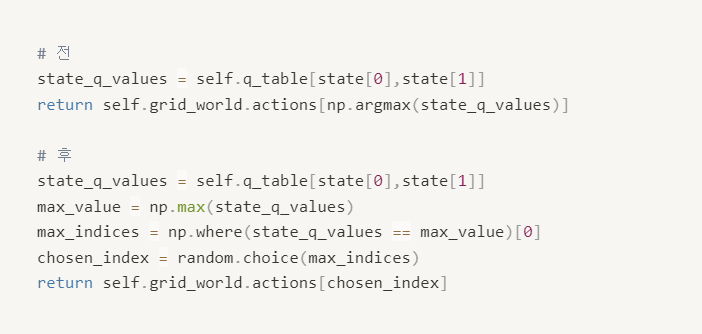
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

numpy의 docs 중 argmax method. 여러 개의 최대값이 발생하면 그 중 첫번째 값을 반환한다.



**- Agent의 Grid Environment 탐색**

Grid를 Agent에게 전달하고 Agent는 Grid를 참고해가며 이동과 학습을 반복하도록 알고리즘을 설계했다. 그러나 이는 로봇청소기가 처음부터 방의 구조를 모두 알고 있는 것과 다름없는 진행 방식이라 판단해 격자 구조를 우선 탐색만 하며 특정 위치 A에서 다음 위치 B로 이동이 가능한지를 파악하는 Mapping 기능을 구현했다. Grid는 각 좌표마다 값을 가지는데, 목적지는 Reward, 장애물은 음수 Penalty, 그리고 나머지 이동 가능한 좌표는 0이다. Grid를 생성한 다음 이를 2중 for문으로 탐색하며 up, down, left, right 이동이 가능한지 확인한다. 이동하려는 좌표의 값이 0보다 작다면 0, 그렇지 않다면 1로 표기한 Map을 생성해 이후 이동 함수에서 이 Map을 참조하여 불가능한 좌표로의 이동은 발생하지 않게 한다. 특정 좌표에서 각 x, y축에 대해 -1 또는 +1 연산한 좌표에 대해 판별하므로 해당 좌표에서의 제자리 이동은 발생하지 않아 Map 내에서 현재 좌표로의 이동 또한 불가하다. 주어진 2차원 Grid를 탐색하며 up, down, left, right 이동 가능 여부를 나타내는 Map을 생성하는 Code는 아래와 같다. 해당 Map은 전체 좌표의 수를 크기로 하는 정방행렬로, [현재좌표][이동할 좌표]가 1이면 이동 가능, 0이면 이동 불가를 나타낸다.

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

출력 결과는 다음과 같다. 3\*3 Grid와 총 좌표 9\*9 Map Matrix로, Map Matrix의 (0,1) = 1는 (0,0)에서 (0,1)로 이동이 가능하다는 의미, (0,3) = 0은 (0,0)에서 (1,0)으로 이동이 불가능하다는 의미다. Grid의 좌표가 Map Matrix의 좌표로 변환되는 수식은 (x index)\*(num of column) + (y index)다.

텍스트, 스크린샷, 폰트, 디자인이(가) 표시된 사진

자동 생성된 설명

**- 이동 불가한 위치로의 Agent 이동**

Agent가 Action을 결정하고 Grid 내에서 이동할 때 Grid 밖으로 이동하려고(벽에 부딪히는 경우) 하거나, 장애물에 부딪힐 경우 Action을 다시 고르도록 Code를 개선했다. Action 결정 후 이동할 좌표를 계산한 다음, 해당 좌표가 이동 가능한 곳인지 Grid를 Mapping한 Matrix에서 확인하여 가능하다면 이동하고, 벽에 부딪히는 이동이라 좌표 변화가 없는 경우 혹은 이동하고자 하는 위치가 장애물이 있어 이동이 불가할 경우에는 Action을 다시 무작위 선택하도록 했다.

Code 1차 개선 이후에도 장애물을 향한 Action의 Q-Value가 0이 아닌 문제가 발생했다. 예를 들어 (0,1)이 장애물일 때, (0,0)의 “right” Action의 Q-Value가 13.148이었다. 이는 앞서 개선한 이동 불가능 시 Action 재 선택 후 해당 Action을 반한해야 했으나, 반환하지 않고 이후 Q-Value Update 시 기존의 불가능한 Action을 그대로 사용해서 발생한 오류이다. 이를 해결하기 위해 Action을 재 선택하고 해당 Action을 반환, Q-Value Update시 사용하게 하여 이동 불가능한 좌표로 향하는 Action의 Q-Value는 0에서 Update되지 못하도록 Code를 2차 개선했다.

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

**3.결과: 실험 결과 작성**

주어진 Grid 환경에서 Agent가 잘 학습했는지 확인하기 위해 Grid 구조에 따른 각 위치 별 Q-Value Table을 확인해보았다.

텍스트, 폰트, 타이포그래피, 디자인이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

우선 각 행동을 수행했을 때 벽에 부딪히게 되는 좌표들은 모두 Q-Value가 0으로 Agent의 선택을 유도하지 않음을 볼 수 있다. 또한 이동했을 때 장애물을 만나게 되는 좌표의 행동들 또한 Q-Value가 0으로 학습한 것을 알 수 있다. 목적지인 (4,4)로 갈 수 있는 길은 (4,2)에서 Right 행동을 두 번 진행해야 하는데, 그렇기 때문에 해당 좌표에서의 Right 행동의 Q-Value가 높은 값을 보이는 것을 알 수 있다. 반복을 통해 Agent가 보상을 얻기 위해 어느 방향으로 움직여야 하는지를 환경과의 상호작용으로 학습했음을 알 수 있다.

성능 확인을 위해 학습의 반복에 따른 시작 위치에서 목표 위치로 가는데 걸리는 시간을 확인해보았다. 우선 위와 같은 크기의 Grid에 같은 개수의 장애물을 두고 각 학습 반복 횟수당 Agent가 Grid 내에서 움직인 걸음 수를 비교해보았다.

텍스트, 라인, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

학습이 진행되면서 Steps가 감소하는 것을 확인할 수 있었다. 학습이 원활히 진행되었으며, 보다 복잡한 환경에서도 학습이 진행되는 지 확인해보기 위해 100\*100 Grid에 장애물이 500개인 환경에서 학습을 진행해보았다. 텍스트, 그래프, 라인, 스크린샷이(가) 표시된 사진

자동 생성된 설명

마찬가지로 초기에는 굉장히 먼 걸음을 하지만 학습이 진행될수록 Steps가 감소하는 것을 볼 수 있다. 여러 참고문헌에서 Epsilon Parameter를 학습을 진행하며 감소시키는 Decaying e-greedy를 사용하는 것을 확인했다. 이를 채택하여 1000 episode 학습을 하며 100 episode당 Epsilon을 0.005 감소시키며 학습하였고, Steps를 다시 확인했다. 그 결과 성능이 향상됨을 확인할 수 있었다.

그래프, 라인, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명

**4.토의내용: 멤버들 각자의 의견 작성, 프로젝트 진행 과정, 프로젝트를 완성하기 위해 했던 학습내용 등**

- 토의한 주제들: 알고리즘 선택(강화학습 / 세부 알고리즘), 환경 구축(미로 or 광장, 장애물 여부), 보상 체계, 환경 시각화(unity, pybullet 등 여러 대안), 성능 평가 지표

1) 의견

2) 진행 과정

3) 학습내용

**5.참고문헌: 저자, 제목, 출처 (예: 논문지 이름, 출판사 이름, 웹 주소), 페이지, 연도**

- Paper

[1]“심층 강화학습을 이용한 로봇의 자율 경로 탐색 알고리즘”, 선우영민, 부경대학교 석사학위논문, 2022

[2]Destination Path Search using Reinforcement Learning Technique, 이태경 ( Taekyung Lee ) , 전준리 ( Junri Jeon ), 한국정보처리학회 2013.05

[3]Local Path Planning and Obstacle Avoidance System based on Reinforcement Learning, 한국컴퓨터정보학회 2019

[4]Journal of Korea Multimedia Society Vol. 21, No. 8, August 2018(pp. 969-981)

[5]경로 탐색 기법과 강화학습을 사용한 주먹지르기동작 생성 기법, 홍정모 2018

[6]Multi-objective Path Finding Using Reinforcement Learning

[7]Implementation of Q-Learning algorithm for solving maze problem

- Book

[8]PyBullet Quickstart Guide

[9]강화학습 첫걸음(한빛미디어)

[10]https://dnddnjs.gitbooks.io/rl/content/markov\_decision\_process.html

- Blog, Article

[11]<https://limitsinx.tistory.com/153>

[12]<https://medium.com/@datathon/exploring-q-learning-a-model-free-and-value-based-reinforcement-learning-algorithm-8934e3d1ca1f>

[13]<https://www.samyzaf.com/ML/rl/qmaze.html>

[14]<https://www.geeksforgeeks.org/what-is-reinforcement-learning/>

[15]<https://www.geeksforgeeks.org/q-learning-in-python/>

[16][https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning#what\_is\_qlearning](https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning#what_is_qlearning(q)

- Github

[17]<https://github.com/giorgionicoletti/deep_Q_learning_maze/tree/master>

[18]<https://github.com/sayakpaul/FloydHub-Q-Learning-Blog/blob/master/Q-Learning%20using%20numpy.ipynb>

[19]<https://github.com/erikdelange/Reinforcement-Learning-Maze/tree/master>