Reinforcement Learning

Contents

Supervised Learning

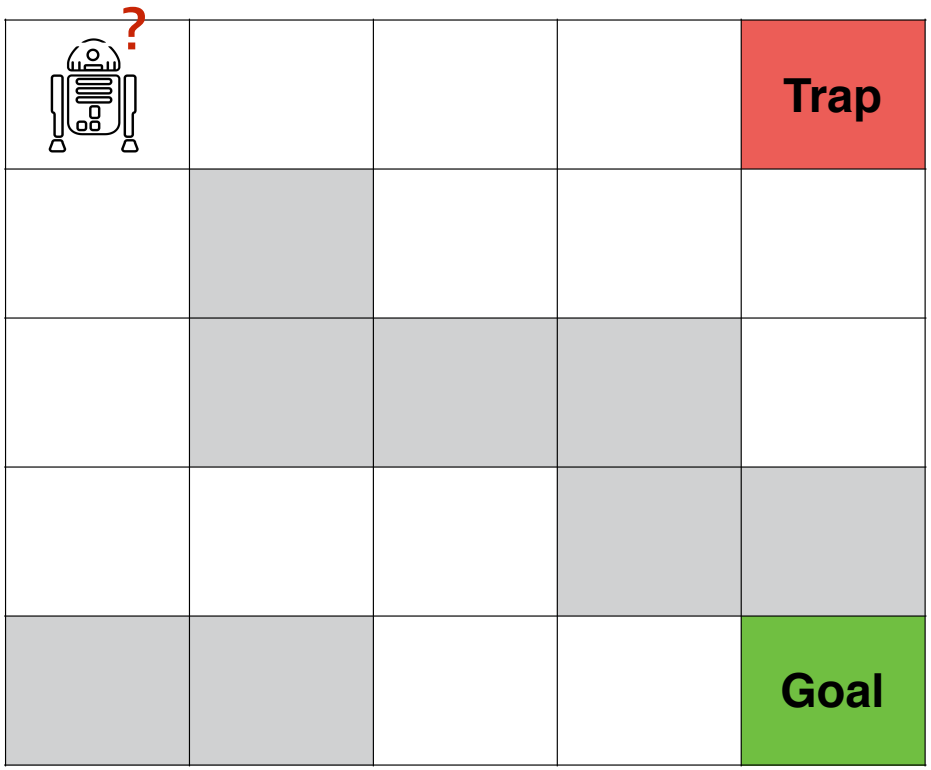
To be used when, "I know how to classify this data, I just need you(the classifier) to sort it."

Point of method: To class labels or to produce real numbers

Unsupervised Learning

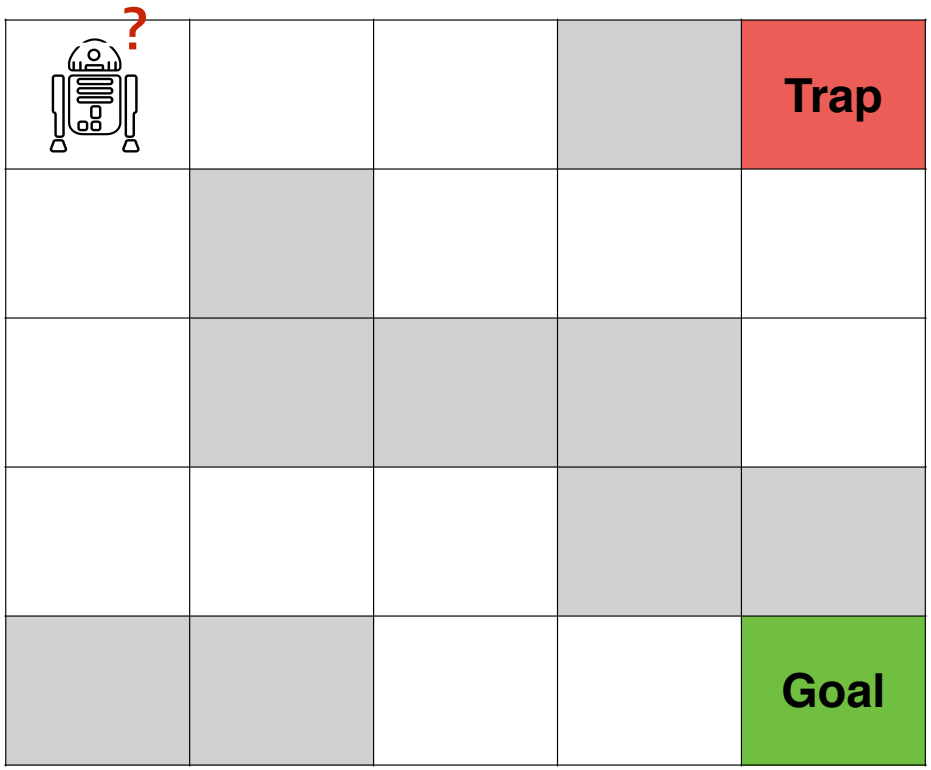
To be used when, "I have no idea how to classify this data, can you(the algorithm) create a classifier for me?"

Point of method: To class labels or to observe



Supervised Learning ?

Unsupervised Learning ?



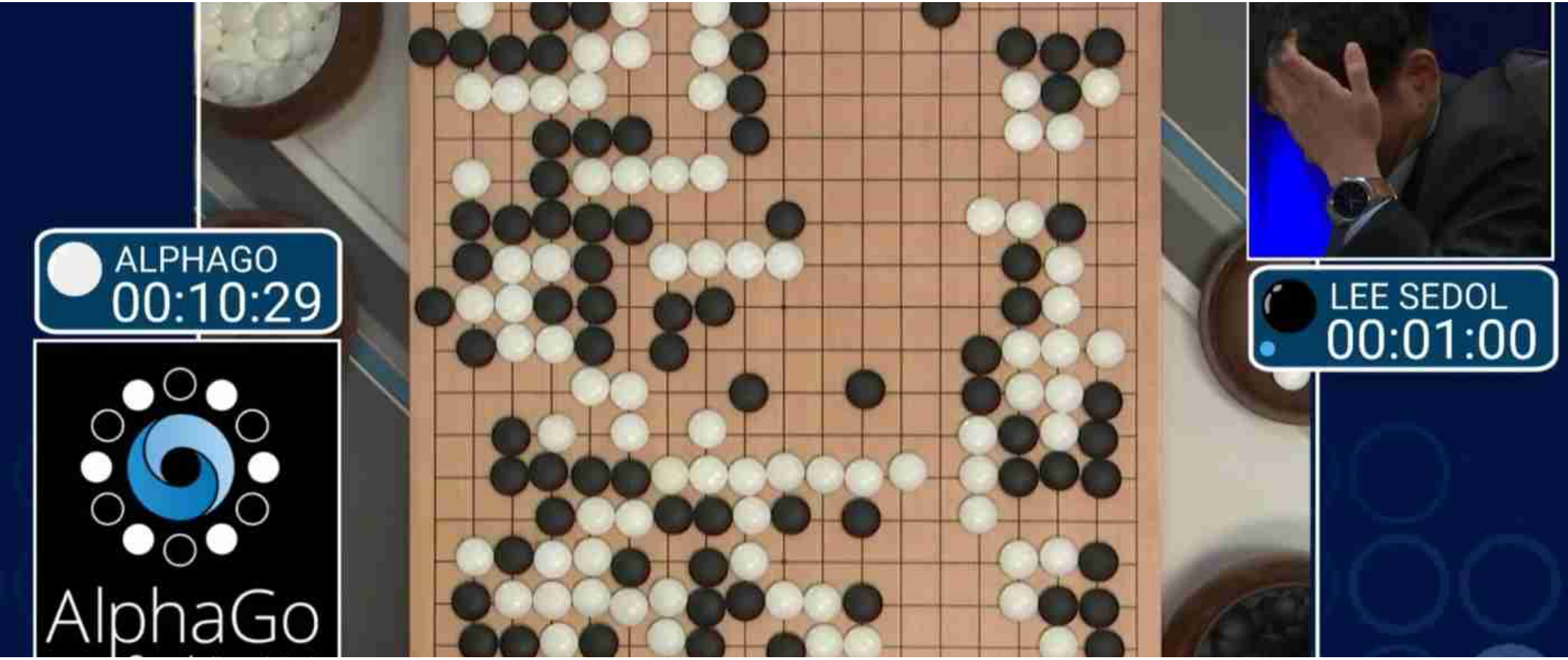
이 문제를 어떻게 풀 것인가?

모든 경로를 탐색해서 제일 짧은 길을 찾는다…?

미로가 너무 **크다면**
함정이 **많다면?**
시간에 따라 함정이 **랜덤하게** 발생한다면?

↓

모든 경로 탐색은 의미가 없다.



Supervised Learning ?

Unsupervised Learning ?

예를 들어, AlphaGo에 대해 가장 많이 받은 질문

모든 경우의 수를 학습해서 두는 것이냐?

바둑판

(가로 19줄) X (세로 19줄)

(19 X 19) ! 의 경우의 수를 가진다.

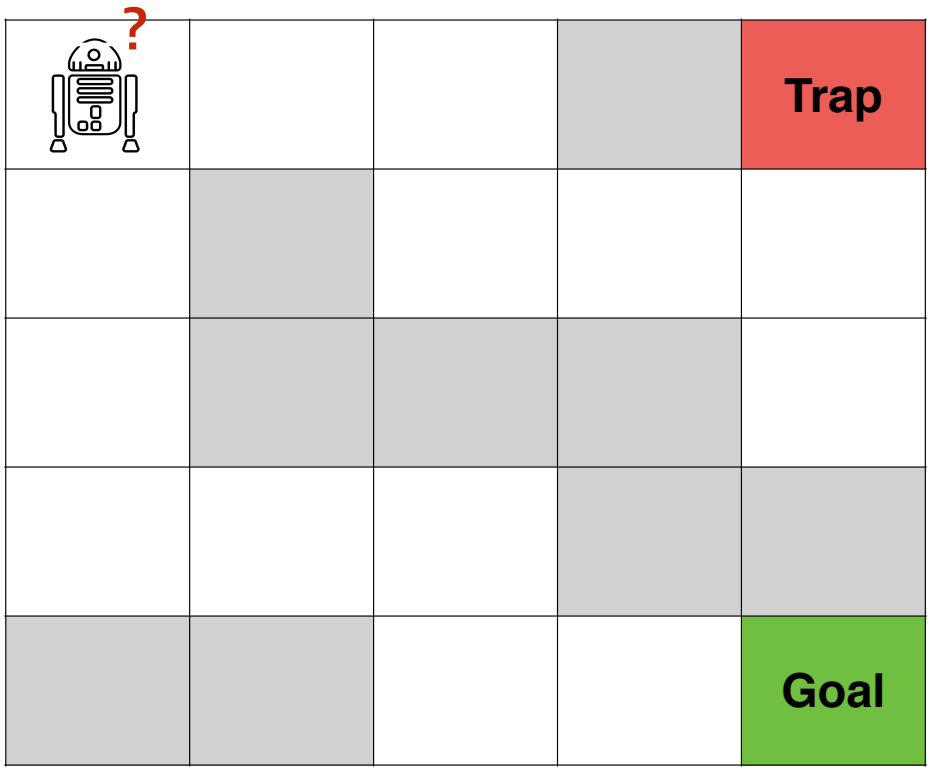
$1.4379232... \times 10^{768}$

우주의 나이 137.98억년 (2013년 발표) = 7,252,228,800,000,000초

Google DeepMind's Deep Q-learning playing Atari Breakout

Reinforcement Learning

To be used when, "I have no idea how to classify this data, can you classify this data and I'll give you a reward if it's correct or I'll punish you if it's not."



각 위치에서 어느방향으로 가야 하는가?

어떤 움직임이

Goal로 가는 움직임인지 분류한다.

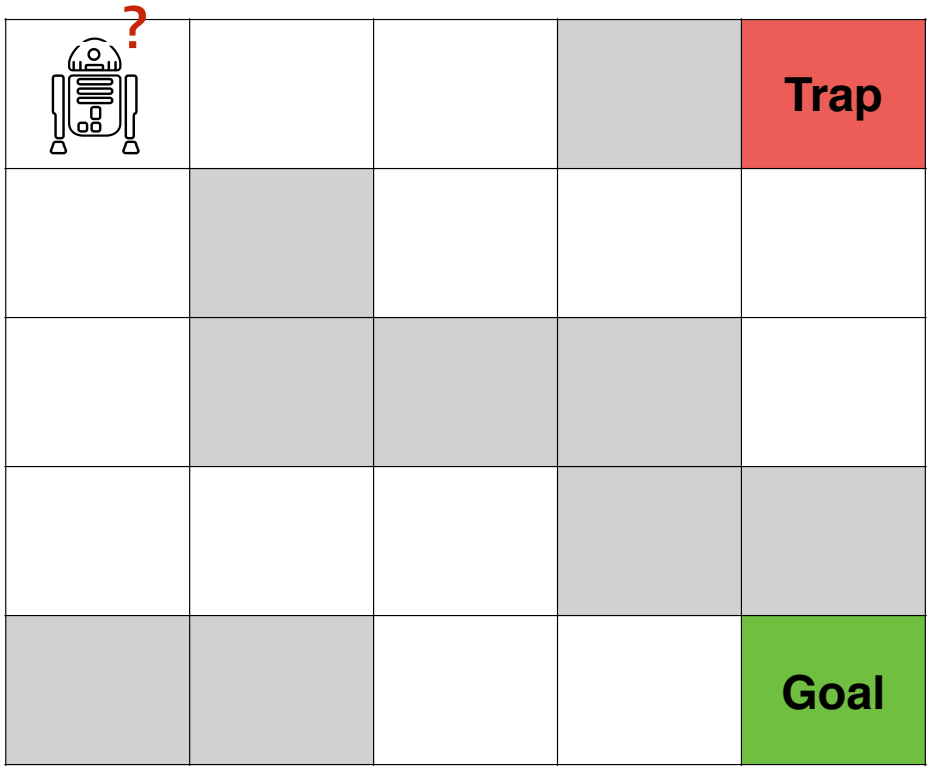
Goal에 도착했을 때는 **Reward**를,

Trap에 도착했을 때는 **Punish**를

이를 학습하여 **최단거리**로

미로를 탈출하는 경로를 분류해보자.

어떻게 설계를 할 것인가?



상태(S, state) : 미로의 각 위치

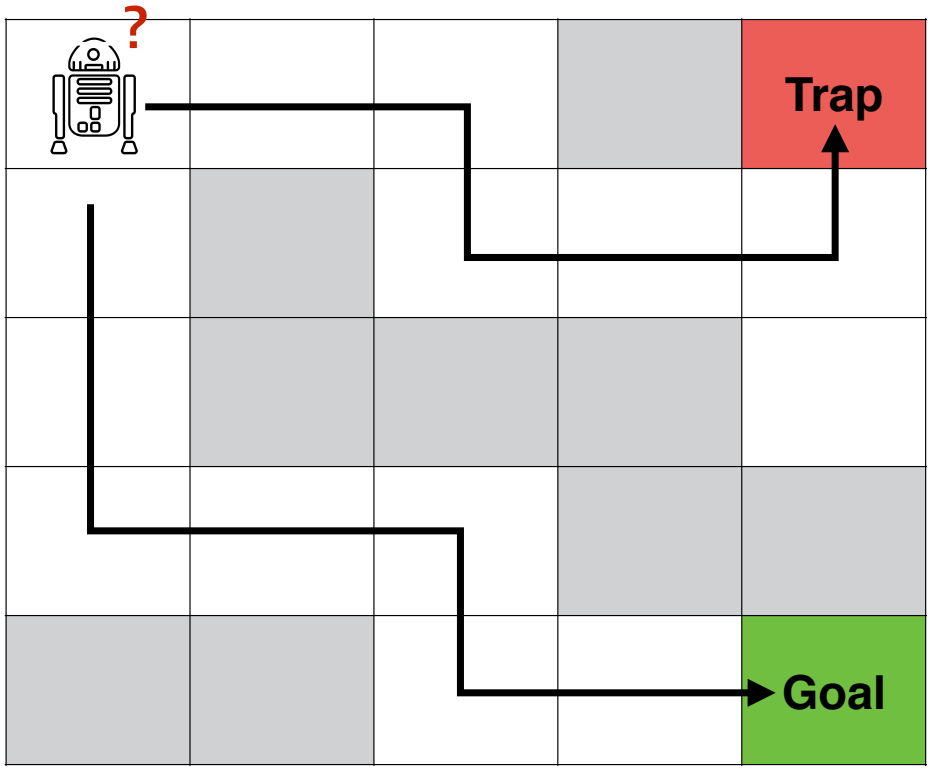
행동(A, action) : 상,하,좌,우 로 이동

보상(R, Reward) : Goal에 도착하면 +100

처벌 : Trap에 도착하면 -100

어떻게 설계를 할 것인가?

모든 경로에 대해 찾고 학습시킨다.



Breadth First Search Algorithm

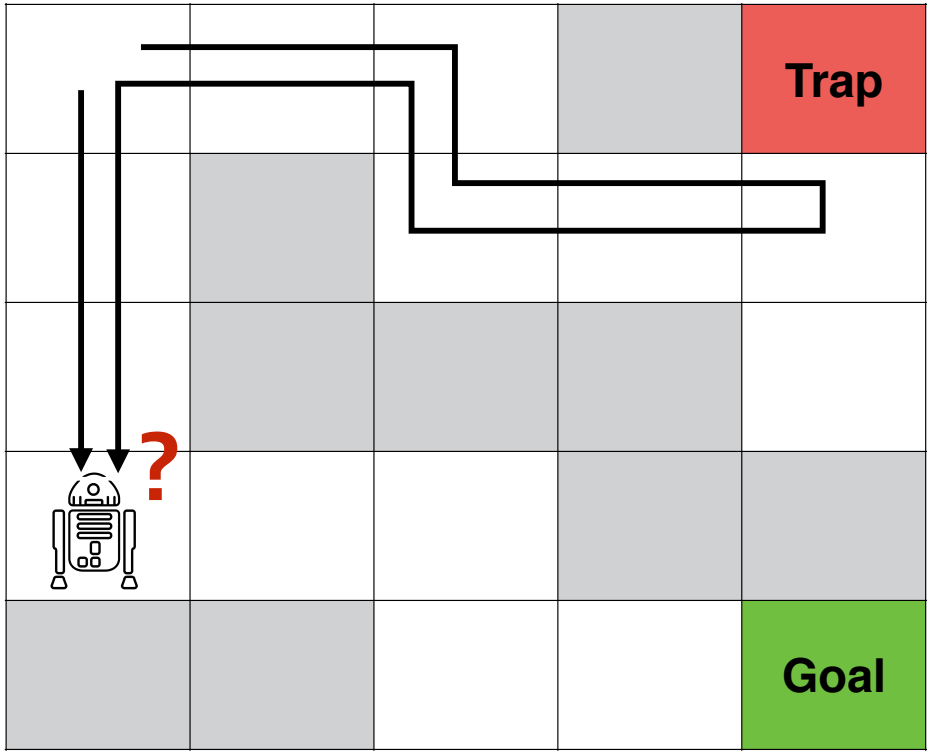
- 훌륭한 길찾기 알고리즘 중 하나.
- **경로상 존재**하는 모든 경우의 수를 탐색

미로에 대한 ***"Global Information"***이 필요하다.

↓ **바둑**은?

특수한 상황에서만 적용할 수 있는 해법

더 **일반적**으로 생각해보자.



미로에서 한 위치에 도달한 과거의 정보는

미래에 영향을 끼치지 않는다.

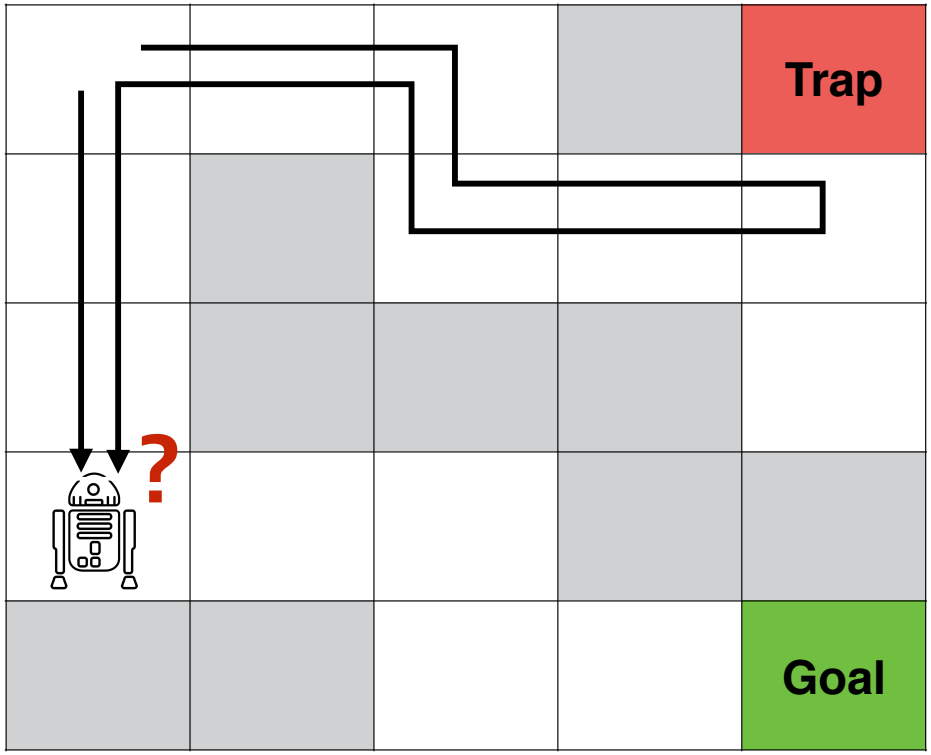
그렇다면,

과거의 경로와 상관없이, **현재의 위치를 기준**으로

최적의 경로를 탐색하면 되지 않을까?

마르코프 성질(Markov Property)

과거와 현재 상태가 주어졌을 때의 미래 상태의 **조건부 확률 분포**가
과거 상태와는 독립적으로 현재 상태에 의해서만 결정된다는 것



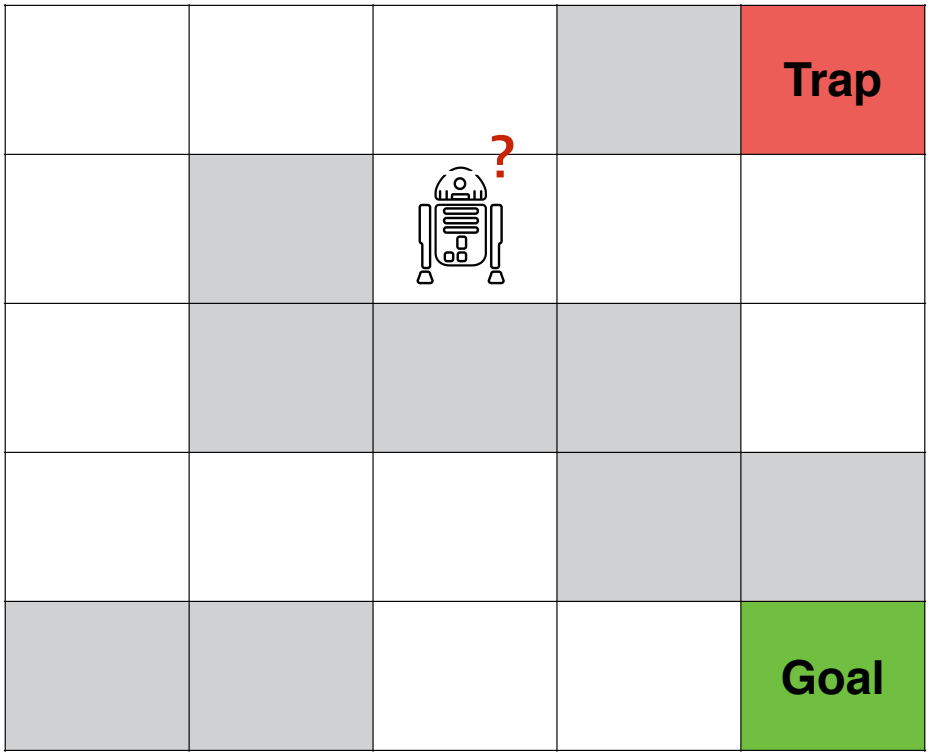
한 위치에 도달했을 때,

과거의 이력은 중요하지 않고,

그 순간 어디로 나갈 것인가만 보면 된다.

Local Information을 통해
목적지로 가는 최적의 방향을 선택한다면?

우리가 해야 할 일은?



매순간 최선을 다한 행동에 대한 평가를 하고,
이를 학습해서 좋은 행동은 더 많이.
나쁜 행동은 적게 해야한다.

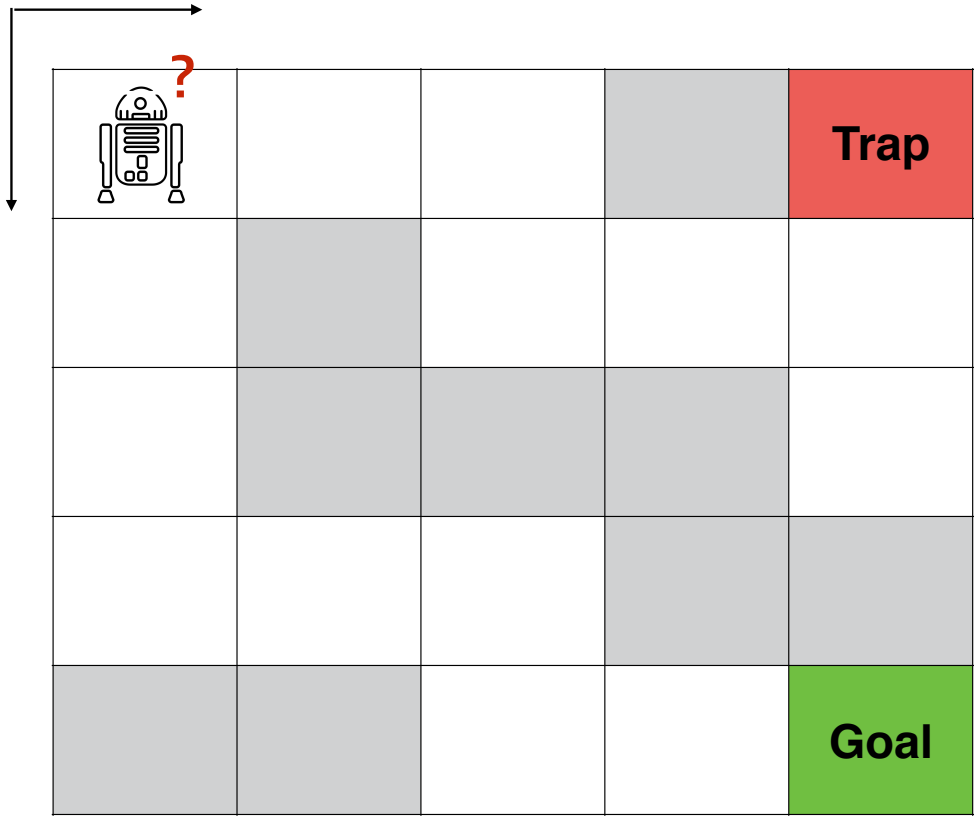
매 순간 action에 대한 reward의 합을 통해 최종적인 reward의 합이 **maximize**되게 하는 것

어떻게, 무엇을 학습할 것인가?

Monte Carlo Policy Evaluation

각 state에 대한 policy를 설정하고, state에 대한 ***policy***의 가치를 학습하자.

state s에서 어떤 action a를 수행할 것인가에 대한 **정책 π**

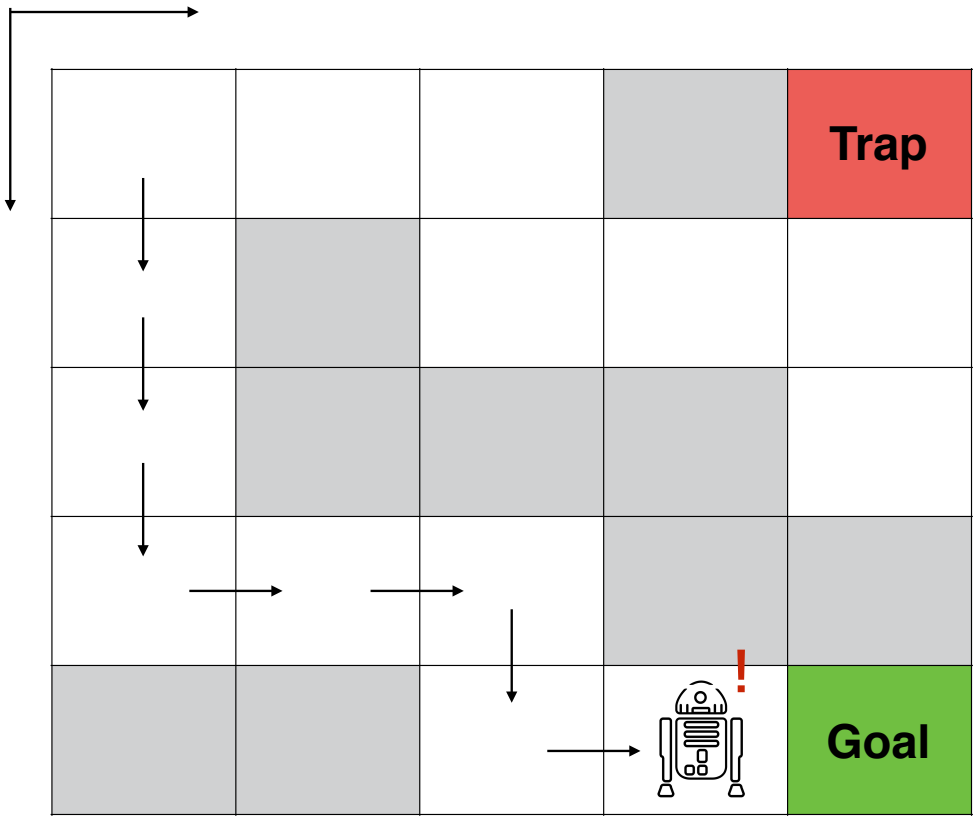


(1,1)번째 위치에서의 가능한 policy는
(R(1,2), D(2,1))

(1,2)번째 위치에서의 가능한 policy는
(L(1,1), R(1,3))

(1,3)번째 위치에서의 가능한 policy는
(L(1,2), D(2,3))

Policy의 가치를 어떻게 학습할 것인가?



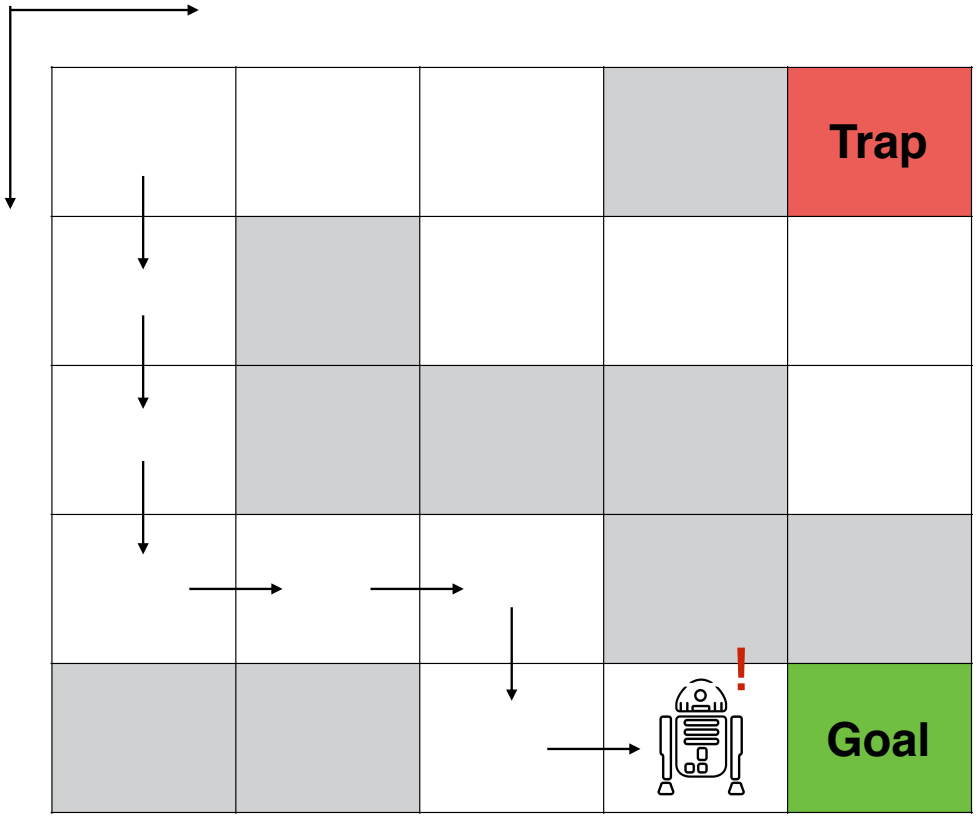
목적지에 도달 했을 때,
그때의 모든 경로에 대해 평가하자!

(1,1) → (2,1) → (3,1) → (4,1) → (4,2)
→ (4,3) → (5,3) → (5,4) → (5,5)

목적지 주변일수록 중요하다.

획득한 reward를
일정비율(discount factor, γ)로 감소시키며,
각 위치에서의 policy에 대해 학습시킨다.

최단경로를 학습하려면?



최단경로로 도착한 경우
가장 큰 Reward를 주자.

Goal / Trap에 도달한 경우를 제외하고
모든 상황에서 -1의 Reward를 제공하자.

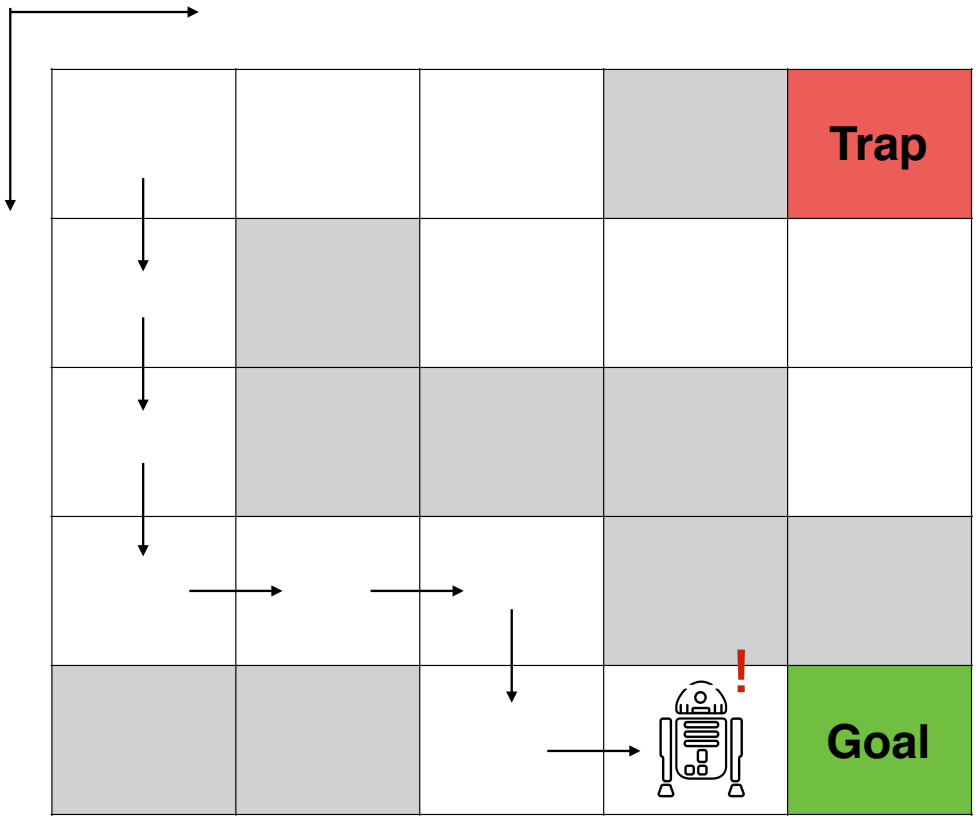
예시에서 최단 경로로 도착한 경우,

$$R = 100 - 8$$

예시에서 최단 경로가 아닌 경로로 도착한 경우,

$$R = 100 - 8 + x$$

Reinforcement Learning에서 **적절한 Reward**를 설정하는 것은 매우 중요한 문제다.



(1,1) → (2,1) → (3,1) → (4,1) → (4,2)
→ (4,3) → (5,3) → (5,4) → (5,5)

(5,4)/R= (5,4)/R + (100-8)*γ = (100-8)*0.7

(5,3)/R = (5,3)/R + (100-8)*γ^2 = …

(4,3)/D = (4,3)/D + (100-8)*γ^3 = …

(4,2)/R = (4,2)/R + (100-8)*γ^4

각 state에 해당하는 policy가
1 혹은 특정값에 수렴할때까지 반복

Value Table

State	Up	Down	Left	Right
(1,1)	0	90	0	10
(1,2)	0	100	0	0
(1,3)	10	90	0	0
…	…	…	…	…

Policy Table

State	Up	Down	Left	Right
(1,1)	0	9/10	0	1/10
(1,2)	0	1	0	0
(1,3)	1/10	9/10	0	0
…	…	…	…	…

Policy Table을 기반으로 모든 state에서
최적의 Policy를 따라가면 최적 경로를 발견할 수 있다.

Pseudo Code for Monte Carlo Policy Evaluation

MC_EVALUTAION(MDP, π, γ)

Inputs	policy π
	discount factor γ
Output	value function V^π
Initialize	$V = 0$

repeat

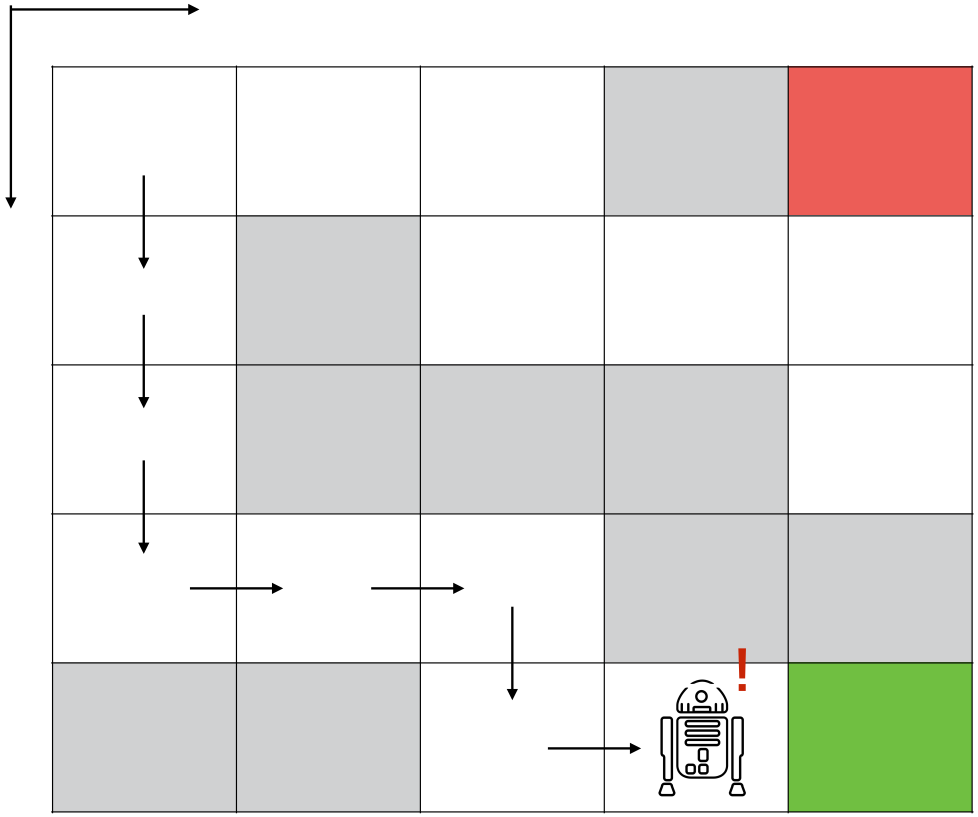
1. initialize s, τ, ρ
2. while $s \notin T$ do
   - **(a)** let $\tau = \tau \cup \{s\}$
   - **(b)** take action $a = \pi(s)$
   - **(c)** observe reward r and next state s'
   - **(d)** for all $s \in \tau$, let $\rho(s) = \rho(s) + r$
   - **(e)** let $s = s'$
3. for all $s \in \tau$, $V(s) = \gamma \cdot \rho(s)$

forever

Monte Carlo Policy Evaluation with Python

Q-Learning

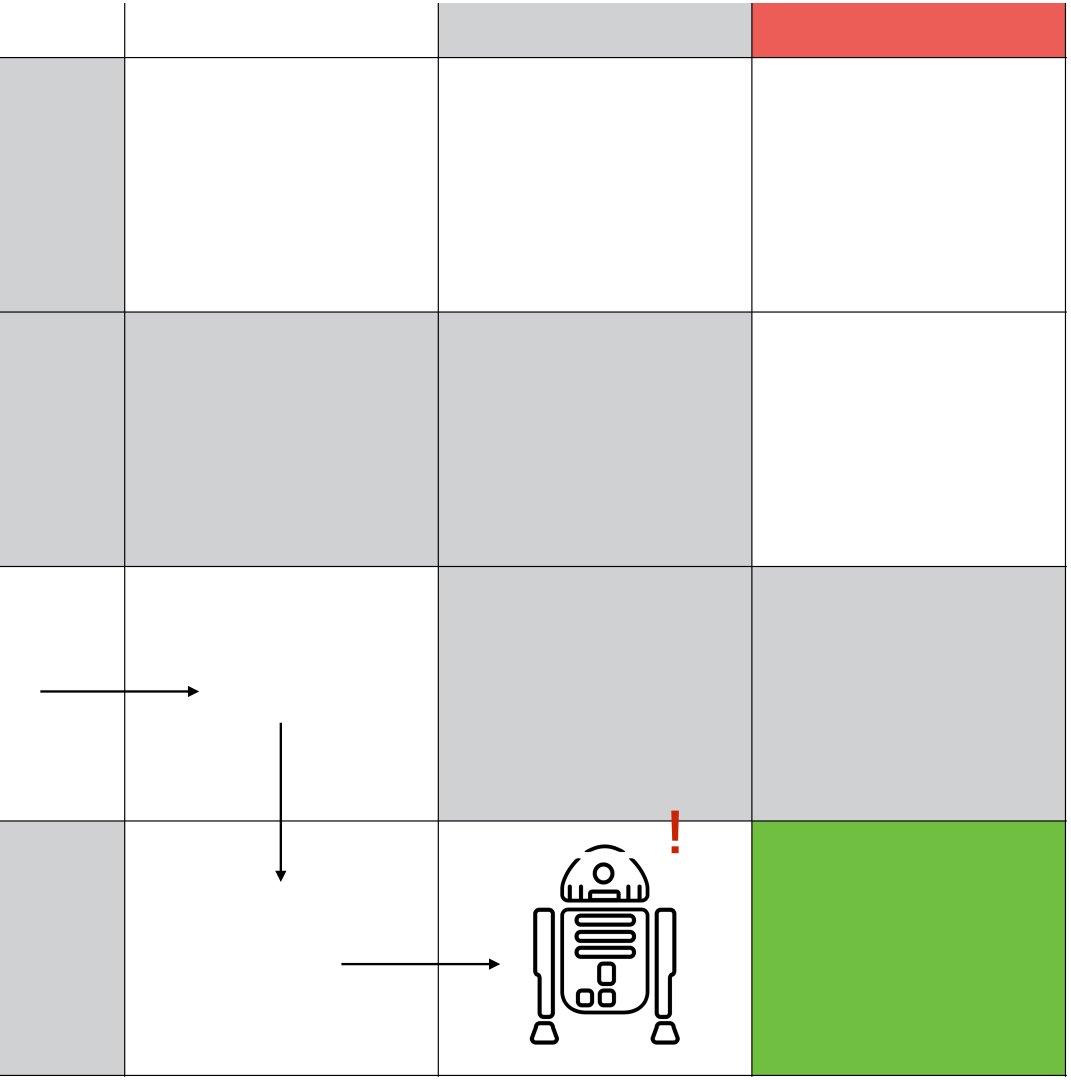
policy에 의한 결과인 action까지 학습해버리면 어떨까?



(1,1) : [0, **9/10**, 0, 1/10]

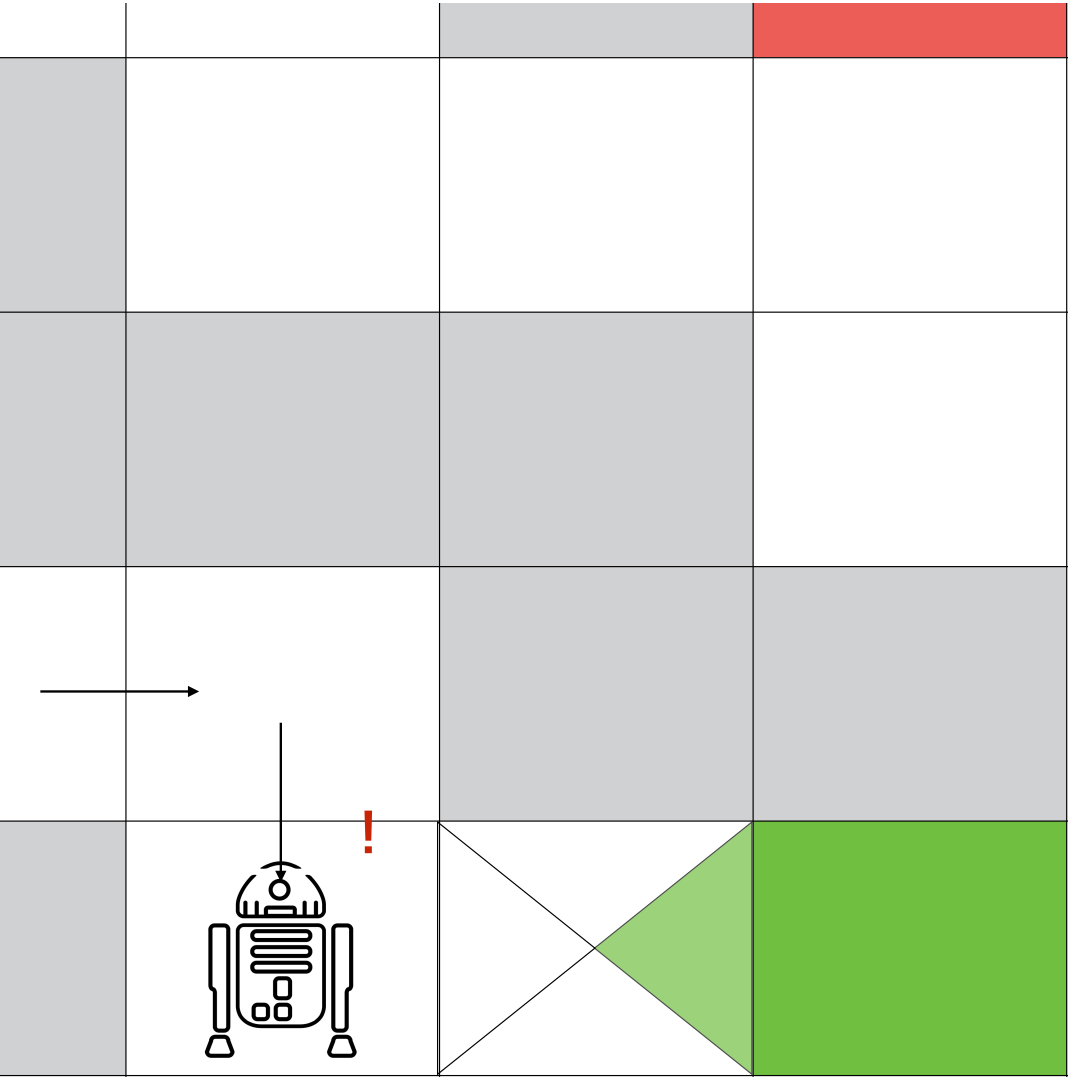
(1,1) : Down!

후보 state 중 Reward가 존재하는 경우,
후보 state와 해당 state의 action까지 학습한다.



현재의 state에서 최대의 Reward를 갖는
다음 state와 action을 학습하자.

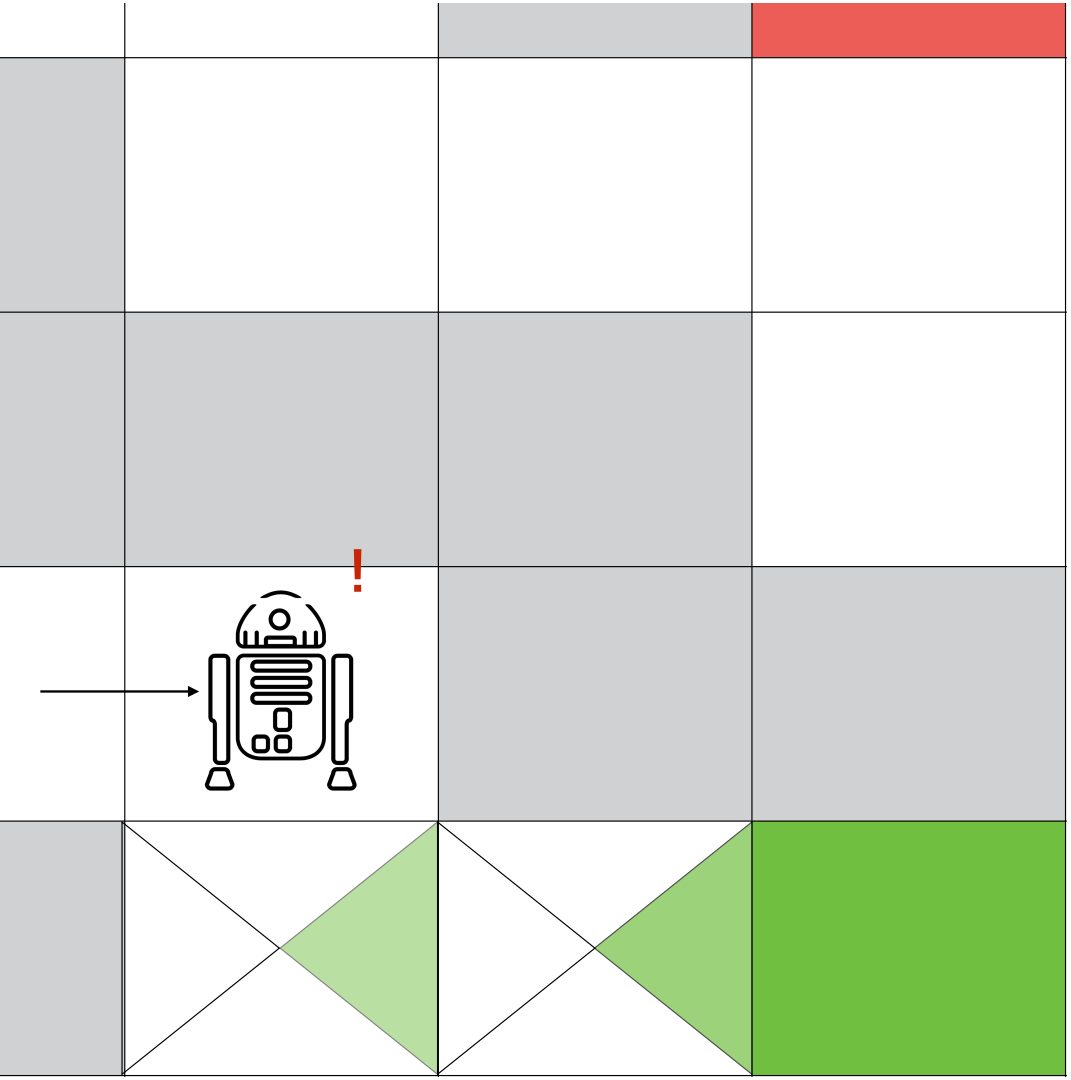
(5,4)에서는 Right가 최적의 움직임이다.



(5,3)에서 가능한 state를 살펴보니,

(4,3)는 Reward가 없는데,
(5,4)는 Right로 Reward가 있다.

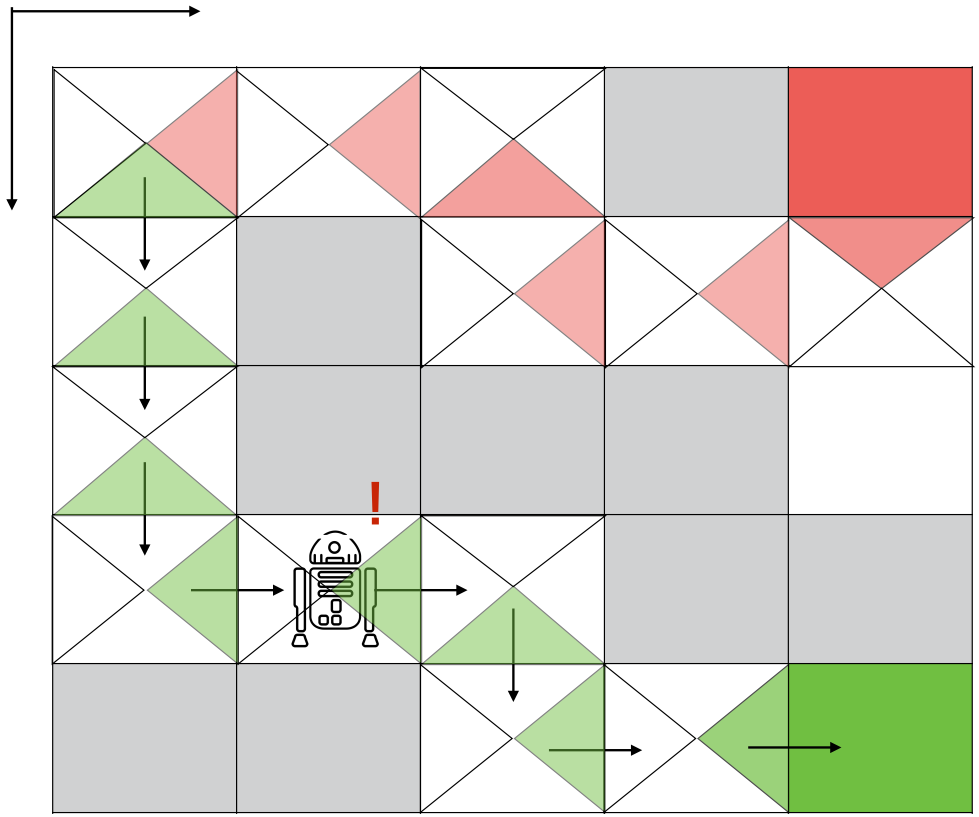
(5,4), Right가 다음 움직임이고,
지금은 (5,3)에서 Right로 가자.



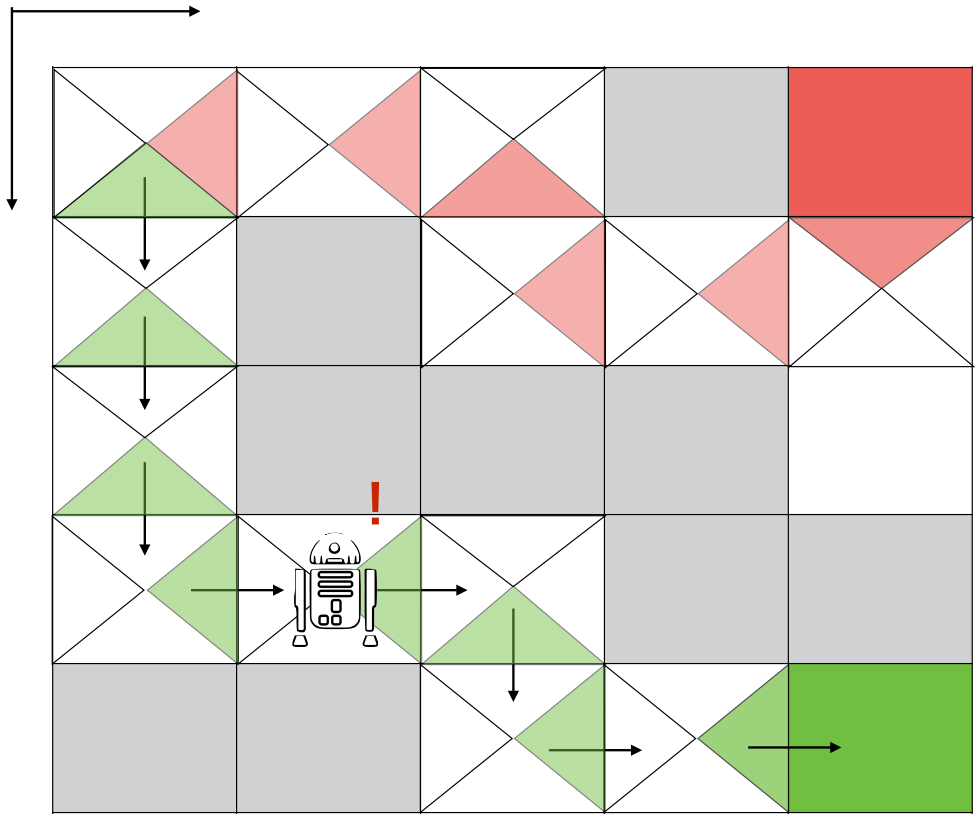
(4,3)에서 가능한 state를 살펴보니,

(4,2)는 Reward가 없는데,
(5,3)는 Right로 Reward가 있다.

(5,3)의 Right가 다음 움직임이고,
지금은 (4,3)에서 Down으로 가자.



Trajectory	Q-Learning
(4,5) →(5,5)	Q((4,5),L) = 100+0.7max(Q((5,5),a)) = 100
(3,5) →(4,5) →(5,5)	Q((3,5),L)=0+0.7max(Q((4,5),a))=70
(3,4) →(3,5) →(4,5) →(5,5)	Q((3,4),D) = 0 + 0.7max(Q((3,5),a))=49
(2,4) → (3,4) →(3,5) →(4,5) →(5,5)	Q((2,4),L) = 0 + 0.7max(Q((3,4),a))=34.3



이와 같은 학습을 반복하면,
최종적으로는 각 state에서
최적화된 action까지 학습

결국
(1, 1)에서 (5, 5)까지
최대 Reward를 갖는 state와
action을 그대로 따라간다.

Bellman's optimality equations for Q:

$$Q^*(s_t, a_t) = R(s_t, a_t) + \gamma E\left[\max_a Q^*(s_{t+1}, a)\right]$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]$$

s_t: state at time t

a_t : action at t

Pseudo Code for Q-learning

Q_LEARNING(MDP, π, ϵ)

Inputs	discount factor γ
	rate of exploration ϵ
Output	action-value function Q^*
Initialize	$Q = 0$

1. initialize s, a
2. while $s \notin T$ do
   - **(a)** take action a
   - **(b)** observe reward r and next state s'
   - **(c)** $Q_{(s,a)} = Q_{(s,a)} + [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
   - **(d)** choose action a'
   - **(e)** $s = s', a = a'$

Q_LEARNING(MDP, π, ϵ)

Inputs	discount factor γ
	rate of exploration ϵ
Output	action-value function Q^*
Initialize	$Q = 0$

Q-learning을 통해 학습을 하면,
현재 state를 기준으로
다음 state와 action이 함께 결정된다.

ε의 확률로 결정된 state,action외에 다른 state를 찾는다.

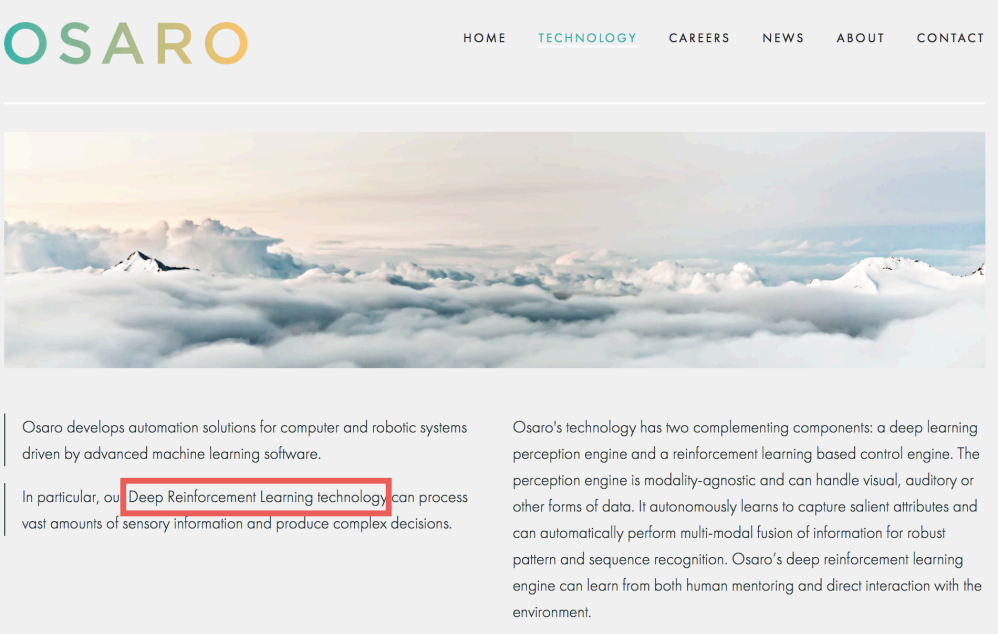
1-ε의 확률로 결정된 state,action를 실행한다.

“Local Minimum”에 빠지는 것을 방지하기 위함,
다른 경로를 학습하기 위함.

Q-learning with Python

Application of Reinforcement Learning

1. OSARO



- 산업용 로봇을 위한 기계 학습 소프트웨어 개발사
- 실리콘벨리에서 330만 달러의 자금 조달 성공
- Q-learning을 사용한 제품 개발 진행 중
- 강화학습을 이용, 완벽한 작동을 위한 학습을 통해 확장성을 강화

Application of Reinforcement Learning

2. Preferred Networks

- 일본 도쿄의 AI 스타트업, 도요타와 자율주행 기술 개발 중
- 자동차 속도와 방향의 변화, 각종 센서의 출력데이터를 이용한 시뮬레이션을 개발
  - 시뮬레이터에 학습 방법으로 강화학습을 채택
- 병렬처리, Deep Learning을 활용하여 실제공간에서보다 100만배의 빠른 학습을 가능하게 함

Application of Reinforcement Learning

3. NC soft - Blade & Soul

장르	MMORPG
제작	NCSoft Team Bloodlust
등급	청소년 이용불가
플랫폼	PC
엔진	언리얼 엔진 3
유통	엔씨소프트
발매	2012년 6월 30일
링크	공식 홈페이지

- 국내 NC soft에서 서비스 중인 MMORPG
- 새로운 컨텐츠에 AI를 접목시킴
  - 100개의 층을 클리어 하는 것을 목표
  - 이 때 각 층의 보스의 공격 패턴에 Reinforcement learning을 접목
  - 매 회 플레이어에 맞춰 강해지는 적을 구현

Application of Reinforcement Learning

4. Microsoft - Minecraft

MINECRAFT	
	
개발	Mojang AB
유통	마이크로소프트
장르	샌드박스, 서바이벌게임
등급	**게임콘텐츠등급분류위원회** 12세 이용가[1]
	ESRB EVERYONE 10+ (전체 이용가이지만, 10세 이상)

- Microsoft에서 서비스하는 샌드박스형 게임
- 채팅을 통해 NPC를 학습, 협업하는 시스템 개발 목적
  - 장애물을 피하는 경로 추적 알고리즘
  - 물건을 인식하여 사용법을 익히는 알고리즘
  - Reinforcement Learning을 이용한 강화학습 활용
- 오픈소스 프로젝트로 모든 코드가 공개 중

  https://github.com/Microsoft/malmo