

|  |
| --- |
| 인공지능 과제 보고서 |
|  |



2019182025 원정우

목차

[1. K-평균 클러스터링(K-means clustering) 4](#_Toc184855711)

[1.1 문제 인식과 준비 4](#_Toc184855712)

[1.2 해결 과정 및 코드 구현 4](#_Toc184855713)

[1.2.1. 초기 클러스터 중심 설정 4](#_Toc184855714)

[1.2.2. 반복적인 클러스터링 수행 5](#_Toc184855715)

[1.2.3. 반복적인 클러스터링 수행 5](#_Toc184855716)

[1.2.4. 출력 코드 6](#_Toc184855717)

[1.3 결과 7](#_Toc184855718)

[2. 유전 알고리듬(숫자 추측하기) 8](#_Toc184855719)

[2.1 문제 인식과 준비 과정 8](#_Toc184855720)

[2.2 해결 과정 및 코드 구현 8](#_Toc184855721)

[2.2.1. 정답 숫자 생성 8](#_Toc184855722)

[2.2.2. 초기 개체군 생성 9](#_Toc184855723)

[2.2.3. 적합도 계산 9](#_Toc184855724)

[2.2.4. 적합도 계산 9](#_Toc184855725)

[2.2.5. 새로운 자손 후보 생성 9](#_Toc184855726)

[2.2.6. 돌연변이 생성 10](#_Toc184855727)

[2.2.7. Main 10](#_Toc184855728)

[2.3 결과 11](#_Toc184855729)

[3. A\* 알고리즘 12](#_Toc184855730)

[3.1 문제 인식과 준비 과정 12](#_Toc184855731)

[3.2 해결 과정 및 코드 구현 12](#_Toc184855732)

[3.2.1. Node 13](#_Toc184855733)

[3.2.2. 휴리스틱 함수 13](#_Toc184855734)

[3.2.3. A\* 함수 13](#_Toc184855735)

[3.3 결과 15](#_Toc184855736)

[4. 팀원 별 역할분담 및 개발일정 16](#_Toc184855737)

[5. 개발환경 19](#_Toc184855738)

# K-평균 클러스터링(K-means clustering)

## 문제 인식과 준비

텍스트, 영수증, 폰트, 화이트이(가) 표시된 사진

자동 생성된 설명K-Means는 데이터를 K개의 클러스터로 그룹화하는 비지도 학습 알고리즘입니다. 알고리즘은 반복적인 계산을 통해 클러스터 중심을 업데이트하여 최적의 분할을 찾습니다.

2023년 자료의 육지면적, 탐방객수를 가져와 NLPRK\_STA.csv를 만들었고 출력에 필요한 폰트 malgunsl.ttf를 준비했습니다. 풀이는 깃허브를 참고하였습니다.

## 해결 과정 및 코드 구현

1. 초기화 - 클러스터 개수 k와 반복 횟수 설정, 랜덤으로 초기 중심 선택

2. 거리 계산 - 각 데이터 포인트와 클러스터 중심 간의 거리 계산. 유클리드 거리 계산을 사용

3. 클러스터 할당 - 가장 가까운 중심에 데이터 포인트를 할당

4. 중심 업데이트: 클러스터 내 데이터 포인트의 평균 좌표로 중심 업데이트

5. 중심 변화가 없을 때까지 또는 최대 반복 횟수까지 2~4단계를 반복

### 초기 클러스터 중심 설정

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 랜덤하게 선택된 데이터 포인트를 초기 중심으로 설정합니다. k만큼 반복합니다.
2. cluster\_centers는 각 클러스터의 중심 좌표를 저장합니다.

### 반복적인 클러스터링 수행

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 유클리드 거리 계산을 통해 데이터 포인트와 모든 클러스터 중심 간의 거리를 계산합니다.  
2. 거리 값은 distances\_to\_centers 배열에 저장됩니다.  
3. 각 데이터 포인트가 가장 가까운 중심(최소 거리)을 기준으로 클러스터에 할당됩니다  
4. 결과는 cluster\_assignments 배열에 저장됩니다

### 반복적인 클러스터링 수행

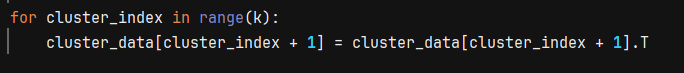


클러스터 데이터를 저장할 공간을 초기화합니다. 클러스터에 속할 데이터는 2개의 특성(예: 육지면적, 탐방객수)을 가지므로 빈 2차원 배열(열은 0개)을 생성합니다.

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

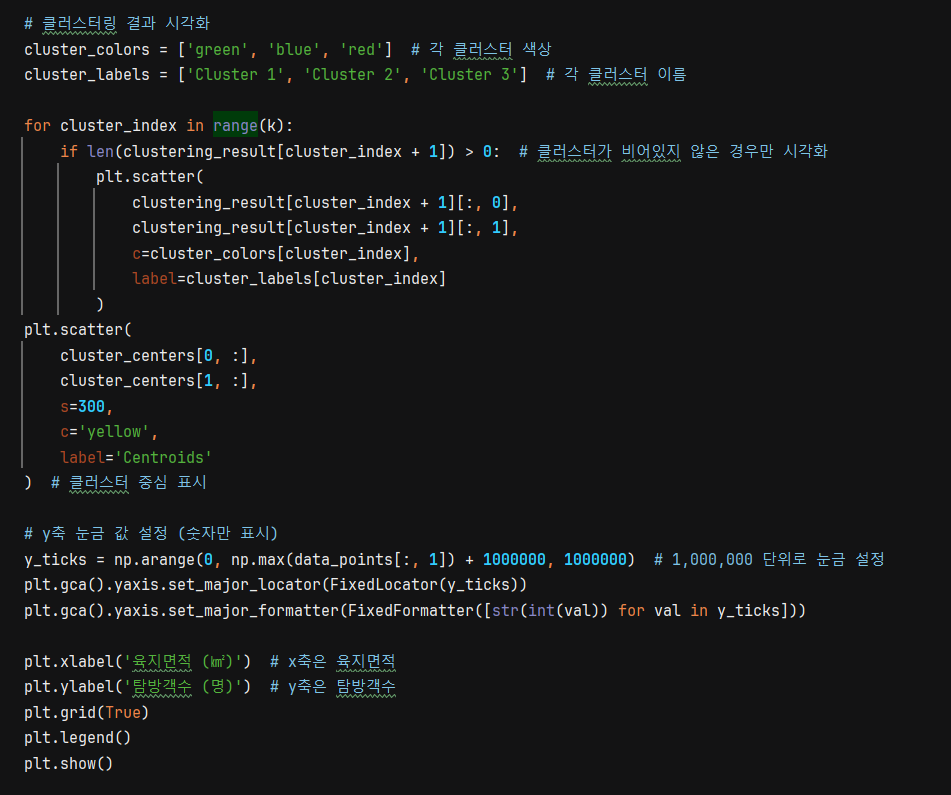
유클리드 거리를 기준으로 데이터 포인트를 클러스터에 그룹화하기 위해 각 데이터 포인트를 해당 클러스터에 추가합니다.



클러스터별 평균 좌표 계산을 쉽게 하기 위해 데이터 형식을 변환하기 위해 클러스터 데이터를 행(row) 기반으로 변환합니다.  

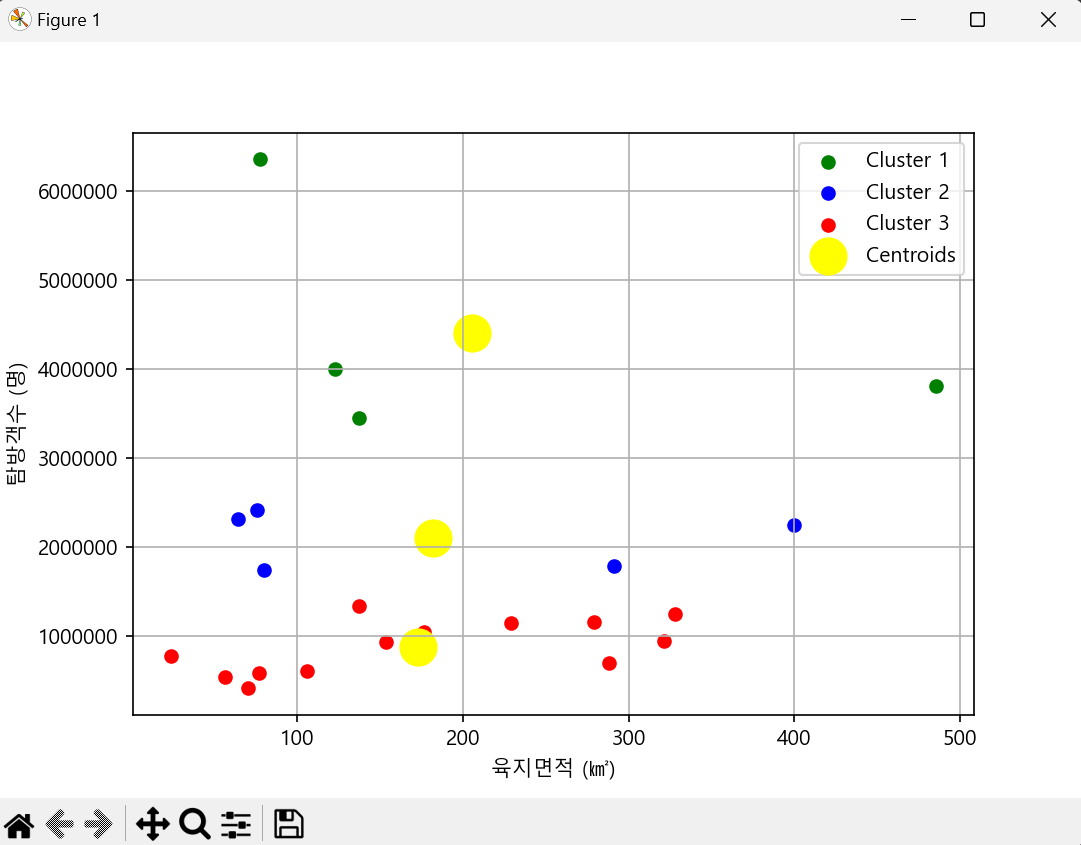

클러스터링이 점진적으로 수렴하도록 중심을 업데이트하기 위해 각 클러스터의 새로운 중심 좌표를 계산합니다.

### 출력 코드



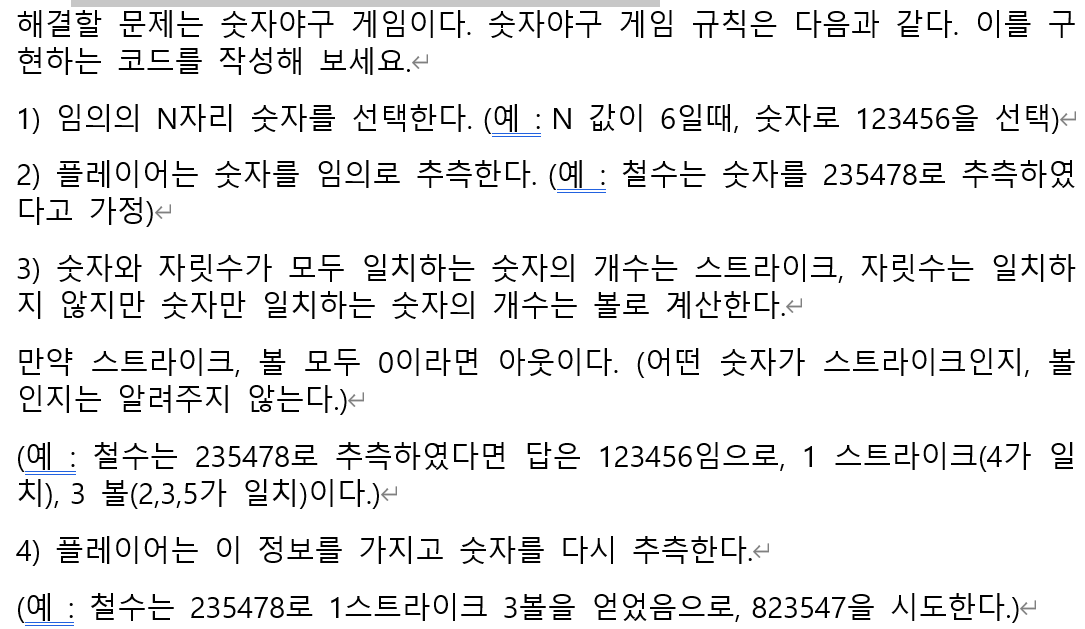
x축은 '육지면적', y축은 '탐방객수'로 설정하고 22개 국립공원을 mapping 한다.

## 결과



# 유전 알고리듬(숫자 추측하기)

## 문제 인식과 준비 과정



유전 알고리즘은 생물의 진화 원리를 모방하여 최적 해를 탐색하는 최적화 알고리즘이며 숫자 야구게임은 두 명의 플레이어가 숫자를 추측하며 승부를 겨루는 간단한 추리 게임입니다.

## 해결 과정 및 코드 구현

1. 정답 숫자와 초기 개체군을 생성한다.
2. 세대별로 적합도 평가, 선택, 교차, 돌연변이 수행한다.
3. 최적 해를 추적하며, 적합도가 최대치에 도달하면 종료한다.
4. 염색체 개수 N을 바꾸면서 비교 분석하기 위해 POPULATION\_SIZE를 점점 증가시키며 알고리즘 성능 비교한다.

설정값



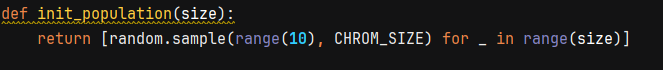
### 정답 숫자 생성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

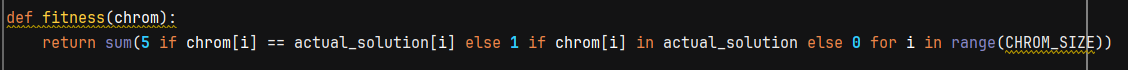
정답 숫자를 생성하기 위한 함수이며 CHROM\_SIZE는 현재 6으로 설정되어있습니다.

### 초기 개체군 생성



size크기만큼 초기 개체군(첫 번째 세대)을 생성합니다.

### 적합도 계산



스트라이크일 경우 5점, 볼일 경우 1점, 아니면 0점을 더합니다.

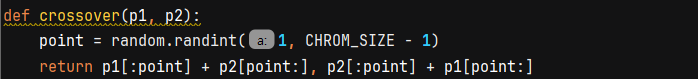
### 적합도 계산

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

유전 알고리즘에서 다음 세대를 생성하기 위해 부모 개체를 선택하도록 하기위해 적합도 높을수록 선택될 확률이 높게 설정합니다.

### 새로운 자손 후보 생성



1부터 CHROM\_SIZE - 1 사이의 교배 지점을 기준으로 부모의 유전자를 분할·교환하여 두 자손을 생성합니다.

### 돌연변이 생성

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

기존의 교배만으로 해결하지 못하는 경우를 대비하여 MUT\_RATE 확률로 일부 유전자를 돌연변이 시킵니다

### Main

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명초기 개체군을 생성하고, 유전 알고리즘의 선택(select), 교배(crossover), 돌연변이(mutate)를 반복하여 최적해를 찾으며 개체군 크기를 10씩 증가시켜 총 8번 실험합니다.

## 결과

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

# A\* 알고리즘

## 문제 인식과 준비 과정

도표, 스크린샷, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

A\* 알고리즘은 그래프 탐색 알고리즘으로, 시작 지점에서 목표 지점까지 최단 경로를 찾습니다. 이 알고리즘은 G값(현재까지의 이동 비용)과 H값(목표까지의 추정 비용)을 더한 F 값을 기준으로 최적의 경로를 탐색합니다.

## 해결 과정 및 코드 구현

상하좌우를 이동할 때 비용은 10, 대각선은 14로 계산한다. 대각선 이동 비용은 유클리드 거리 (약 1.414)를 반영하여 상하좌우 이동 비용(10)의 약 1.414배인 14로 설정했습니다.

* g(n): 시작 노드에서 현재 노드까지의 실제 비용 (현재까지 이동한 거리).
* h(n): 현재 노드에서 목표 노드까지의 예상 비용 (휴리스틱, 예측값).
* f(n) = g(n) + h(n): 현재 노드에서 목표 노드까지의 총 비용 (우선순위 평가).

A\*는 f(n) 값이 가장 낮은 노드를 우선적으로 탐색합니다.

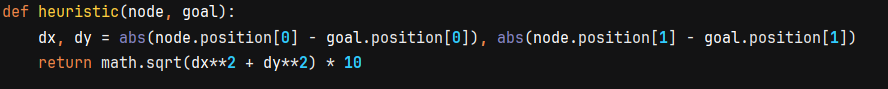
### Node

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

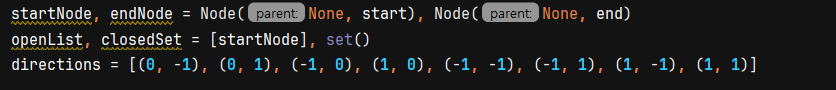
A\* 알고리즘에서 각 위치를 표현하는 노드를 정의하는 클래스입니다. 노드는 현재 위치, 비용(g, h, f), 그리고 부모 노드 정보를 포함합니다.

### 휴리스틱 함수



현재 노드에서 목표 노드까지의 직선 거리(유클리드 거리)를 계산하여 예상 비용으로 반환합니다.

### A\* 함수



시작과 끝 노드 초기화, 리스트 및 집합 초기화, 이동방향 (상하좌우+대각선) 정의

스크린샷, 텍스트, 폰트이(가) 표시된 사진

자동 생성된 설명

openList가 비어 있지 않으면 반복하며 openList에서 f값이 가장 낮은 노드를 currentNode로 설정

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

만약 currentNode가 목표 노드와 같으면 탐색 종료합니다. 현재 노드부터 시작노드까지 부모노드를 따라가며 경로를 추적하고 역순으로 저장된 경로를 뒤집어서 반환합니다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

방향 벡터를 통해 현재 노드의 위치에서 새로운 이웃 노드의 좌표를 계산하고, 이웃 노드가 이미 방문 완료된 노드 집합에 포함되어 있거나 유효한 범위를 벗어나거나 장애물인 경우 탐색에서 제외합니다.

g\_cost는 현재 노드에서 이웃 노드로 이동하는 데 드는 실제 비용으로, 상하좌우 이동 시 10, 대각선 이동 시 14로 설정하여 현실적인 거리 계산을 반영합니다.

현재 노드를 부모로 가지는 새 이웃 노드 객체를 생성하고, 만약 새 경로의 g\_cost가 기존 경로openNode보다 크거나 같다면 더 나은 경로가 아니므로 탐색에서 제외합니다.

유효한 이웃 노드는 총 비용(f = g + h)을 계산한 후, openList에 추가합니다.

## 결과

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

시작 (3,1) 끝 (4,7)

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

시작 (6,0) 끝 (4,6)

# 강화학습

## 문제 인식과 준비 과정

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

Q-Learning은 환경과의 반복적 상호작용을 통해 최적의 행동 가치를 학습하는 알고리즘입니다.

에이전트는 즉각적인 보상과 다음 상태의 최대 가치를 결합하여 Q-값을 업데이트하며, 시간이 지남에 따라 최적 정책으로 수렴하게 됩니다.

텍스트, 도표, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

에이전트는 시작 지점에서 목표 지점까지 이동하며, 장애물에 접촉하면 에피소드가 종료됩니다. 행동은 랜덤하게 선택되며, 보상은 목표 도달 시 +100, 장애물 접촉 시 -100, 일반 이동 시 0입니다. 텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명

현재 Q-값을 q\_1에 저장합니다.

q\_2에 벨만 방정식에 따른 새로운 Q-값 계산 - 현재 보상에 할인율을 곱한 다음 상태의 최대 Q-값을 더해 계산합니다.

self.q\_table[state][action] += self.step\_size \* (q\_2 - q\_1)을 통해 Q-값 업데이트합니다.

step\_size = 0.01: 학습률, Q-값이 얼마나 빠르게 업데이트되는지 결정합니다.

discount\_factor = 0.9: 할인율, 미래 보상의 중요도를 나타냅니다.

epsilon = 0.9: 탐험-활용 균형, 최적 행동을 선택할 확률을 나타냅니다.

q\_table = [0.0, 0.0, 0.0, 0.0]: 초기 Q-값은 모든 상태-행동 쌍에 대해 0으로 설정합니다.

## 해결 과정 및 코드 구현

Q-Learning 알고리즘을 사용해 에이전트가 주어진 환경에서 최적의 행동 정책을 학습합니다.

시작 지점에서 목표 지점까지 이동하며 장애물을 피하고, 각 셀에서 위, 아래, 왼쪽, 오른쪽 중 하나의 행동을 선택합니다. 행동 후 즉각적인 보상(목표 도달 시 +100, 장애물 접촉 시 -100, 그 외 0)을 받습니다.

벨만 방정식을 기반으로 Q-값을 업데이트합니다.

### Env 클래스

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

강화학습 환경을 정의하며, 상태, 행동, 보상, 목표 지점 등을 설정합니다. 동작 가능한 행동 정의 (u, d, l, r) 하고 출력될 캠버스를 정의합니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

캔버스를 그리는 함수와 이미지를 가져오는 함수 입니다.

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Q-테이블의 값을 전체 캔버스에 출력하여 학습 상태를 시각적으로 표시합니다. 각 셀마다 Q-값이 출력되며, 행동 방향(위, 아래, 왼쪽, 오른쪽)에 따라 위치가 다릅니다

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

coords\_to\_state : GUI 캔버스의 픽셀 좌표를 강화학습 알고리즘이 사용하는 상태 값으로 변환합니다.

state\_to\_coords : 강화학습 알고리즘이 반환한 상태 값을 GUI 캔버스에서 사용할 수 있는 픽셀 좌표로 변환합니다.

Reset : 에이전트의 위치를 시작점으로 초기화하여 학습 에피소드를 새로 시작합니다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

강화학습 환경에서 에이전트가 특정 행동(action)을 수행한 후의 상태, 보상, 종료 여부를 반환합니다.

가능한 행동(위, 아래, 왼쪽, 오른쪽)을 이동량으로 정의하고 현재 에이전트의 캔버스 좌표(current)에서 행동에 따라 다음 좌표(next\_coords)를 계산합니다.

next\_coords가 캔버스 경계 내에 있는지 확인하고 에이전트 이동, 출력되게 합니다.

에이전트가 파란 원의 좌표에 도달했을 때

* 상태 [2, 2] 반환
* 보상 100
* 종료 여부 True

에이전트가 연두색 삼각형 좌표에 도달했을 때

* 상태 [1, 1] 반환
* 보상 -10
* 종료 여부 True

에이전트가 경계 내에서 유효한 이동을 했지만 목표 또는 장애물에 도달하지 않은 경우:

* 새로운 상태는 next\_coords를 상태 값으로 변환한 [row, col].
* 보상 0.
* 종료 여부 False

### QLearning 클래스

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

주어진 환경(Env)에서 학습하며, 목표 지점에 도달하거나 장애물을 피하는 최적의 행동 정책을 학습하는 과정을 보여줍니다. 에이전트가 수행할 수 있는 모든 행동의 리스트와 Q-테이블을 초기화하여, 모든 상태-행동 쌍의 Q-값을 저장합니다.

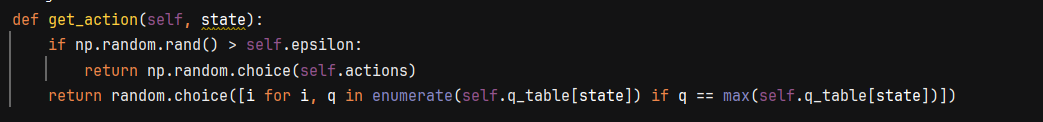
텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

현재 Q-값 계산: 현재 상태-행동 쌍의 Q-값을 가져옵니다.

새로운 Q-값 계산: 벨만 방정식을 사용해 새로운 Q-값을 계산합니다.

Q-값 업데이트: 학습률과 새로운 Q-값을 사용해 현재 Q-값을 업데이트합니다.



np.random.rand()로 0과 1 사이의 난수를 생성하여 탐험과 활용을 결정합니다.

np.random.rand()가 self.epsilon보다 작을 경우, 현재 상태의 Q-값을 참조하여 최적 행동을 선택합니다. 클 경우에는 랜덤 행동(탐험)을 선택합니다

### Env 클래스

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

환경을 구성하고 총 50개의 에피소드를 반복하며 학습을 진행합니다. 에이전트가 선택한 행동을 환경에서 실행하고 벨만 방정식을 사용해 Q-값을 업데이트합니다.

Q-테이블의 값을 GUI 캔버스에 표시하여 학습 상태를 시각적으로 확인할 수 있도록 했습니다.

done이 True(목표 도달 또는 장애물 접촉)라면 루프를 종료하고 다음 에피소드로 이동합니다

## 결과

텍스트, 스크린샷, 도표이(가) 표시된 사진

자동 생성된 설명