기말고사 일정

- 일자 :2018년 12월 17일
- 시간 : 오후 3시~ (수업시간과 동일)
- 장소 : 제5공학관 5120호 (또는 5125호)

  * 장소 변경시에는 Ucheck와 5120호 문앞에 공지예정

주요 범위

- 기계학습(Machine Learning) + 그 종류
- 강화학습(Reinforcement Learning)
- MDP(Markov Decision Process)와 그 구성요소

- 장기적 보상, 가치함수, Q함수
- 벨만 방정식
- Value Iteration 동작 예
- Q-Learning의 ξ-탐욕정책을 쓰는 이유
- DQN(Deep Q-Learning Network)의 상태(state)

가깝고도 먼 DeepRL

이웅원

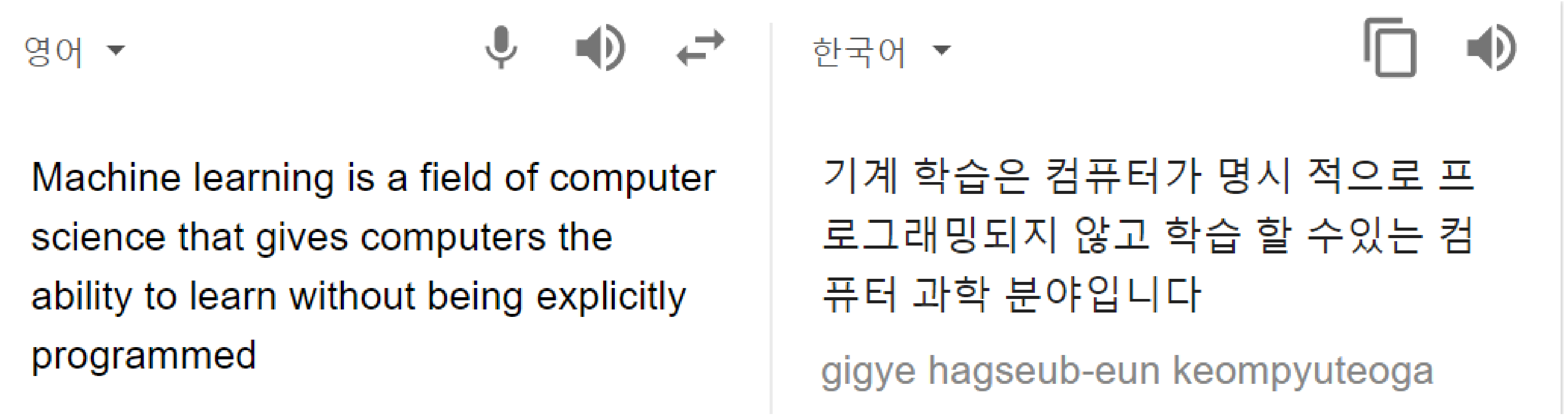
2018.02.07

J.MARPLE

Machine Learning 은 무엇인가

- 위키피디아 정의

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed



Machine Learning 은 무엇인가

Explicit Programming		Machine Learning
If 배가 고프면, then 밥을 먹어라	→	데이터 기반 → 예측 + 학습
간단한 문제		복잡한 문제

Ex) 스팸필터, 추천 시스템,

날씨와 교통상황 사이의 상관관계

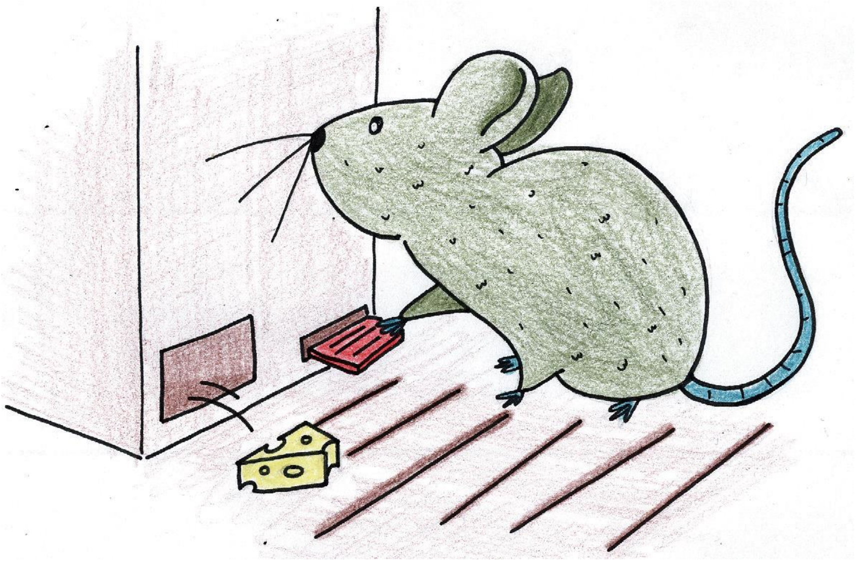
Machine Learning 은 무엇인가

Machine Learning의 종류

1. Supervised Learning : 정답이 있는 데이터 학습
2. Unsupervised Learning : 데이터 자체의 특성 학습
3. Reinforcement Learning : 보상으로부터 학습

Reinforcement 는 무엇인가

1. 행동주의와 Skinner → "눈으로 관찰가능한 행동을 연구"
2. Skinner의 문제상자 : 레버를 누르면 먹이가 나오는 상자 안에 굶긴 쥐를 넣고 실험



Reinforcement 는 무엇인가

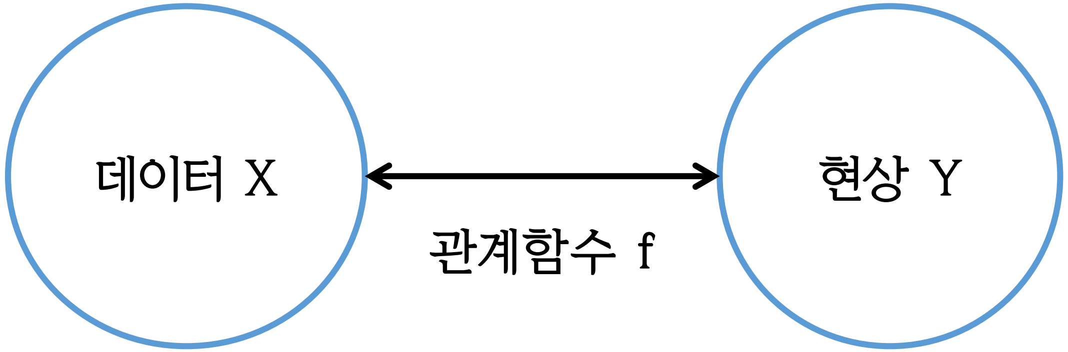
1. 굶긴 쥐를 상자에 넣는다
2. 쥐는 돌아다니다가 우연히 상자 안에 있는 지렛대를 누르게 된다
3. 지렛대를 누르자 먹이가 나온다
4. 지렛대를 누르는 행동과 먹이와의 상관관계를 모르는 쥐는 다시 돌아다닌다
5. 그러다가 우연히 쥐가 다시 지렛대를 누르면 쥐는 이제 먹이와 지렛대 사이의 관계를 알게 되고 점점 지렛대를 자주 누르게 된다
6. 이 과정을 반복하면서 쥐는 지렛대를 누르면 먹이를 먹을 수 있다는 것을 학습한다[1]

https://namu.wiki/w/%ED%96%89%EB%8F%99%EC%A3%BC%EC%9D%98

Reinforcement : 배우지 않았지만 직접 시도하면서 행동과 그 결과로 나타나는 보상 사이의 상관관계를 학습하는 것 → 보상을 많이 받는 행동의 확률을 높이기

강화학습은 무엇인가

1. Reinforcement Learning = Reinforcement + Machine Learning

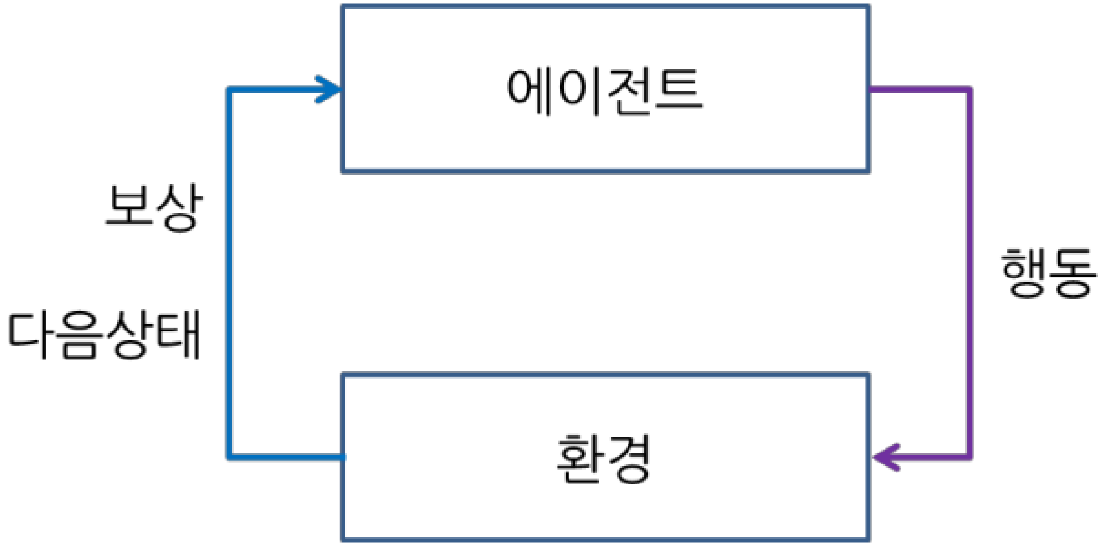


2. 데이터 X : 어떤 상황에서 어떤 행동을 했는지, 현상 Y: 보상을 얼마나 받았는지

→ 어떤 행동을 해야 보상을 많이 받는지를 데이터로부터 학습

강화학습은 무엇인가

1. 에이전트와 환경의 상호작용 → 데이터 생성 (미리 모아 놓을 수 없다)
2. 특정 상태에서 특정 행동을 선택 → 보상 → 학습

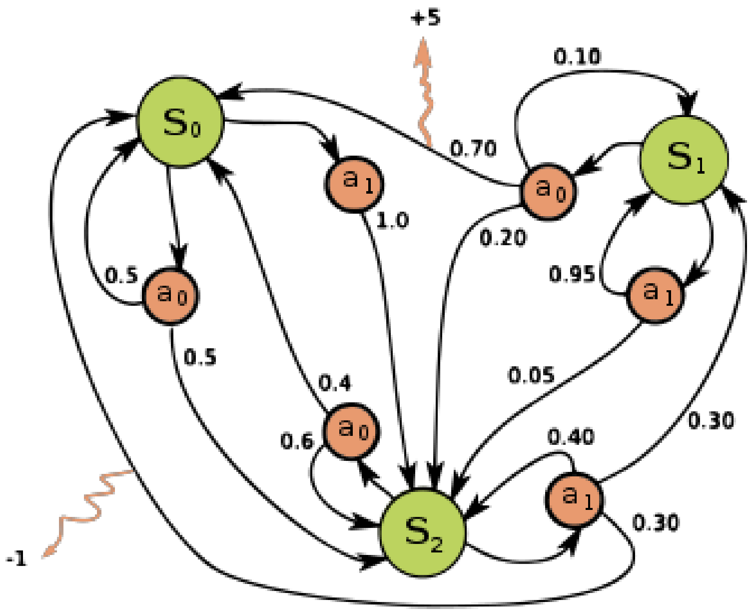


2-2. Markov Decision Process

Markov Decision Process

- 시간에 따라 변하는 “상태”가 있으며 상태 공간 안에서 움직이는 “에이전트”가 있다
  - 에이전트는 행동을 선택할 수 있다 → 확률적
  - 에이전트의 행동에 따라 다음 상태와 보상이 결정된다 → 확률적

→ 확률적 모델링 : Markov Decision Process



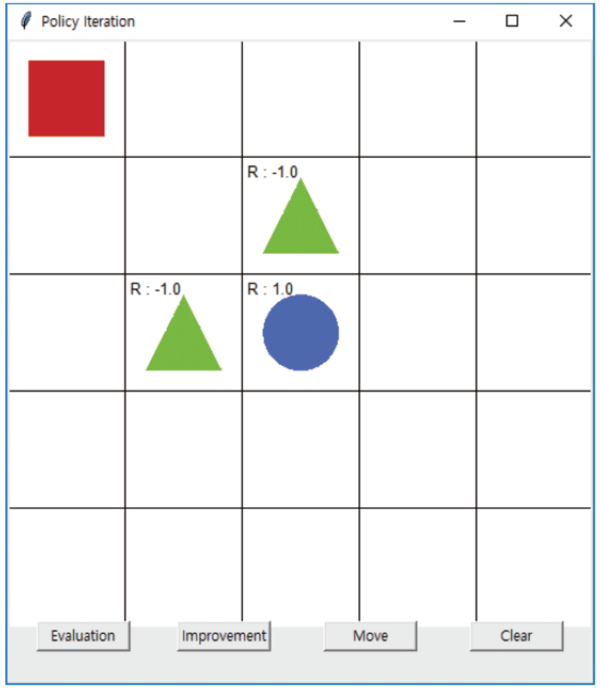
https://en.wikipedia.org/wiki/Markov_decision_process

Markov Decision Process

1. MDP = $\{S,\ A,\ R,\ P_{ss'}^{a},\ \gamma\}$ 로 정의되는 tuple
2. MDP의 구성요소
   - S : 상태(state)
   - A : 행동(action)
   - R : 보상(reward)
   - $P_{ss'}^{a}$: 상태변환확률(state transition probability)
   - γ : 할인율(discount factor)

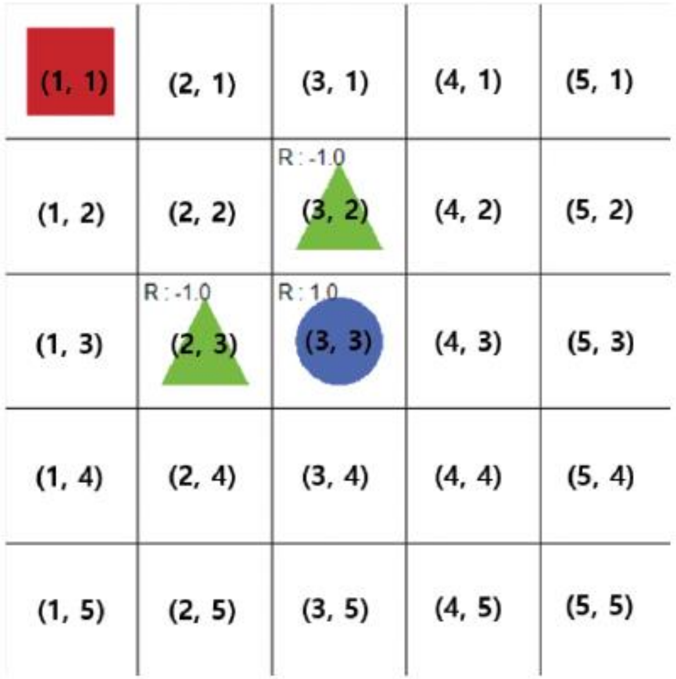
Grid World 예제

1. 격자를 기반으로 한 예제 : 5 X 5 = 25개의 격자를 가짐
2. 고전 강화학습의 가장 기본적인 예제 : 에이전트가 학습하는 과정을 눈으로 보기 쉬움
3. 목표 : 세모를 피해서 파란색 동그라미로 가기



MDP 1 : 상태

- 상태 : 에이전트가 관찰 가능한 상태의 집합
- 그리드월드의 상태 : $\boldsymbol{S} = \{(1,1),(2,1),(1,2),\ \cdots,\ (5,5)\}$

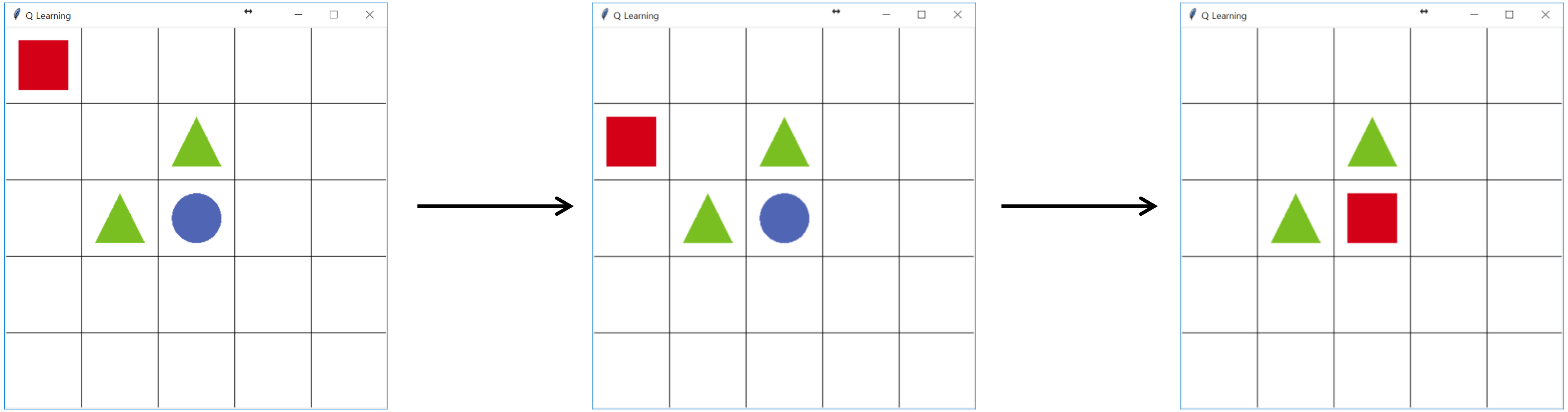


MDP 1 : 상태

1. 에이전트는 시간에 따라 환경을 탐험 → 상태도 시간에 따라 변한다
2. 시간 t일 때 상태 : $S_t = s \quad or \quad S_t = (1, 3)$
   - 확률변수(random variable)은 대문자, 특정 상태는 소문자
3. Episode : 처음 상태부터 마지막 상태까지

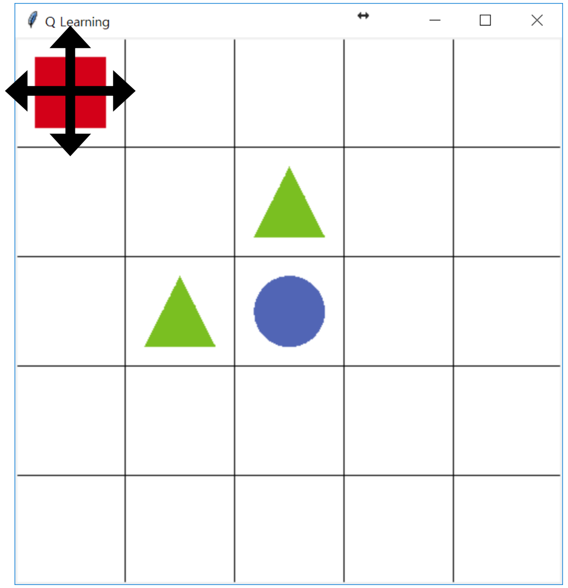
$$\tau = s_0, s_1, s_2, \cdots, s_{T-1}, s_T$$

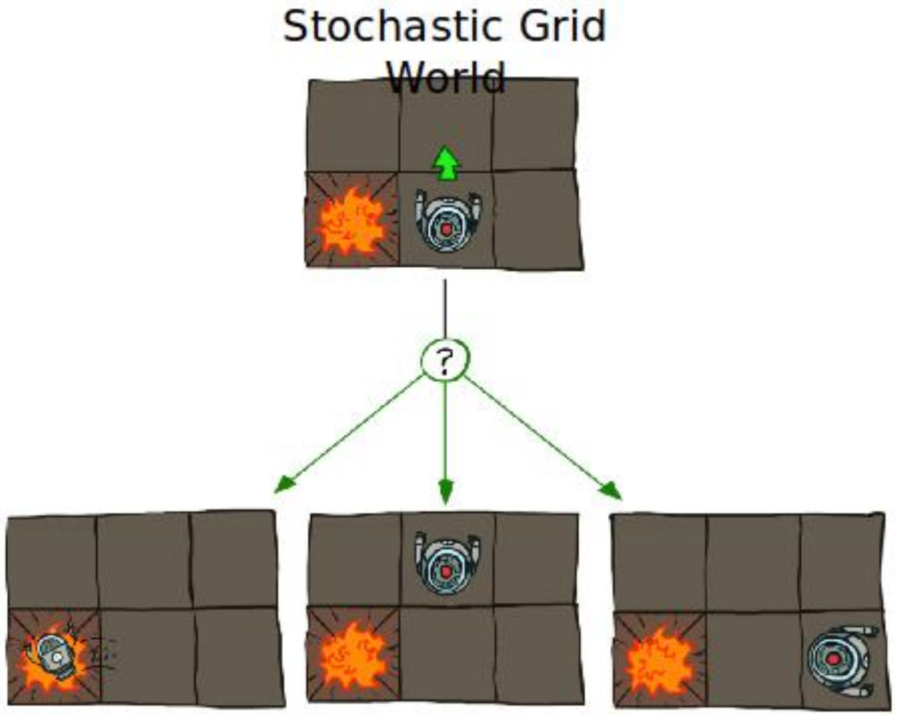
s_0 s_t s_T



MDP 2 : 행동

- 에이전트가 할 수 있는 행동의 집합 : $\boldsymbol{A} = \{위, 아래, 좌, 우\}$
- 시간 t에 취한 행동 $A_t = a$
- 만약 $A_t =$ 우 라면 항상 (3, 1)에서 (4, 1)로 갈까?
  - 상태 변환 확률에 따라 다르다

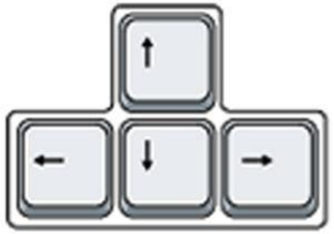


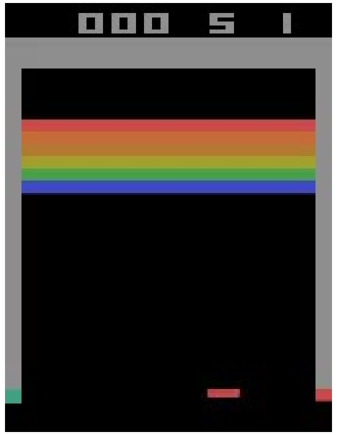


MDP 2 : 행동

- 행동 → (1) discrete action (2) continuous action

discrete action





continuous action

MDP 3 : 보상

- 에이전트가 한 행동에 대한 환경의 피드백 : 보상(+ or –)
- 시간이 t이고 상태 $S_t = s$ 에서 $A_t = a$를 선택했을 때 받는 보상

$$R_s^a = \boldsymbol{E}[R_{t+1} | S_t = s, A_t = a]$$

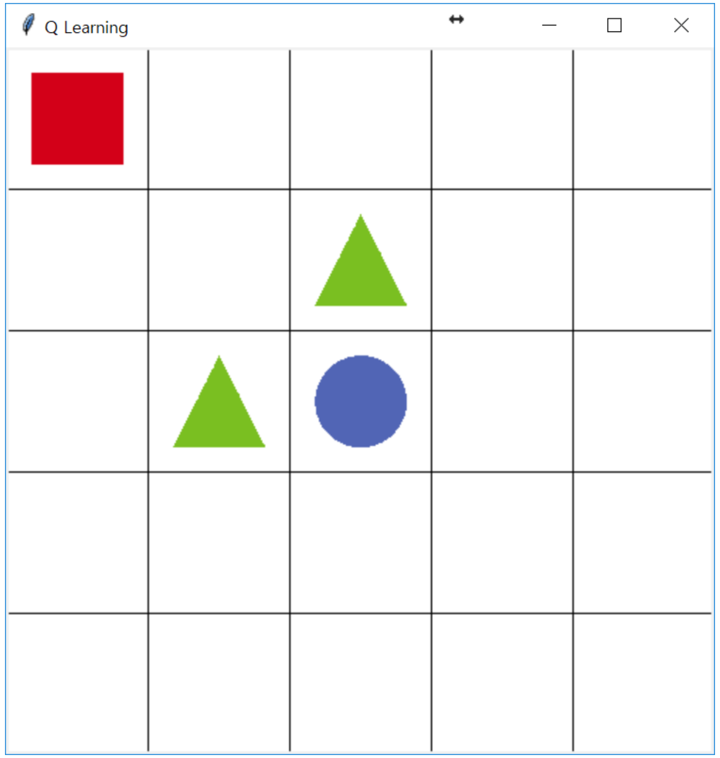
- 보상은 R_s^a으로 표현되거나 $R_{ss'}^a$으로 표현된다
- 보상은 현재 시간 t가 아닌 $t+1$에 환경으로부터 받는다
- 같은 상태 s에서 같은 행동 a를 했더라도 그때 그때마다 보상이 다를 수 있음

  → 기댓값(expectation)으로 표현

MDP 3 : 보상

1. 보상은 에이전트의 목표에 대한 정보를 담고 있어야 함
2. 그리드월드의 보상 : 초록색 세모 (-1), 파란색 동그라미 (+1)

→ 초록색 세모를 피해 파란색 동그라미로 가라!



MDP 4 : 상태변환확률

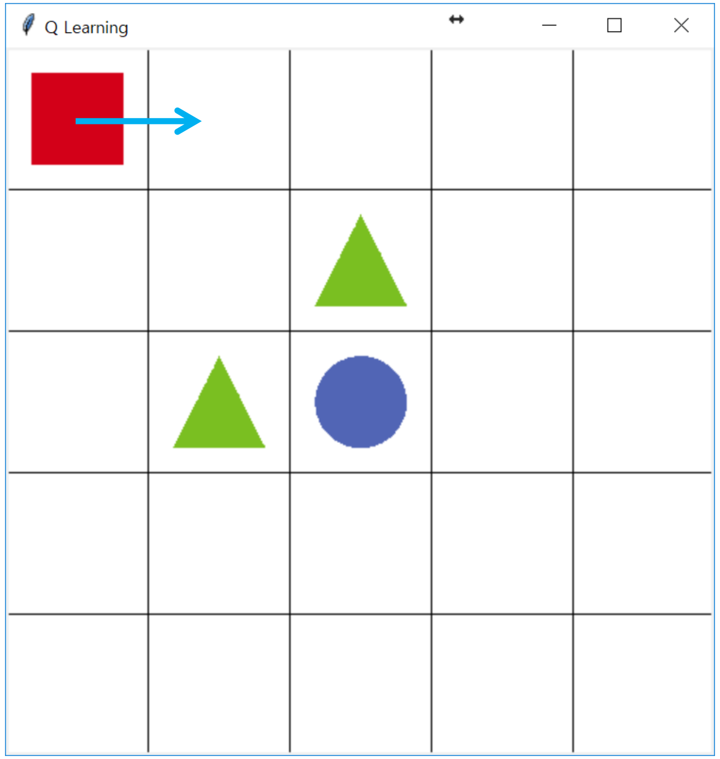
- 상태변환확률 : 상태 s에서 행동 a를 했을 때 상태 s'으로 갈 확률

$$P_{ss'}^{a} = \boldsymbol{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- model or dynamics of environment
- 상태변환확률을 안다면 : model-based
  - Dynamic Programming
- 상태변환확률을 모른다면 : model-free
  - Reinforcement Learning
- 상태변환확률을 학습한다면 : model-based RL
  - Dyna-Q

MDP 4 : 상태변환확률

$$P_{ss'}^{a} = \boldsymbol{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

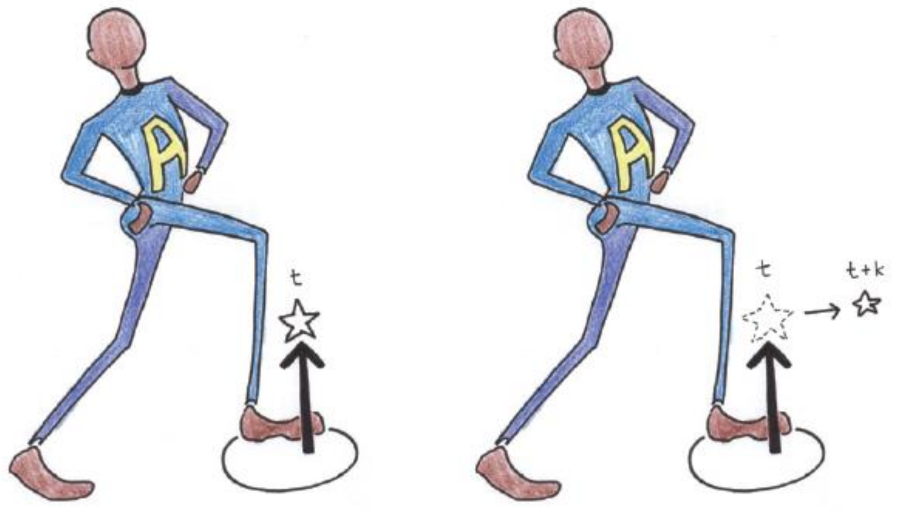


상태 (1, 1)에서 행동 “우”를 했을 경우

1. 상태 (2, 1)에 갈 확률은 0.8
2. 상태 (1, 2)에 갈 확률은 0.2

MDP 5 : 할인율

- 할인율 : 미래에 받은 보상을 현재의 시점에서 고려할 때 할인하는 비율
- 만약 복권에 당첨되었다면 당첨금 1억원을 당장 받을지 10년 뒤에 받을지?
  - 가까운 보상이 미래의 보상보다 더 가치가 있다 → 할인
- 보상에서 시간의 개념을 포함하는 방법



MDP 5 : 할인율

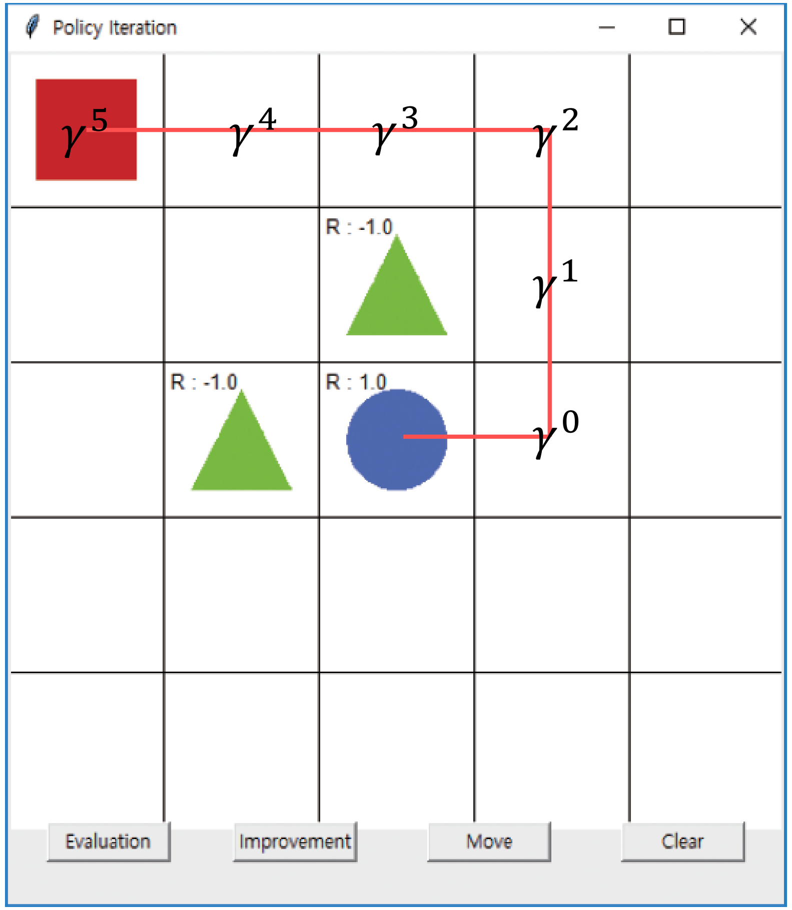
- 할인율은 0에서 1 사이의 값

$$\gamma \in [0, 1]$$

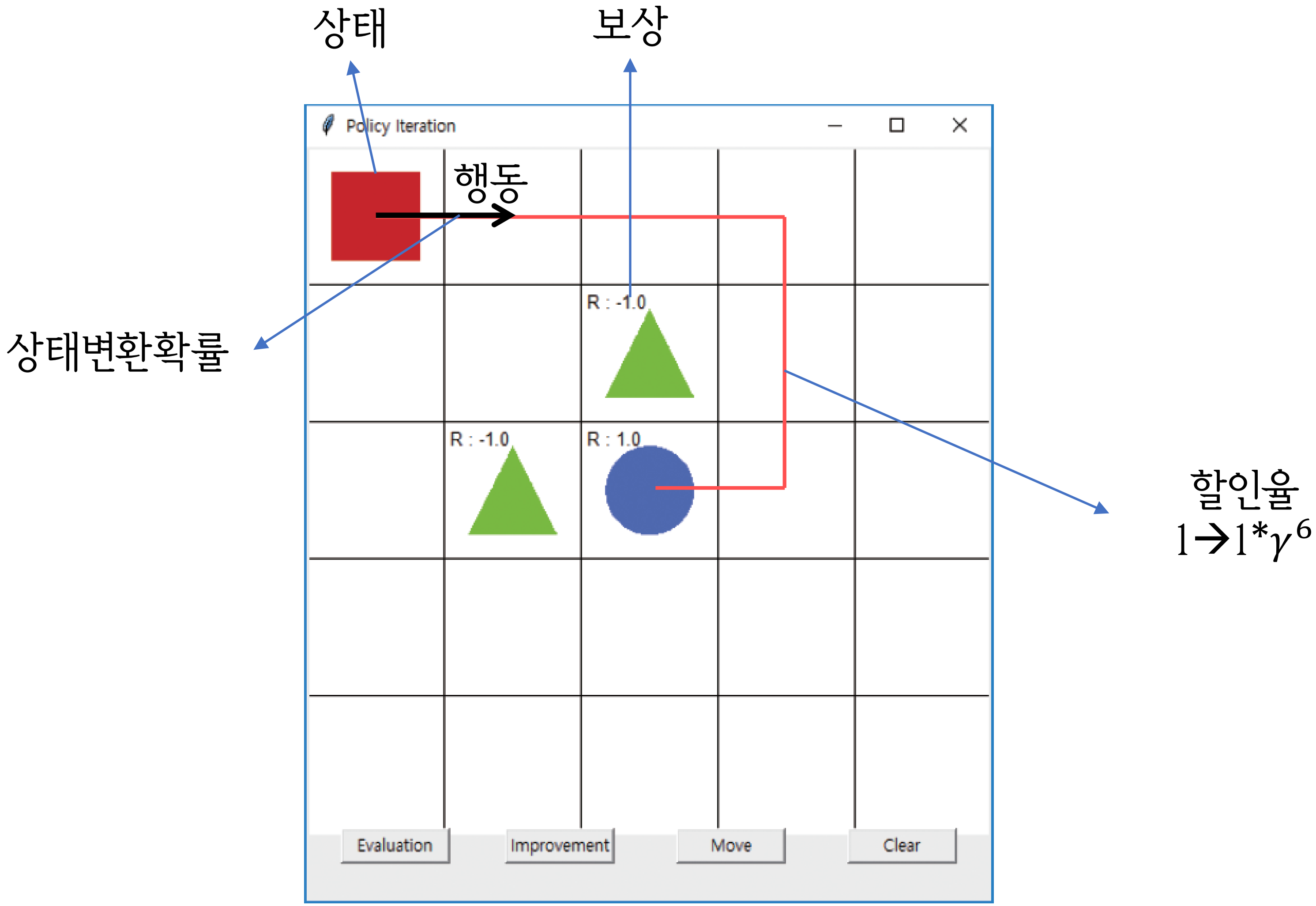
- 현재의 시간 t로부터 k만큼 지난 후 받은 보상의 현재 가치

$$\gamma^{k-1} R_{t+k}$$

- 할인율을 통해 보상을 얻는 최적의 경로를 찾을 수 있다



정리



그리드월드 문제에서의 MDP

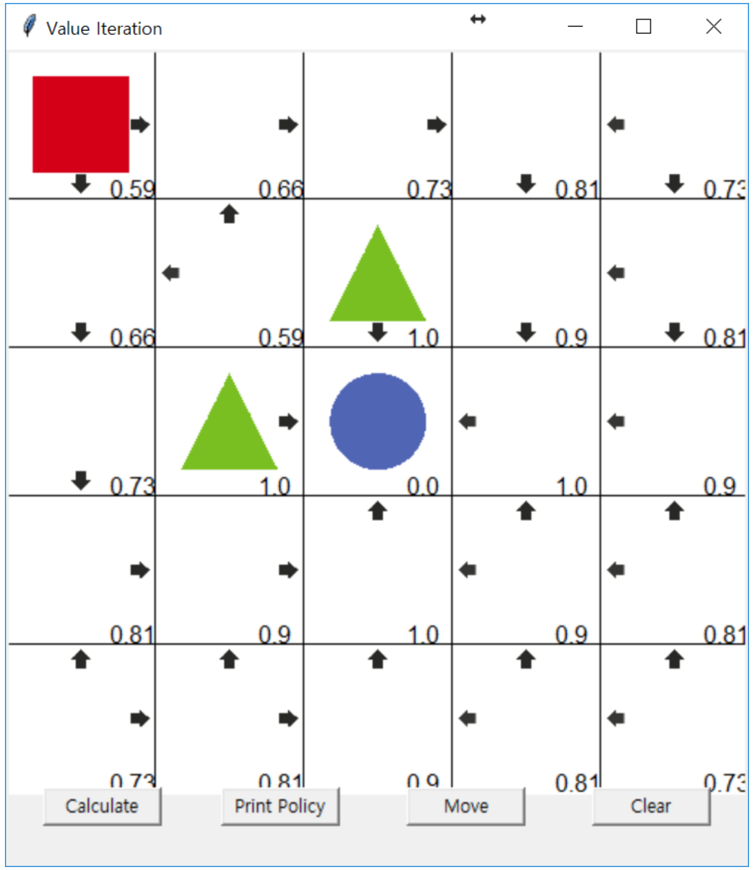
정책

1. 에이전트는 각 상태마다 행동을 선택

2. 각 상태에서 어떻게 행동할지에 대한 정보 : 정책(Policy)

   • 상태 s에서 행동 a를 선택할 확률

$$\pi(a|s)$$



3. 두 가지 형태의 정책

   • 행동 = 정책(상태) → 명시적(explicit) 정책

   • 행동 = 선택(가치함수(상태)) → 내재적(implicit) 정책

정리

1. 강화학습이 풀고자 하는 문제
   - Sequential Decision Problem
2. Sequential Decision Problem의 수학적 정의
   - MDP
3. MDP의 구성요소
   - 상태, 행동, 보상, 상태변환확률, 할인율
4. 각 상태에서 에이전트가 행동을 선택할 확률
   - 정책

2-3. Bellman Equation

MDP 에이전트의 행동 선택

1. 에이전트와 환경의 상호작용(Value-based)

   (1) 에이전트가 상태를 관찰

   (2) 어떠한 기준에 따라 행동을 선택

   (3) 환경으로부터 보상을 받음

   * 어떠한 기준 : 가치함수, 행동 선택 : greedy action selection

2. 에이전트 행동 선택의 기준

   (1) 에이전트는 매 타임스텝마다 보상을 더 많이 받으려 함

   (2) 단기적 보상만 고려한다면 최적의 정책에 도달할 수 있을까?

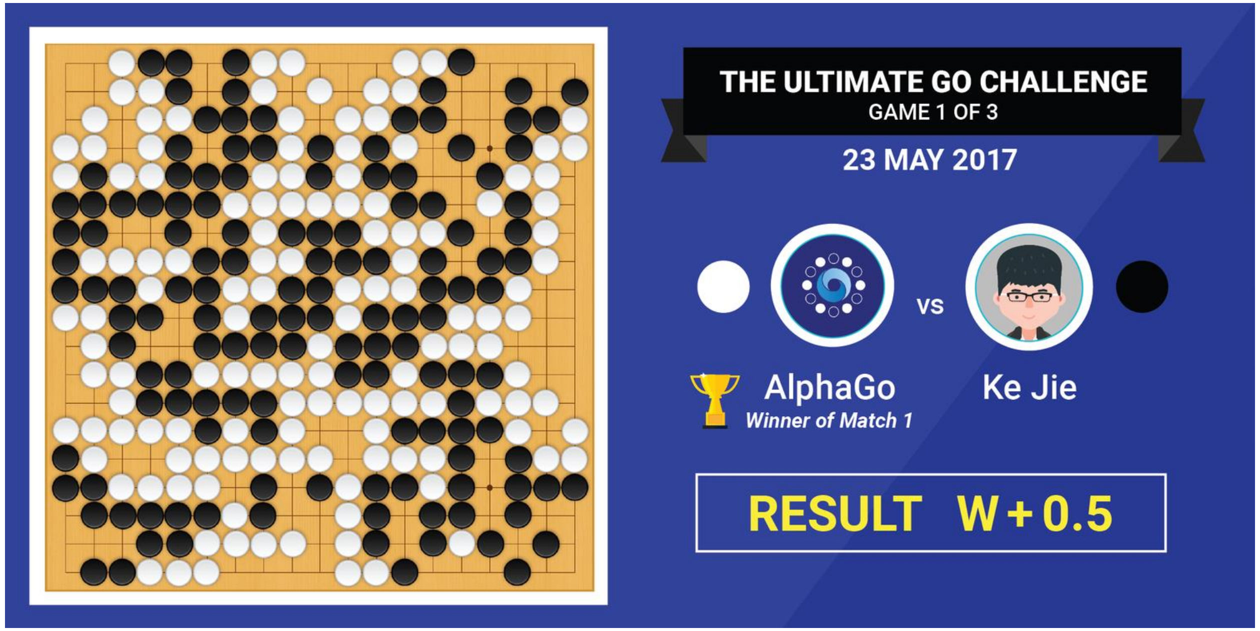
   - Problem 1: sparse reward
   - Problem 2: delayed reward

Sparse Reward

- 매 타임스텝마다 보상이 나오지 않는다면?

  Ex) 바둑

- 대부분의 경우 보상이 sparse 하게 주어짐



http://www.ibtimes.co.uk/alphago-defeats-worlds-best-go-player-ke-jie-humans-prove-no-match-ai-again-1622960

Delayed Reward

- 보통 선택한 행동에 대한 보상은 delay되어서 에이전트에게 주어진다

→즉각적 보상만 고려해서 선택하면 어떤 행동이 좋은 행동이었는지 판단 어려움

→"credit assignment problem"

이 행동만이 좋은 행동이고 나머지는 아니다?

장기적 보상

- 단기적 보상만 고려했을 때의 문제
  (1) Sparse reward (2) delayed reward

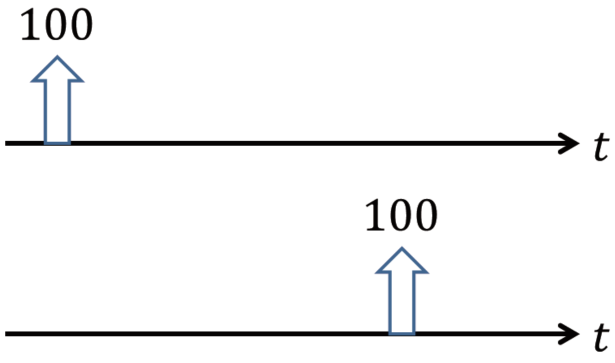
- 단기적 보상이 아닌 지금 선택한 행동에 대한 장기적인 결과(보상)를 보자!
- 장기적 보상을 어떻게 알아낼 수 있을까?
  (1) 반환값(Return)
  (2) 가치함수(Value function)

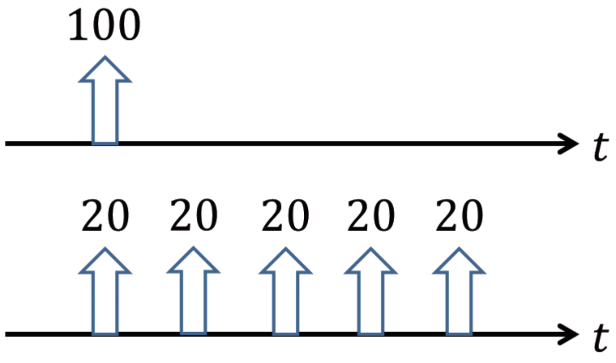
장기적 보상 1 : 단순합

- 단기적 보상이 아닌 장기적 보상 → 앞으로 받을 보상을 고려
- 현재 시간 t로부터 앞으로 받을 보상을 다 더한다

$$R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$$

- 현재 시간 t로부터 앞으로 받을 보상을 다 더한다

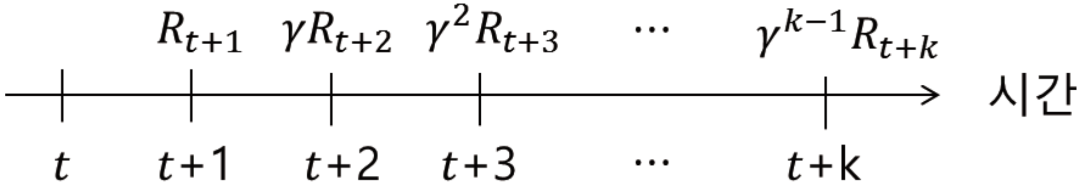




$$0.1 + 0.1 + \cdots = \infty$$
$$1 + 1 + \cdots = \infty$$

장기적 보상 2 : 반환값

- 현재 시간 t로부터 에피소드 끝까지 받은 보상을 할인해서 현재 가치로



- 반환값(Return) : 현재 가치로 변환한 보상들을 다 더한 값

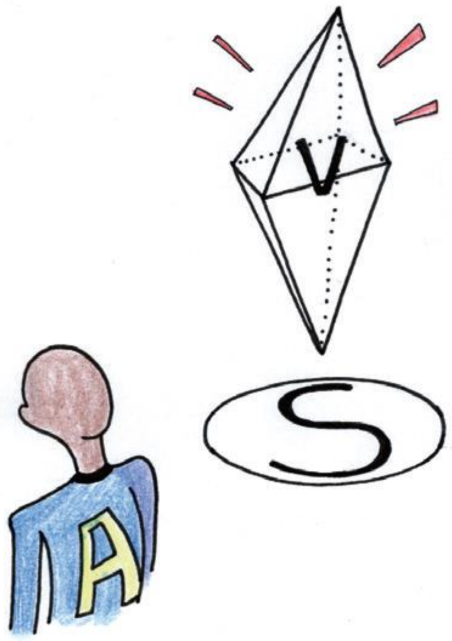
$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$$

- 실제 에이전트와 환경의 상호작용을 통해 받은 보상을 이용
  - Unbiased estimator : 실제 환경으로부터 받은 값
  - High variance : 시간 t 이후에 행동을 어떻게 하는지에 따라 값이 크게 달라짐

장기적 보상 3 : 가치함수

- 가치함수(Value function) : 반환값에 대한 기댓값
  - 어떠한 상태 s에 갈 경우 그 이후로 받을 것이라 예상되는 보상에 대한 기대
  - 반환값은 에이전트의 정책에 영향을 받음

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[G_t | S_t = s]$$



- 반환값은 상태 s에서 어떤 행동을 선택하는지에 따라 다름
- 가치함수는 상태 s로만 정해지는 값 → 가능한 반환값들의 평균
- 기댓값을 계산하기 위해서는 환경의 모델을 알아야함
  - DP는 가치함수를 계산
  - 강화학습은 가치함수를 계산하지 않고 sampling을 통한 approximation

가치함수 식의 변형

- 벨만 기대 방정식(Bellman expectation equation)의 유도

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[G_t | S_t = s]$$

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma R_{t+2} + \cdots \, | S_t = s]$$

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma (R_{t+2} + \cdots) \, | S_t = s]$$

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma \boldsymbol{E}_{\boldsymbol{\pi}}(R_{t+2} + \cdots) \, | S_t = s]$$

$$v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma \boldsymbol{E}_{\boldsymbol{\pi}}(G_{t+1}) | S_t = s]$$

$$\underline{v_\pi(s) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]}$$

큐함수에 대한 정의

- 큐함수(Q-function)

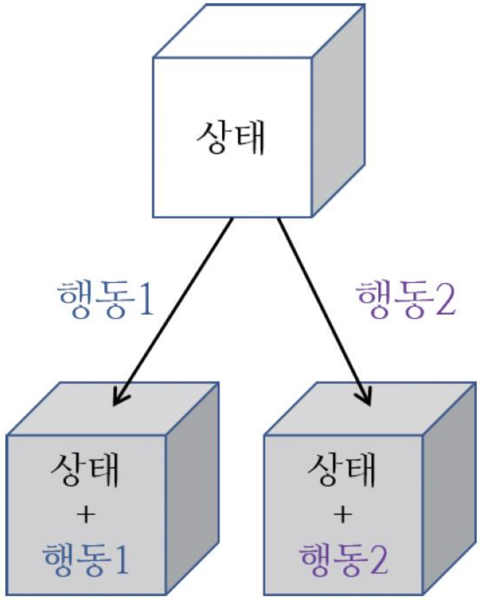
상태 s에서 행동 a를 했을 경우 받을 것이라 예상되는 반환값에 대한 기댓값

$$q_\pi(s,a) = \boldsymbol{E_\pi}[G_t | S_t = s, A_t = a]$$

- 가치함수는 큐함수에 대한 기댓값

$$v_\pi(s) = \boldsymbol{E}_{a\sim\pi}[q_\pi(s,a) | S_t = s]$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$



정책을 고려한 벨만 기대 방정식

- 정책 π에 따라 행동을 선택할 때의 벨만 기대 방정식

1. 가치함수에 대한 벨만 기대 방정식

$$v_\pi(s) = \boldsymbol{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

2. 큐함수에 대한 벨만 기대 방정식

$$q_\pi(s, a) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$$

벨만 방정식은 현재 상태 s와 다음 상태 S_{t+1}의 가치함수(큐함수) 사이의 관계식

벨만 기대 방정식과 최적의 정책

- 벨만 기대 방정식 → 정책 π를 따라갔을 때의 가치함수

$$v_\pi(s) = \boldsymbol{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$$

- 에이전트가 알고자 하는 것 : π^*
  - 가장 높은 보상을 얻게 하는 최적의 정책 π^*

- 최적의 정책 π^*는 deterministic policy
  - 상태 s에서 가장 큰 큐함수를 가지는 행동 a를 반환
  - 이 때, 큐함수 또한 최적의 큐함수

벨만 최적 방정식

- 벨만 최적 방정식 : 행동을 선택할 때는 max, 가치함수 or 큐함수도 최적
- 가치함수에 대한 벨만 최적 방정식

$$v^*(s) = \max_a \boldsymbol{E}[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a]$$

- 큐함수에 대한 벨만 최적 방정식

$$q^*(s, a) = \boldsymbol{E}[R_{t+1} + \gamma \, max_{a'} \, q^*(S_{t+1}, a') \, | S_t = s, A_t = a]$$

정리

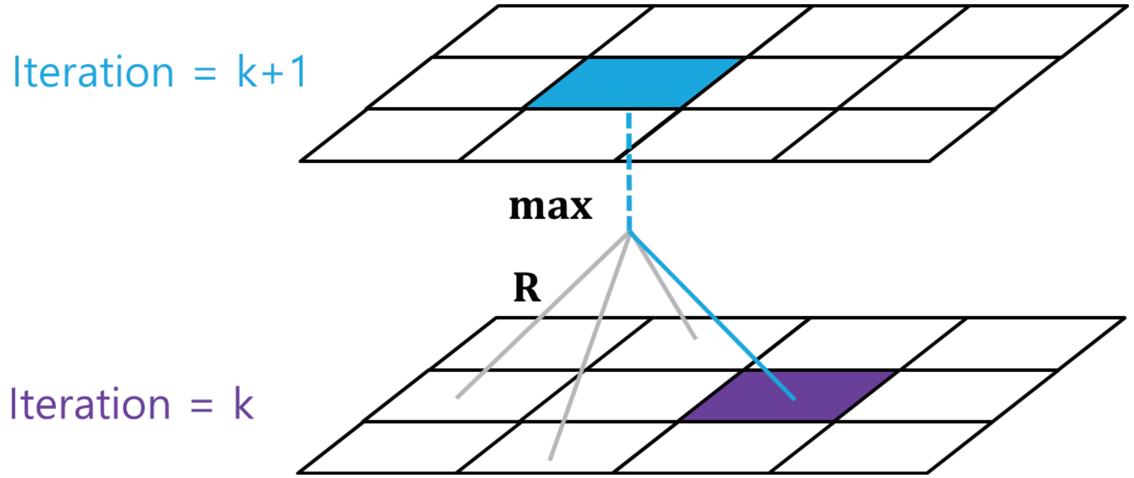
1. 단기적 보상 → 장기적 보상
   - Sparse reward & delayed reward
   - 가치함수 & 큐함수
2. 벨만 기대 방정식 (Bellman Expectation Eqn.)
   - $v_\pi(s) = \boldsymbol{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s]$
   - $q_\pi(s, a) = \boldsymbol{E}_{\boldsymbol{\pi}}[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]$
3. 벨만 최적 방정식 (Bellman Optimality Eqn.)
   - $v^*(s) = \max_a \boldsymbol{E}[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a]$
   - $q^*(s, a) = \boldsymbol{E}[R_{t+1} + \gamma \, max_{a'} \, q^*(S_{t+1}, a') \, | S_t = s, A_t = a]$

2-4. Value Iteration

Value Iteration

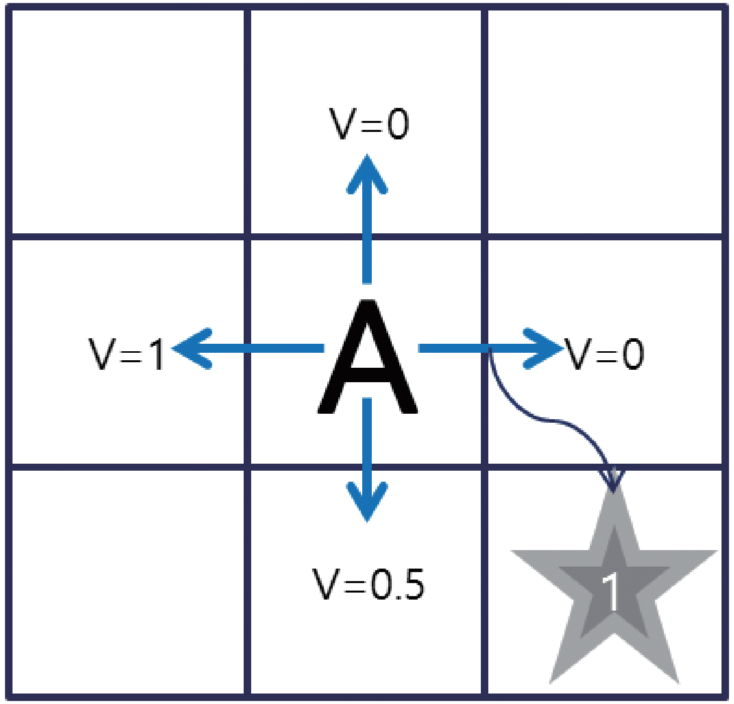
- Iteration 1번 : 모든 상태에 대해서 벨만 최적 방정식을 통한 업데이트 1번

$$\text{for } s \in \boldsymbol{S},\ v_{k+1}(s) \leftarrow \max_{a} \left[R_s^a + \gamma \sum_{a \epsilon A} P_{ss'}^a v_k(s') \right]$$



Grid world에서 Value Iteration

$$v_{k+1}(s) \leftarrow \max_a [R_s^a + \gamma v_k(s')]$$

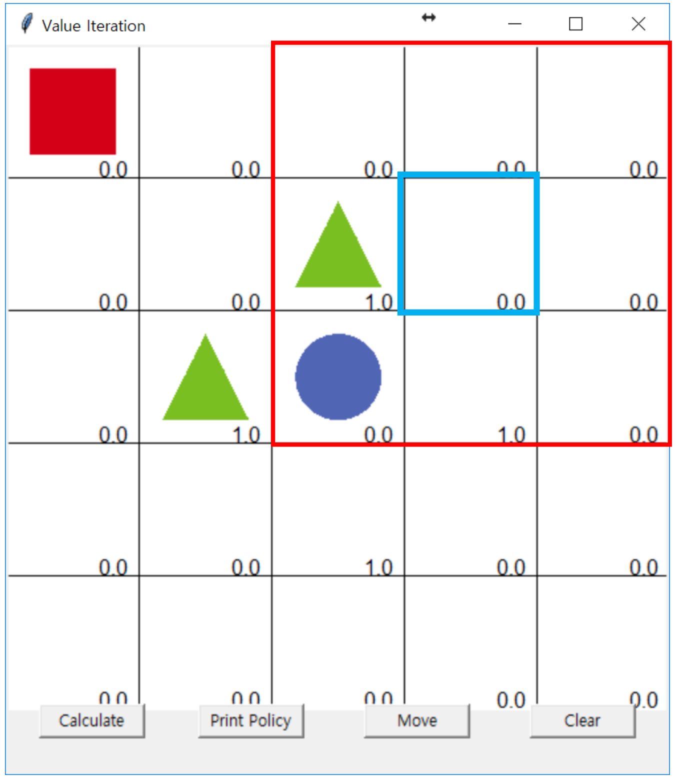


- '상' : $0 + 0.9 \times 0 = 0$
- '하' : $0 + 0.9 \times 0.5 = 0.45$
- '좌' : $0 + 0.9 \times 1 = 0.9$
- '우' : $1 + 0.9 \times 0 = 1$

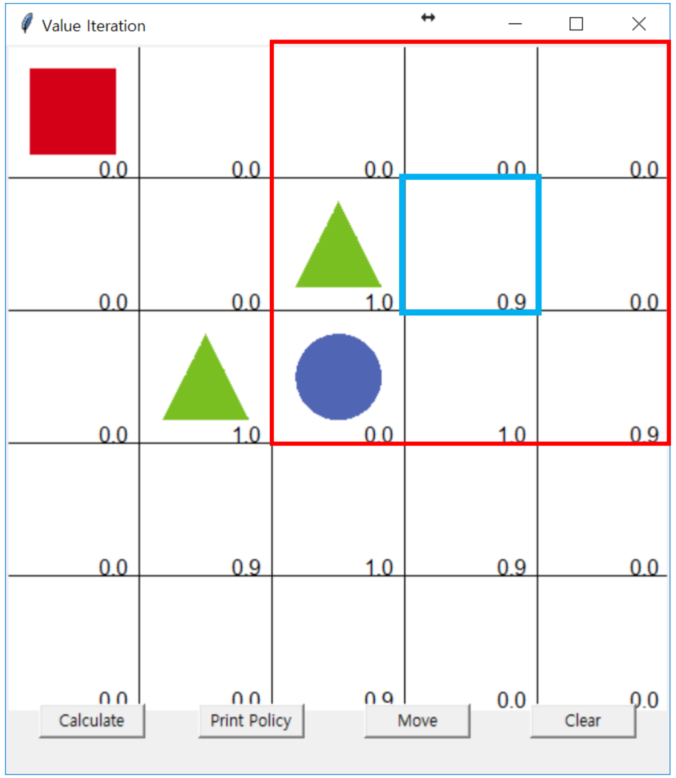
$$v_{k+1}(s) = \max[0, 0.45, 0.9, 1] \\ = 1$$

Grid world에서 Value Iteration

$$v_{k+1}(s) \leftarrow \max_{a}[R_s^a + \gamma v_k(s')]$$



K=1



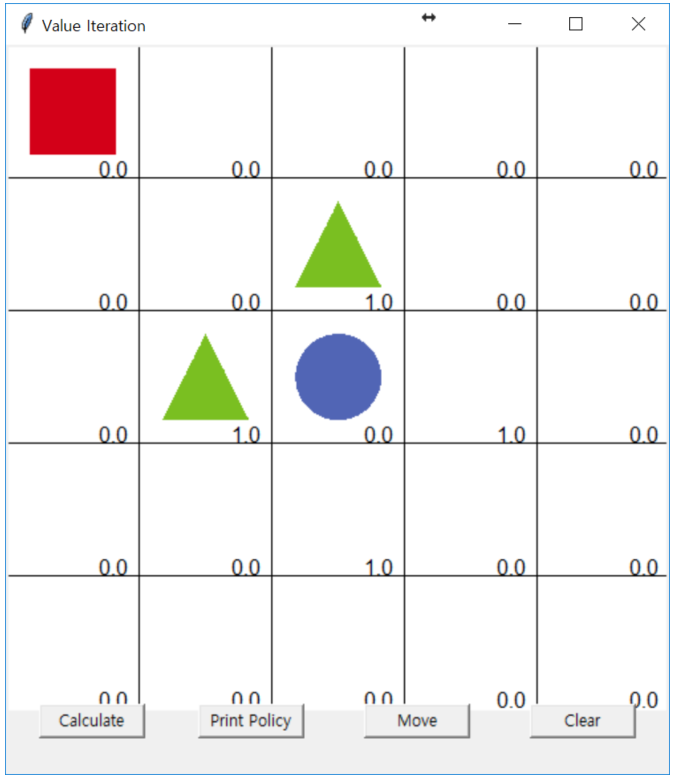
K=2

- '상' : $0 + 0.9 \times 0 = 0$
- '하' : $0 + 0.9 \times 1.0 = 0.9$
- '좌' : $-1 + 0.9 \times 1.0 = -0.1$
- '우' : $0 + 0.9 \times 0 = 0$

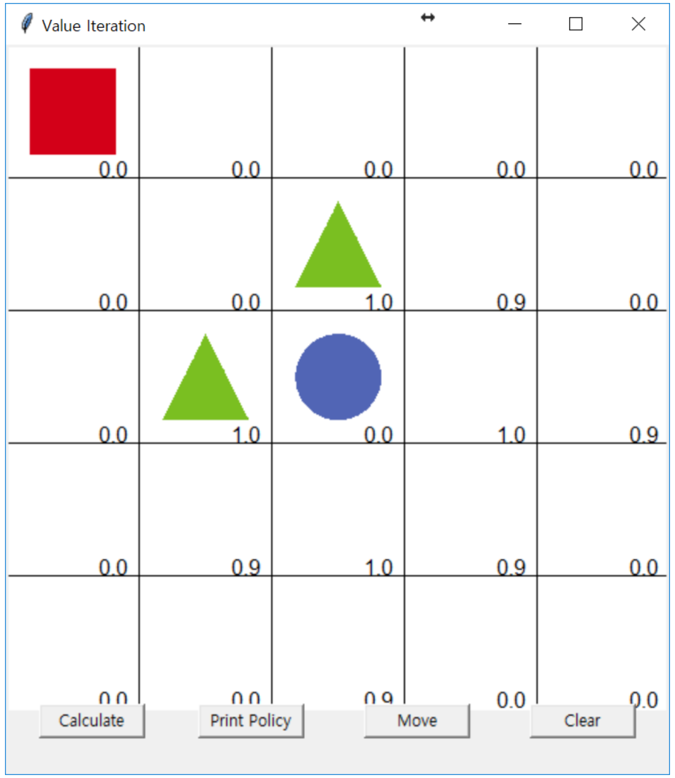
$$v_{k+1}(s) = \max[0, 0.9, -0.1, 0] = 0.9$$

Grid world에서 Value Iteration

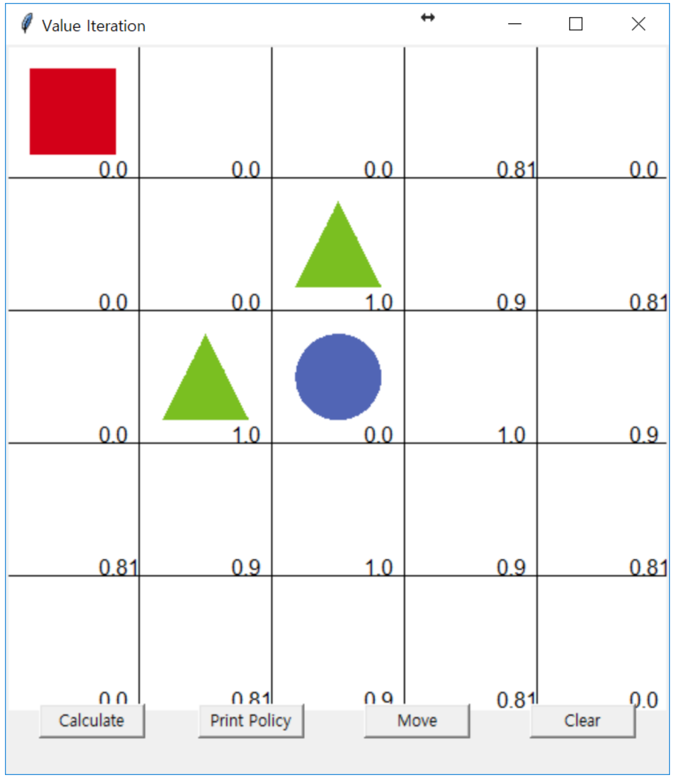
$$v_{k+1}(s) \leftarrow \max_{a}[R_s^a + \gamma v_k(s')]$$



K=1



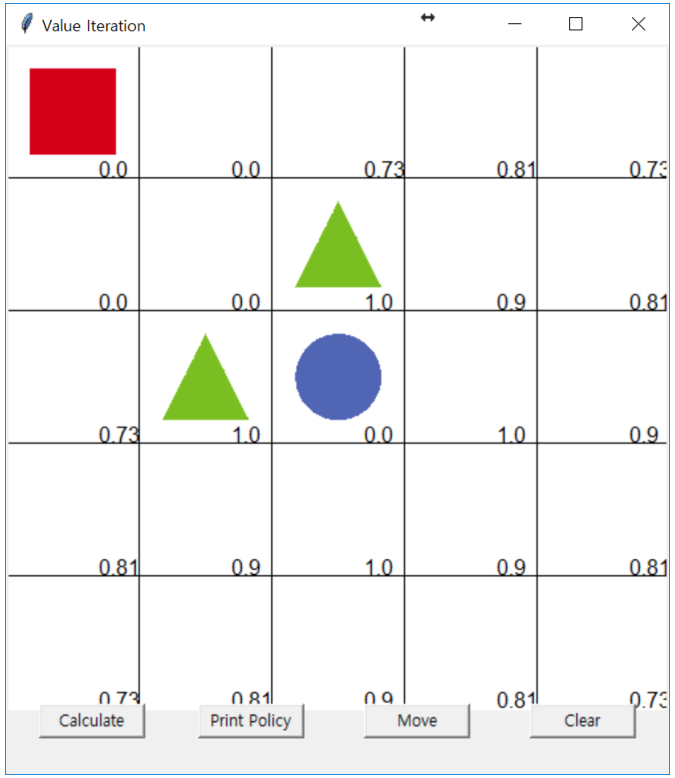
K=2



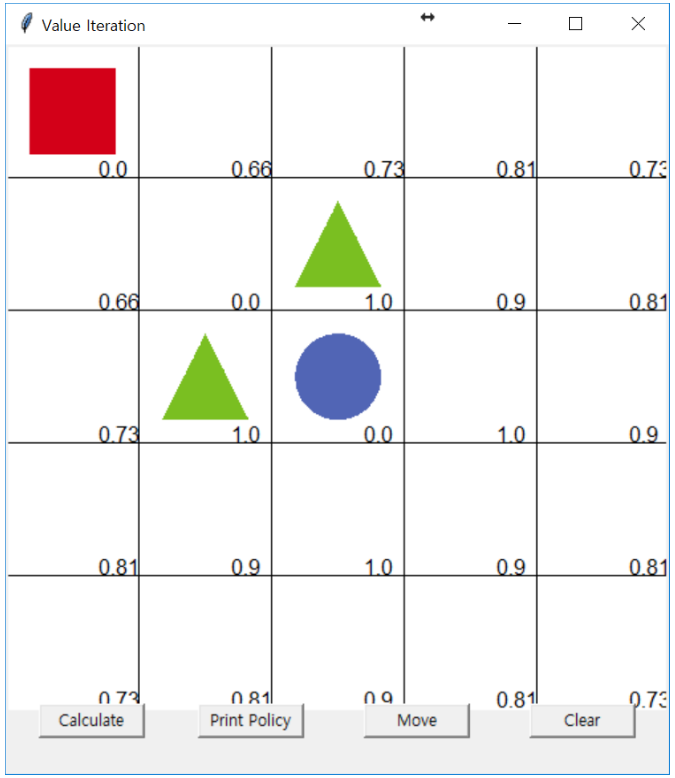
K=3

Grid world에서 Value Iteration

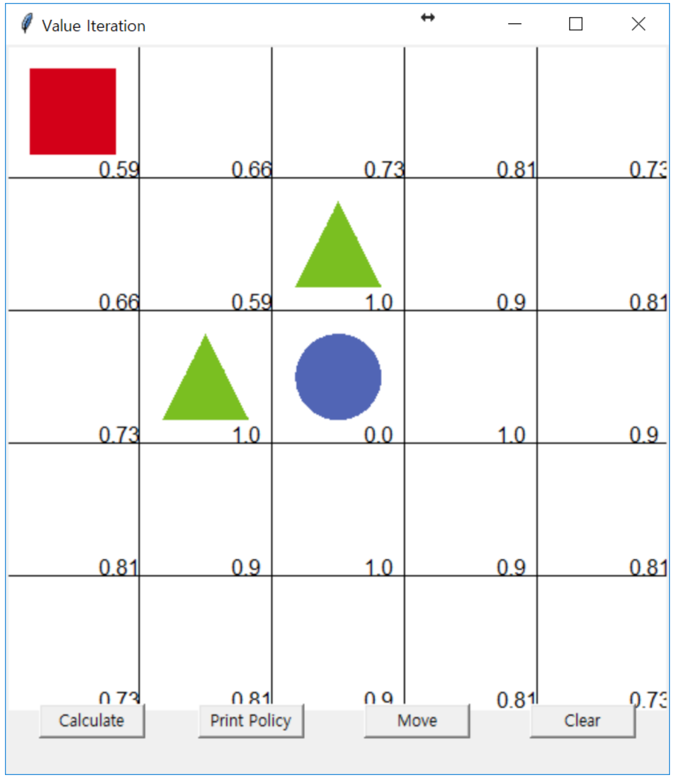
$$v_{k+1}(s) \leftarrow \max_a [R_s^a + \gamma v_k(s')]$$



K=4



K=5

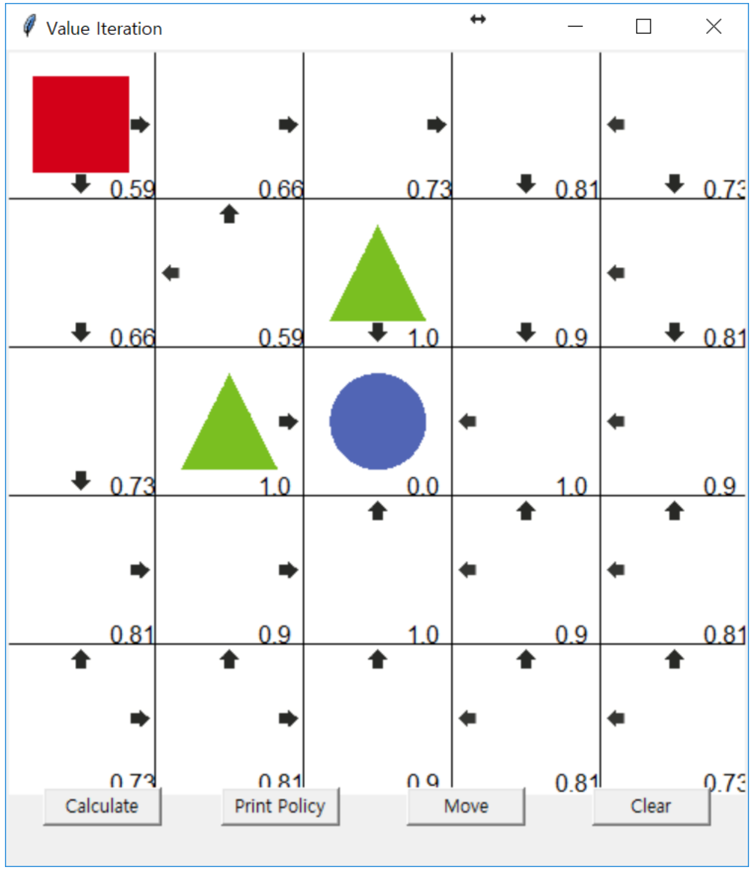


K=6

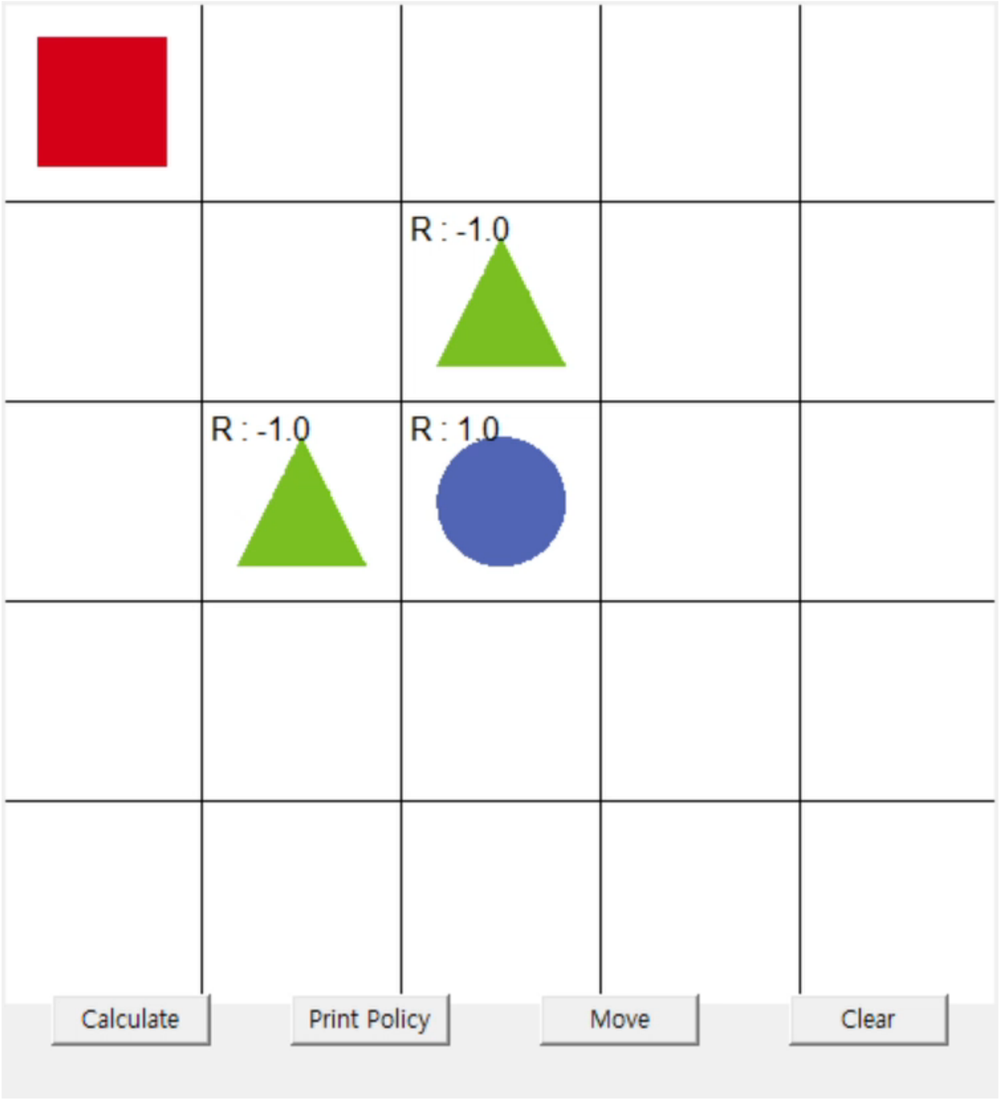
Grid world에서 Value Iteration

최적의 정책과 최적의 가치함수

$$\pi^*(s) = argmax_{a \in A} \boldsymbol{E}[R_s^a + \gamma v^*(s')]$$



Grid world에서 Value Iteration



정리

1. Dynamic Programming

- 큰 문제를 작은 문제로, 반복되는 문제를 값을 저장하면서 해결
- 큰 문제 : 최적 가치함수 계산 $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \cdots \rightarrow v^*$
- 작은 문제 : 현재의 가치함수를 더 좋은 가치함수로 업데이트 $v_k \rightarrow v_{k+1}$
- Bellman Eq.를 이용해서 1-step 계산으로 optimal을 계산하기

2. Value Iteration

- 가치함수가 최적이라고 가정하고 그 사이의 관계식인 벨만 최적 방정식 이용

$$v_{k+1}(s) \leftarrow \max_a [R_s^a + \gamma v_k(s')]$$

- 수렴한 가치함수에 대해 greedy policy
- Q-Learning으로 연결

2-5. Q-Learning

Q-Learning

1. Value Iteration

- 가치함수가 최적이라고 가정하고 그 사이의 관계식인 벨만 최적 방정식 이용

$$v_{k+1}(s) \leftarrow \max_a [R_s^a + \gamma v_k(s')]$$

- 수렴한 가치함수에 대해 greedy policy

2. Q-Learning

- 행동 선택 : $\varepsilon -$ 탐욕정책

$$\pi(s) = \begin{cases} a^* = argmax_{a \in A} q(s,a), & 1-\varepsilon \\ \quad a \neq a^* \qquad\qquad\qquad\quad , & \varepsilon \end{cases}$$

- 큐함수 업데이트 : 벨만 최적 방정식 이용

$$q(s,a) = q(s,a) + \ \alpha \left(r + \gamma \max_{a'} q(s',a') - q(s,a) \right)$$

ε − 탐욕정책

1. 탐욕 정책

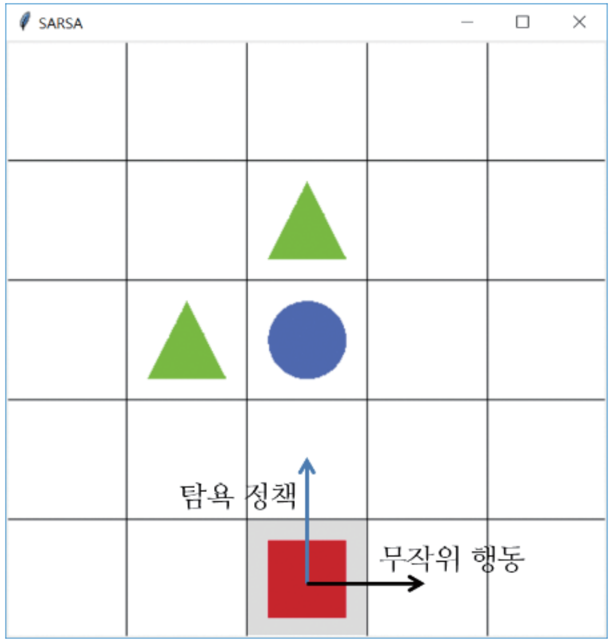
- 가치함수를 사용할 경우 $P^a_{ss'}$를 알아야 함

$$\pi(s) = argmax_{a\epsilon A}\left[R^a_s + \gamma \sum_{s'\in S} P^a_{ss'}\, v_{k+1}(s')\right]$$

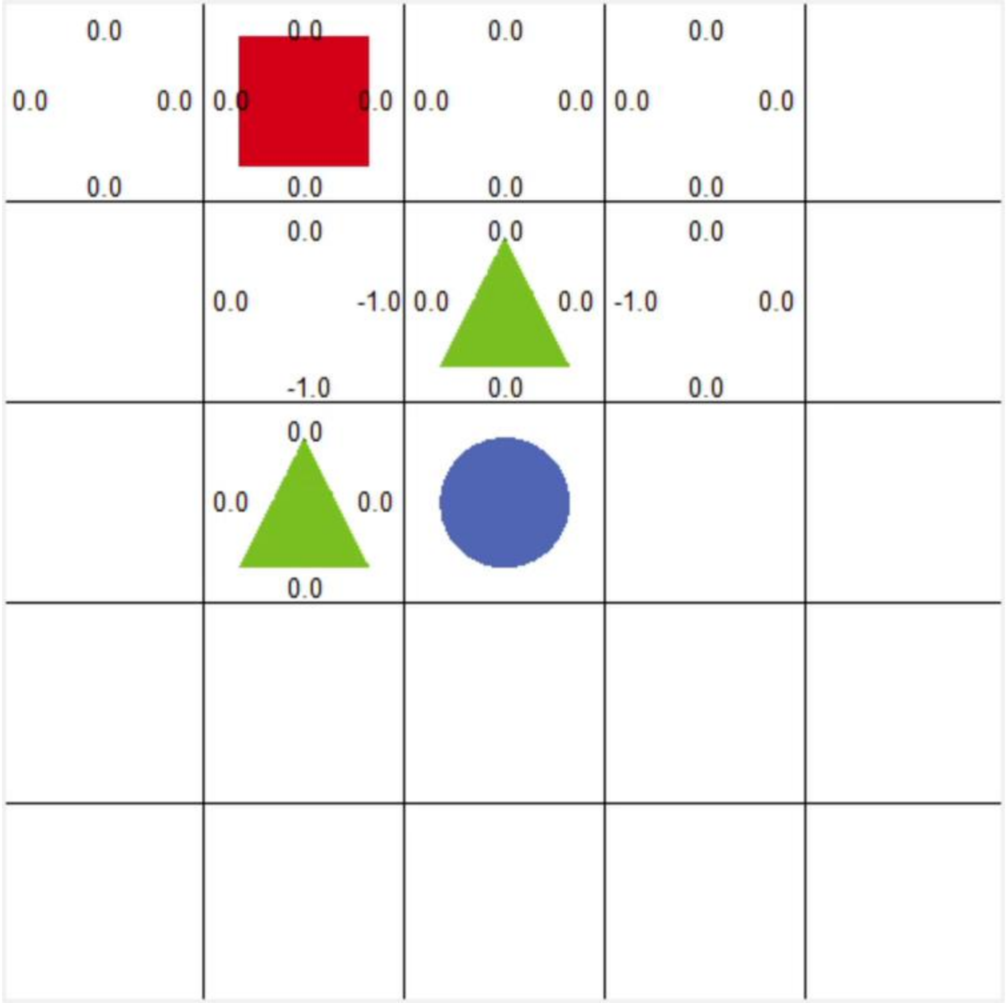
- 큐함수를 이용한 탐욕 정책 발전 : model−free$\pi(s) \leftarrow argmax_{a\epsilon A}[q(s,a)]$

2. ε − 탐욕정책

- ε의 확률로 랜덤한 행동을 선택 : $\pi(s) = \begin{cases} a^* = argmax_{a\in A} q(s,a), & 1-\varepsilon \\ random\ action \quad , & \varepsilon \end{cases}$



Grid world에서의 Q-Learning



강화학습과 Function approximation

2. 함수 형태로 근사한 큐함수