AI HW 보고서

전공: 컴퓨터공학과 학년: 4 학년 학번: 20202106 이름: OSHIMA ASUKA

1. 구현 방법

- Def compute_transition_matrix(model):
  각 state (r,c)에서 (r′,c′)에 이동하는 확률을 계산하는 함수다. 모둔 셀에 대해 가능한 action 를 실행한 경우 이동하는 셀 (r′,c′)을 결정한다. 이동할 셀이 벽아니면 그리드 밖으로 나가는 경우는 이동을 못하기 때문에 이동시키지 않는다. 즉 이동 가능하는 지를 확인 후 이동시킨다. Model.D 를 이용해서 Model.D 에서 얻은 확률을 P 의 요소에 정하고 transition probability 을 업데이트한다.
- def update_utility(model, P, U_current):
  가능한 discounter rewards 의 기대 값의 합을 가장 크게 만들도록 utility 를 업데이트하는 함수다. 먼저 U_next 배열을 0 으로 초기화한다. 루프를 이용해서 모둔 셀에 대하여 처리를 한다. Terminal state 의 셀에서는 utility 는 model.R[r, c]와 같으며, 더 이상 업데이트할 필요가 없다. Terminal state 가 아닌 셀에서는 현재의 reward 에 discounter reward 를 더한다. 다음은 최대 가대 값의 utility 를 계산한다. 각 액션에 대하여 이동 후 상태 (r′,c′) transition probability 와 현재 utility 의 곱의 합을 계산하여 이중에서 최대값을 선택한다.
- def value_iteration(model):
  그리드를 얻어 배열 U 를 초기화한다. P = compute_transition_matrix(model) 를 실행해 transition probability 행렬을 얻고 할인율은 MDP model 에서 얻는다. update_utility 함수를 사용하여 유틸리티 값을 갱신하고, 갱신된 유틸리티 값과 이전 유틸리티 값 사이의 최대 변화량 delta 가 작은 임계값 epsilon 보다 작아질 때까지 반복한다.

2. 실행 결과 후 저장된 사진 첨부

-0.040	-0.040	-0.040	1.000
-0.040	x	-0.040	-1.000
-0.040	-0.040	-0.040	-0.040

U_next.png

0.812	0.868	0.918	1.000
0.762	x	0.660	-1.000
0.705	0.655	0.611	0.387

result.png