Reinforcement Learning

Introduction to Reinforcement Learning (Day 1)

Jeiyoon Park

Department of Computer Science and Engineering

8/10/2020

k4ke@korea.ac.kr

Outline

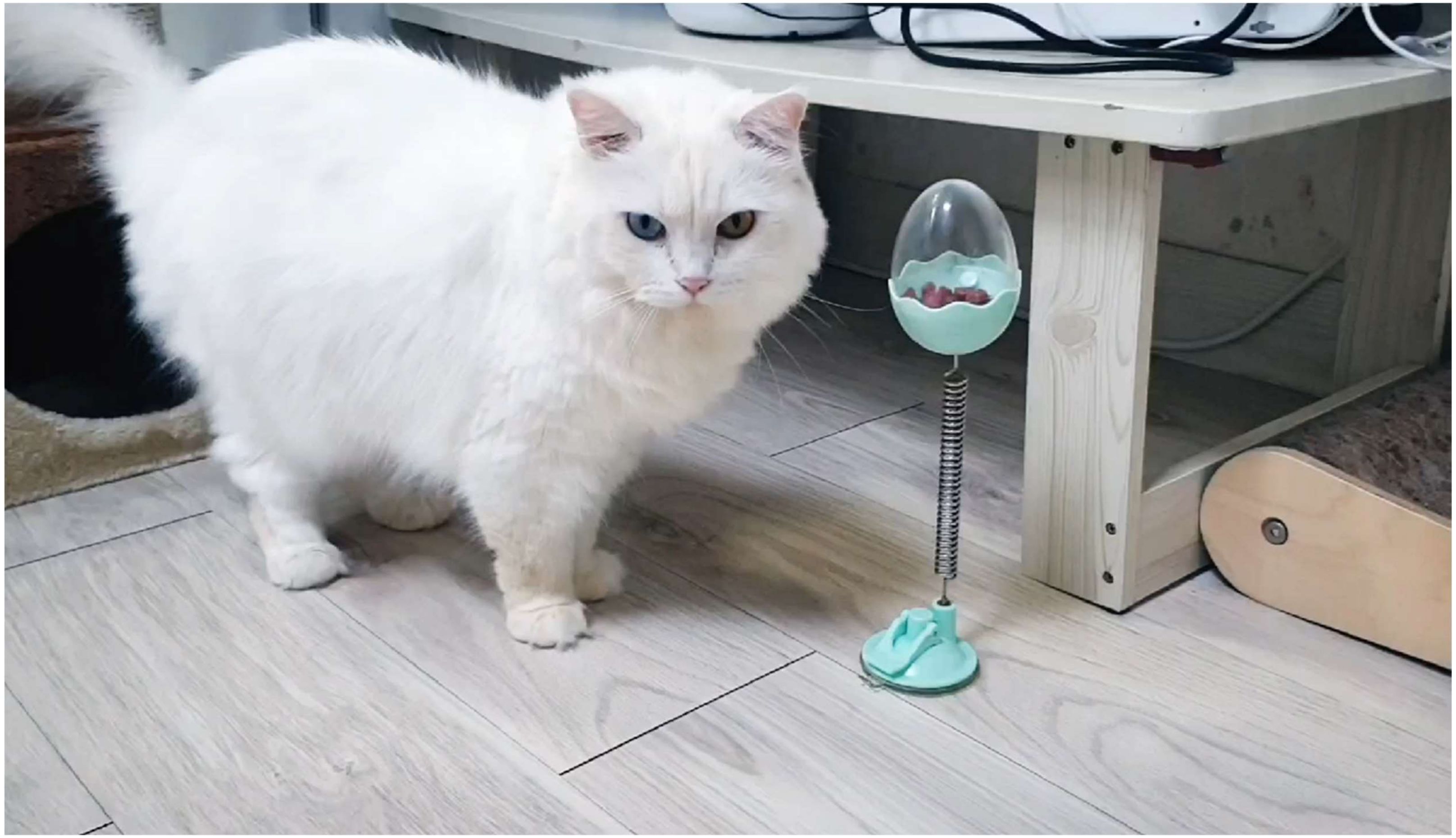
- Overview
- Basics
- Dynamic Programming
- Q-Learning
- Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Applications

Outline

- Overview
- Basics
- Dynamic Programming
- Q-Learning
- Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Applications

Overview

1. 강화학습이란?



https://www.youtube.com/watch?v=3Sxgrahdeyw

Overview

1. 강화학습이란?

강화학습이란 에이전트(Agent)가 환경(Environment)으로 부터 얻어지는 보상정보(Reward)를 통해 좋은 행동을 점점 더 많이 하도록 하는 학습 방법이다.

Overview

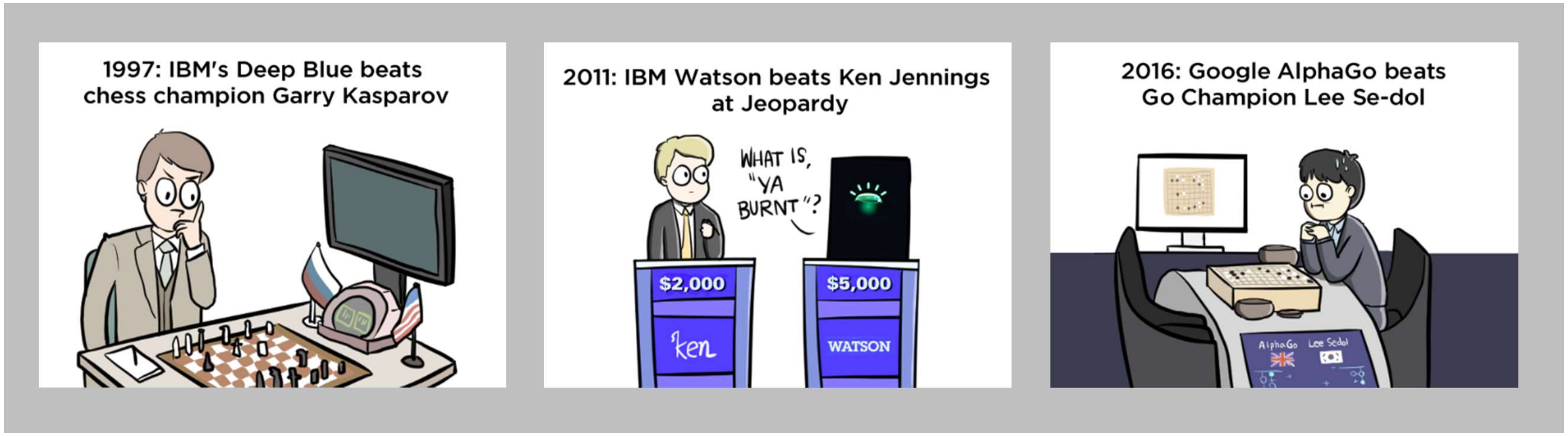
2. 강화학습의 장점

(1) 기존의 방식으로 풀기 어려웠던 복잡한 문제들을 해결할 수 있다.

(2) 학습 데이터가 없어도 경험으로부터 학습할 수 있다.

(3) 인간의 학습방법과 굉장히 유사하다. 따라서 학습이 직관적이고 완벽함을 추구하며 현실세계의 문제들을 해결할 수 있다.

실제로 복잡한 문제들을 해결할때 사람보다 더 잘할 수 있다.



http://www.theverge.com/2016/3/11/11208078/lee-se-dol-go-google-kasparov-jennings-ai

Overview

진단 평가

1) 강화학습이란?
2) 에이전트란?
3) 환경이란?
4) 보상이란?
5) 관찰결과란? 관찰결과의 역할은?

Outline

- Overview
- Basics
- Dynamic Programming
- Q-Learning
- Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Applications

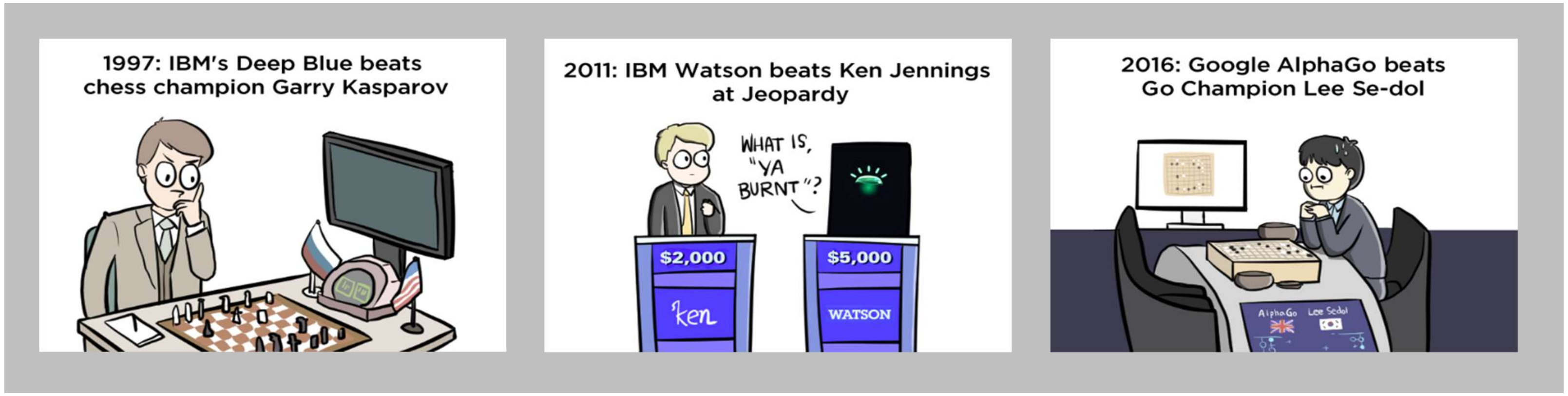
Basics (1/6)

1. Markov Decision Process

- Markov Decision Process(MDP)란 강화학습 같은 순차적으로 행동을 결정하는 문제를 정의할 때 사용하는 방법

- MDP의 구성요소는 크게 다섯가지임. 상태(state), 행동(action), 보상함수(reward), 상태변환확률(state transition probability), 감가율(discount factor)

- **모든 강화학습은 MDP를 "사용자"가 정의하는 것 부터 시작**



Basics (1/6)

1. Markov Decision Process

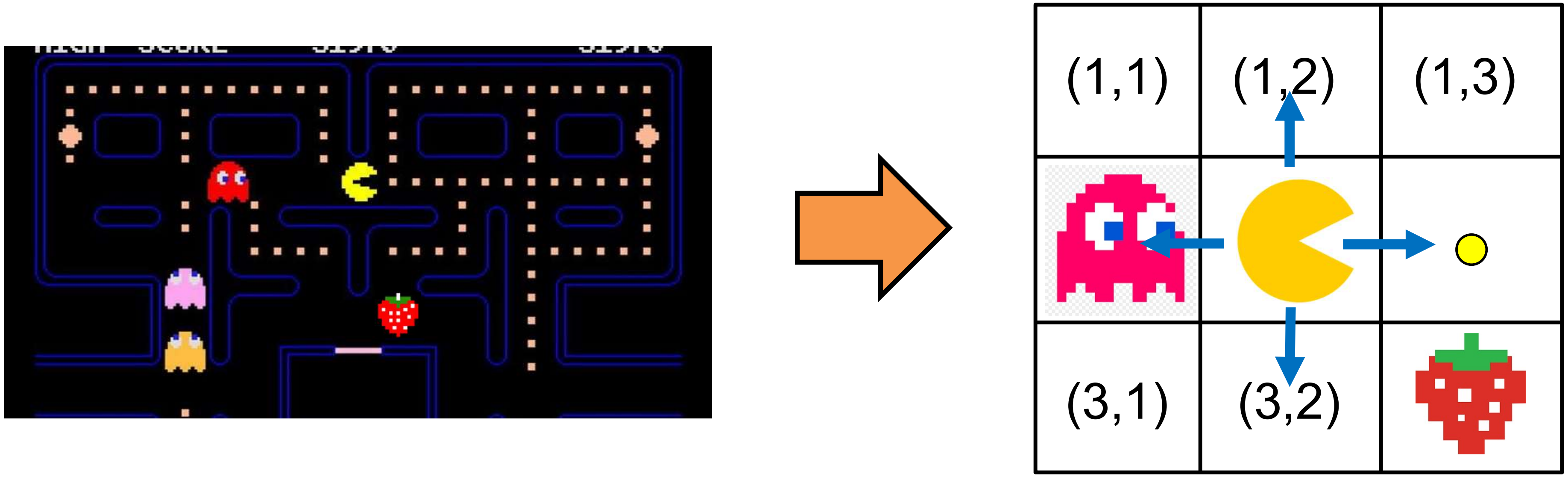
- ex) Pac Man

https://www.youtube.com/watch?v=QilHGSYbjDQ

Basics (1/6)

1. Markov Decision Process

- ex) Pac Man



- 상태(state) : $S = \{(1,1), \dots, (3,3)\}$

- 행동(action) : $A = \{\leftarrow, \uparrow, \rightarrow, \downarrow\}$

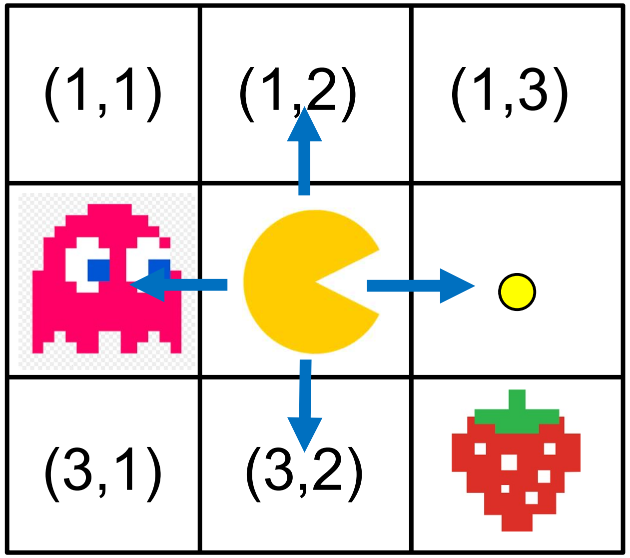
$$\boxed{R_s^a = E[R_{t+1} \mid S_t = s, A_t = a]}$$

- 보상함수(reward) : $R = \{+1 \;\; or \; +10 \;\; or \; -1\}$

Basics (1/6)

1. Markov Decision Process

- ex) Pac Man



\- 상태변환확률(state transition probability) : 팩맨이 (2, 2)에 있을때, (2, 3)에 있는 공을 먹으러 갈 확률

\- 감가율(discount factor) : 보상이 모두 똑같다면 지금 먹은 공과 시간이 흐른 뒤 먹은 공을 구분할 수 없음. 또한 딸기(+10)를 먹는 것과 공을 열개(+10)먹는 것을 구분할 수 없음. 따라서 보상에 가감율을 곱해줌.

Basics (1/6)

1. Markov Decision Process

- 상태(state) : 에이전트가 관찰 가능한 상태의 집합

- 행동(action) : 에이전트가 특정 상태에서 할 수 있는 행동의 집합

- 보상함수(reward) : 환경이 에이전트에게 주는 정보. 에이전트가 학습할 수 있는 유일한 정보

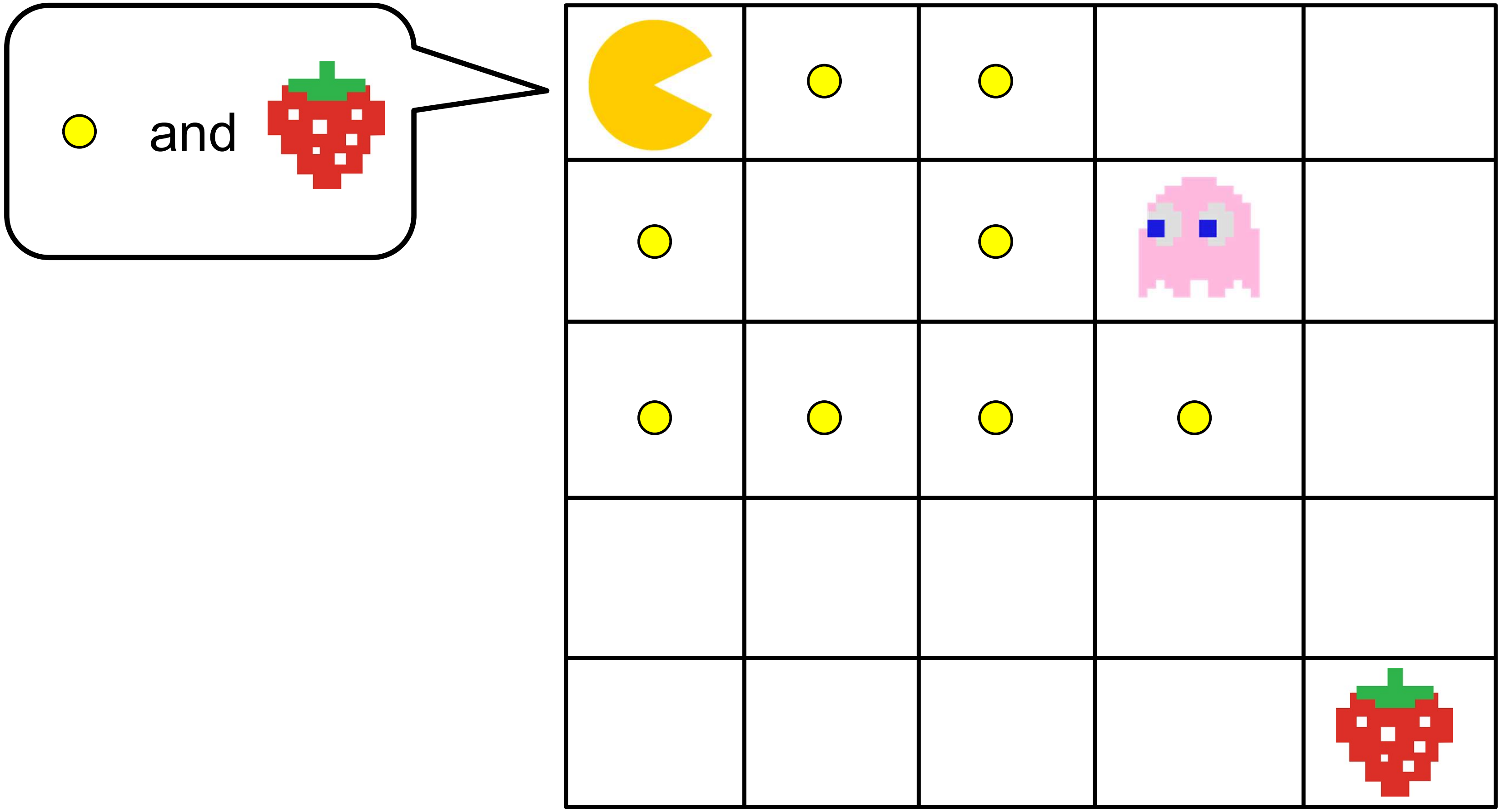
- 상태변환확률(state transition probability) : 에이전트가 어떠한 상태 s에서 행동 a를 해서 다른 상태 s'에 도달할 확률

- 감가율(discount factor) : 같은 보상이면 나중에 받을 수록 가치가 떨어짐. 이를 수학적으로 표현하기 위한 개념

Basics (2/6)

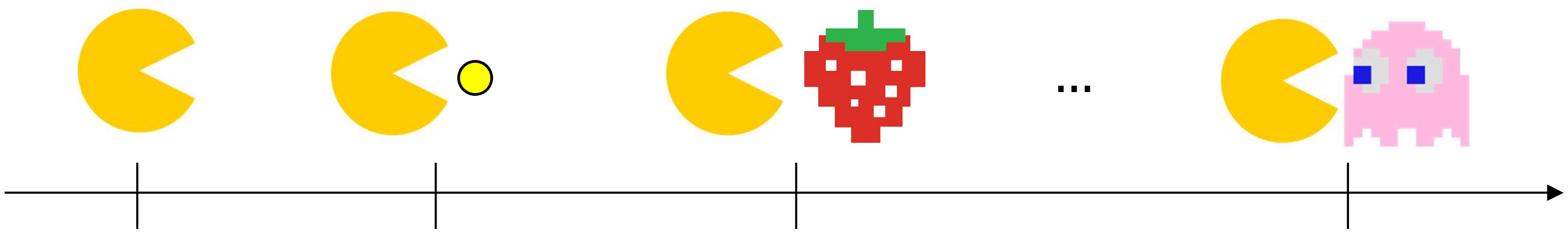
2. Value Function

에이전트(팩맨) 입장에서는 어떤 행동을 하는 것이 좋은지 어떻게 알까? 아직 받지 않은 보상들을 어떻게 고려하고 행동할 수 있을까?



Basics (2/6)

2. Value Function



t	$t+1$	$t+2$	$\cdots$	$t+k$
	R_{t+1}	$\gamma^1 \times R_{t+2}$	$\cdots$	$\gamma^{k-1} \times R_{t+k}$

- 반환값(return): 에이전트가 실제로 환경을 탐험하며 받은 보상

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

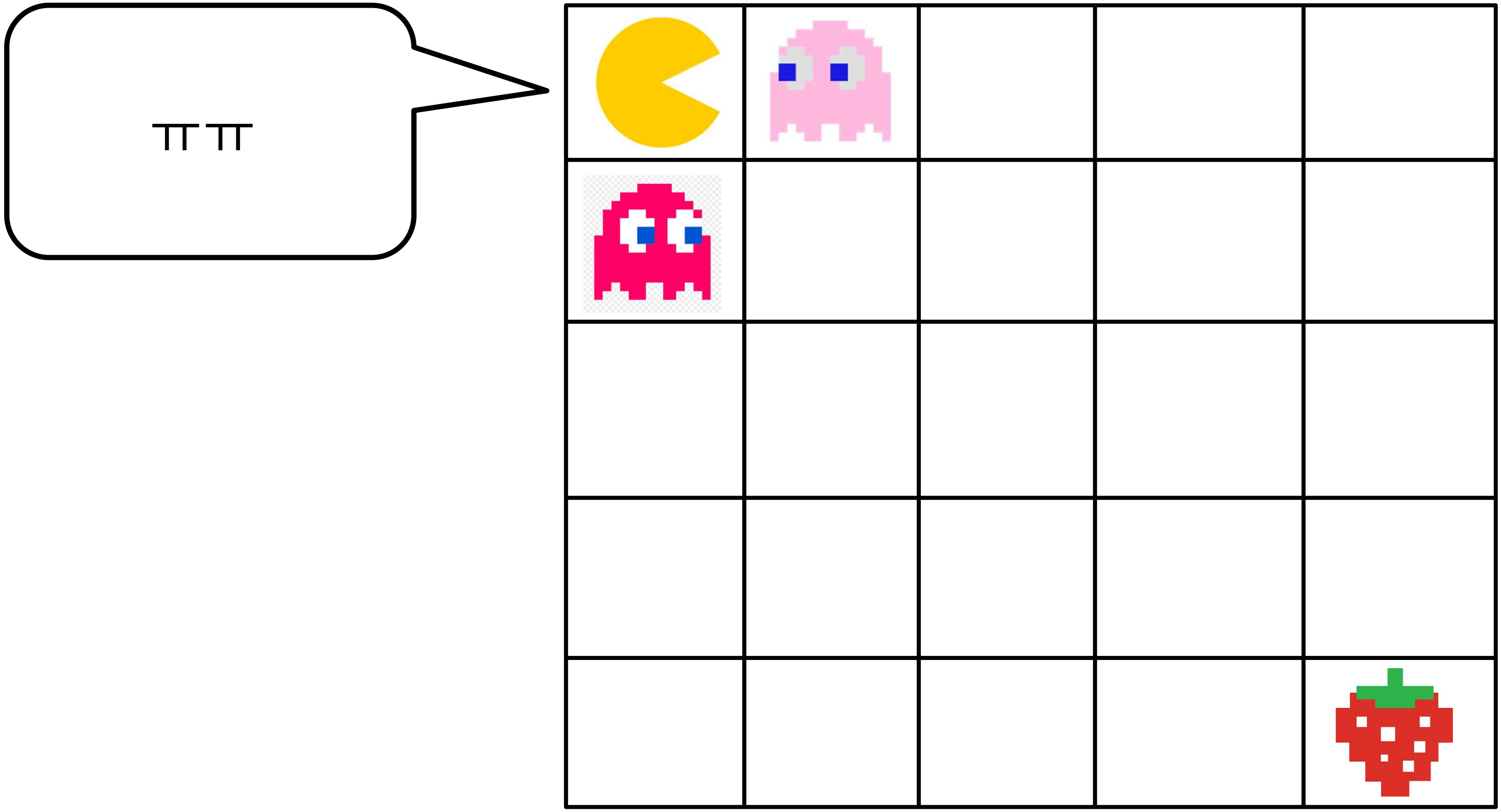
- 가치함수(value function): 에이전트가 얼마의 보상을 받을 것인지에 대한 기댓값. 매 시행마다 값이 다르기 때문에 기댓값 사용.

$$v(s) = E[G_t \mid S_t = s] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

Basics (2/6)

2. Value Function

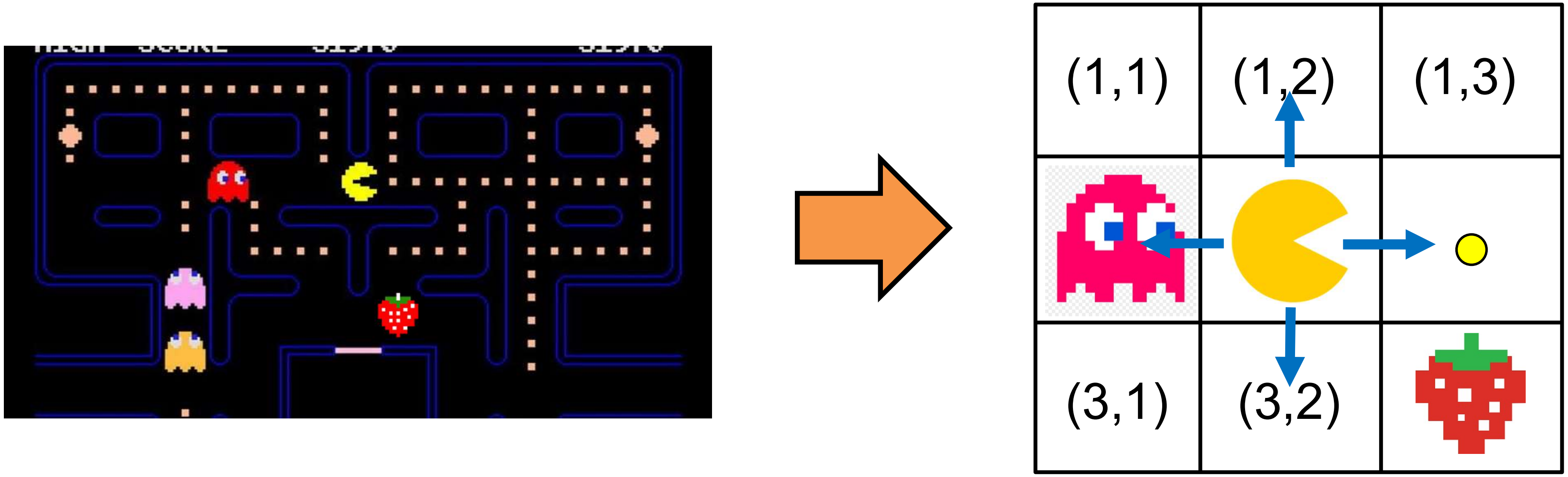
즉, 가치함수란 어떠한 상태에 있을때 이 상태에 있으면 앞으로 얼마의 보상을 받을 지에 대한 기댓값이다.



Basics (3/6)

3. Policy

- 정책(policy)이란 모든 상태에서 에이전트가 할 행동



$$\pi(a|s) = P[A_t = a \mid S_t = s]$$

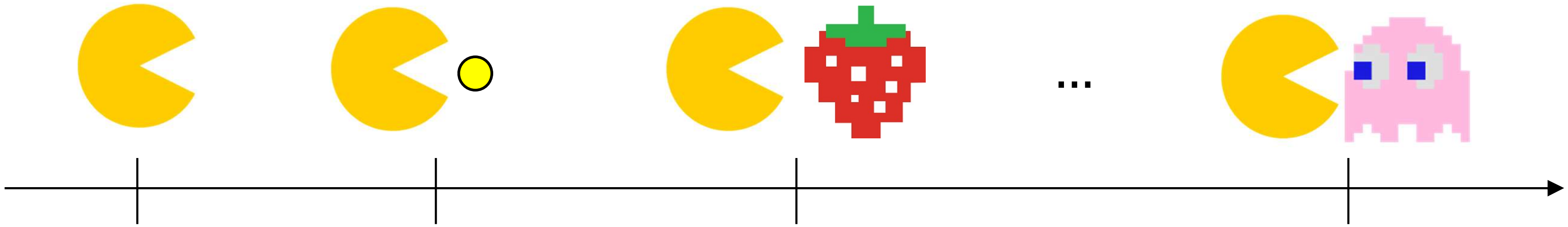
- ex) $\pi\big(a = \text{왼쪽} \mid s = (2,2)\big) = 0.1$

  $\pi(a = \text{오른쪽} \mid s = (2,2)) = 0.9$

Basics (4/6)

4. Bellman Expectation Equation

벨만 기대 방정식이란 현재 상태의 가치함수와 다음상태의 가치함수 사이의 관계를 말해주는 방정식이다.

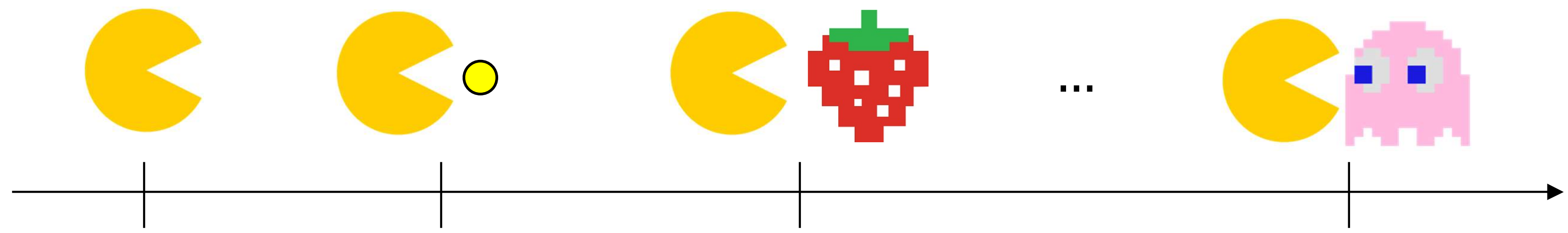


t	$t+1$	$t+2$	⋯	$t+k$
	R_{t+1}	$\gamma^1 \times R_{t+2}$	⋯	$\gamma^{k-1} \times R_{t+k}$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \ldots$$

$$v(s) = \ E[R_{t+1} + \ \gamma v(S_{t+1}) | \ S_t = s]$$

4. Bellman Expectation Equation



t	$t+1$	$t+2$	…	$t+k$
	R_{t+1}	$\gamma^1 \times R_{t+2}$	…	$\gamma^{k-1} \times R_{t+k}$

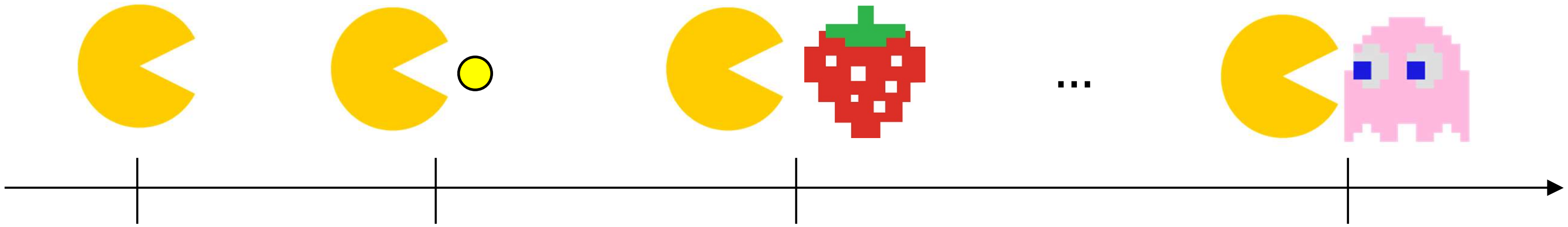
$$
\begin{aligned}
v(s) &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s] \\
&= E[R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \ldots) \mid S_t = s] \\
&= E[R_{t+1} + \gamma(G_{t+1}) \mid S_t = s] \\
&= E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]
\end{aligned}
$$

반환값이긴 하지만 사실 에이전트가
실제로 받은 보상이 아직은 아님.
따라서 가치함수 형태로 나타낼 수 있음

$\therefore \; v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$: 현재상태의 가치함수와 다음상태의 가치함수 사이의 관계를 말해줌

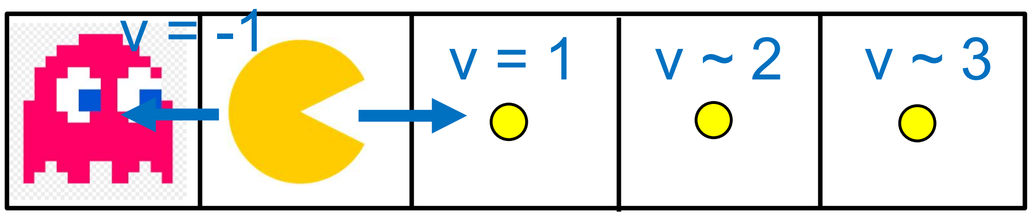
Basics (4/6)

4. Bellman Expectation Equation



t	$t+1$	$t+2$	…	$t+k$
	R_{t+1}	$\gamma^1 \times R_{t+2}$	…	$\gamma^{k-1} \times R_{t+k}$

$$\therefore \; v(s) = E[R_{t+1} + \gamma v(S_{t+1}) | \, S_t = s]$$



Basics (4/6)

4. Bellman Expectation Equation

벨만 기대방정식을 계산할때는 상태에서의 가치함수 뿐만 아니라 정책(에이전트가 어떤 행동을 할지에 대한 확률) 또한 고려해야 함.

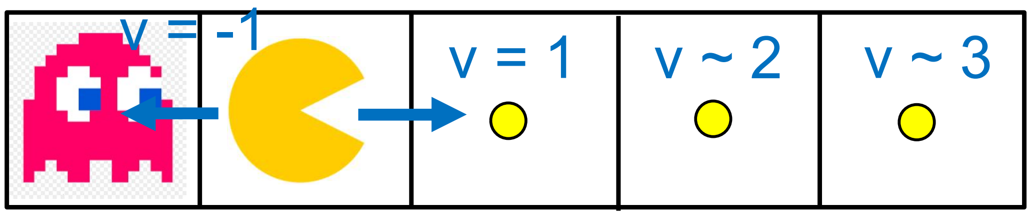
- ex) $\pi\left(a = \text{왼쪽} \mid s = (2,2)\right) = 0.1$

  $\pi(a = \text{오른쪽} \mid s = (2,2)) = 0.9$

$$\pi(a|s) = P[A_t = a \mid S_t = s]$$

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

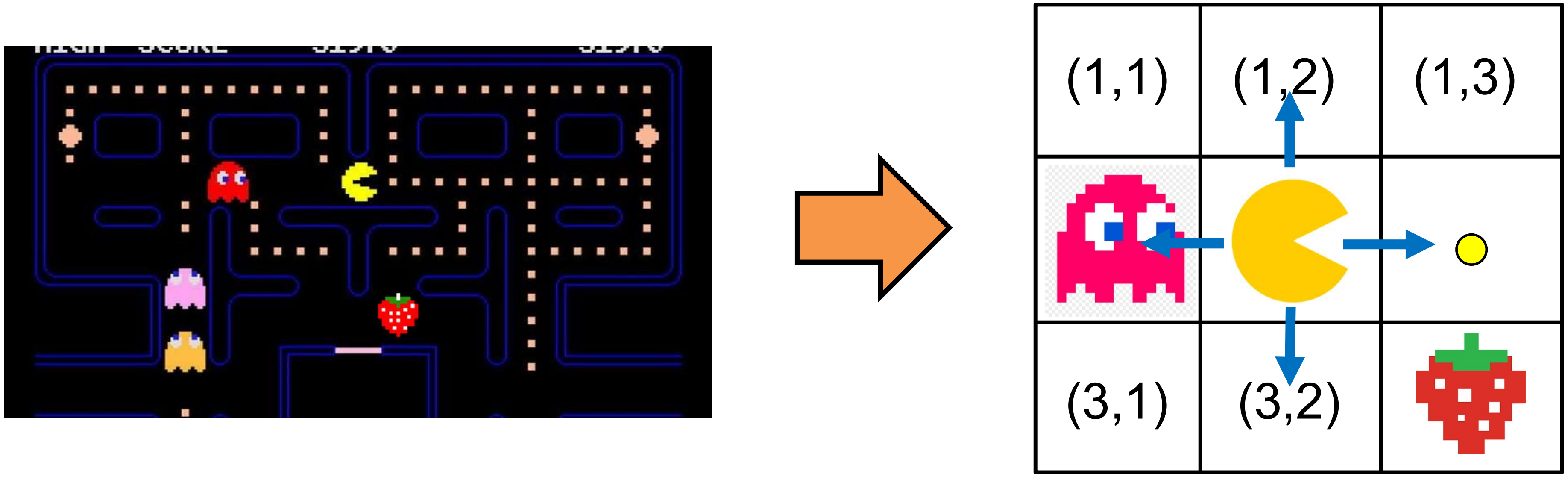
$$\therefore \quad v_\pi(s) = E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$$



Basics (5/6)

5. Q-Function

- 가치함수(value function)는 어떤 ‘상태’에 대한 보상의 기댓값
- 큐함수(q-function)는 어떤 ‘상태’에서 어떤 ‘행동’이 얼마나 좋은지 알려주는 함수. 행동 가치함수라고도 함.
- 따라서 큐함수는 상태와 행동이라는 두가지 변수를 가짐



Basics (5/6)

5. Q-Function

- 어떠한 상태에 있을때 어떠한 보상을 얻을 지 아는 것도 중요하지만 어떠한 행동을 했을때 어떠한 보상을 얻을 지, 즉 행동 가치함수의 기댓값을 아는 것도 중요하다.

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

$$\pi(a|s) = P[A_t = a \mid S_t = s]$$

- 특히 각 행동에 대해 가치함수를 계산하여 정보를 가져올 수 있으면 굳이 상태에 대한 가치함수를 계산할 필요가 없다.

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

Basics (6/6)

6. Bellman Optimality Equation

- 그렇다면 가치함수의 역할은 뭘까? 정책을 정하고 그 정책을 따라갔을때 받는 보상들의 합인 가치함수로 더 좋은 정책을 찾아내는 것
- 그렇다면 최적의 가치함수는 어떻게 구할 수 있을까?

$$v_*(s) = \max_{\pi}[v_\pi(s)]$$

$$= E\left[R_{t+1} + \gamma \max_{a'} q_*(s_{t+1}, a') \middle| S_t = s,\ A_t = a\right]$$

Basics (6/6)

진단 평가

1) MDP의 다섯가지 요소는?
2) 가치함수란? 필요한 이유는?
3) 정책이란?
4) 벨만 기대 방정식은? 의미는?
5) 큐함수란?
6) 최적의 가치함수는 어떻게 구할까? 필요한 이유는?

Outline

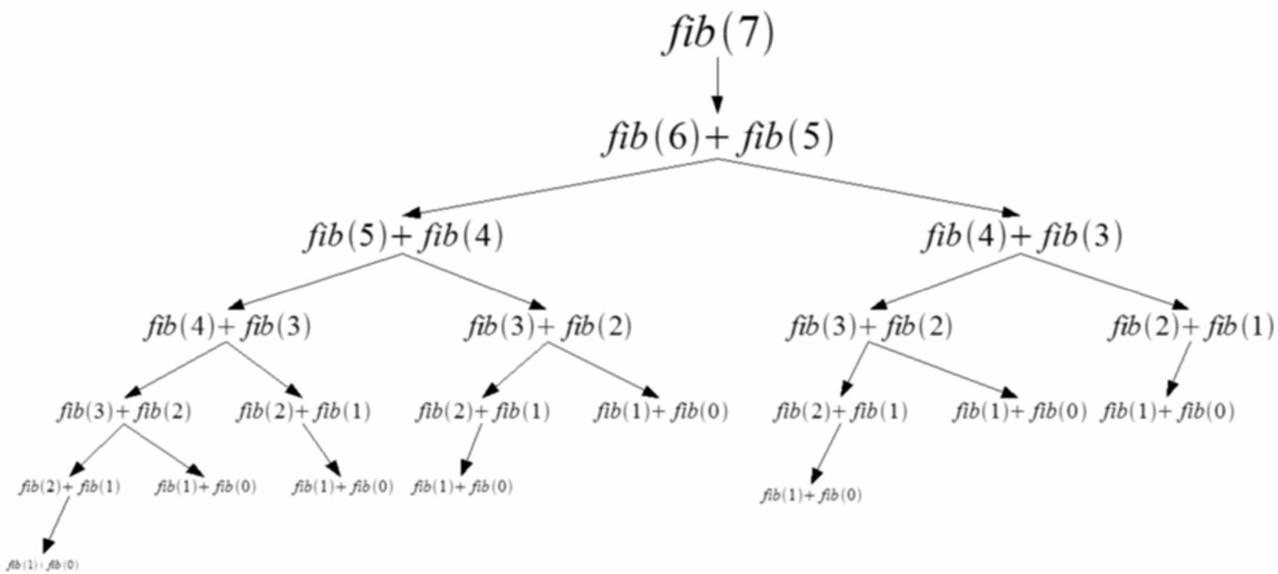
- Overview
- Basics
- Dynamic Programming
- Q-Learning
- Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Applications

Dynamic Programming

1. 동적 프로그래밍 이란?

- ~~동적~~ → 기억하기
- ~~프로그래밍~~ → (순차적 프로세스에 대한) 테이블 만들기
- 큰 문제를 한 번에 해결하기 힘들때 작은 여러개의 문제로 나누어서 푸는 기법

e.g.) 피보나치 수열



Dynamic Programming

2. 그렇다면 무엇을 어떻게 작은문제로 나누어서 풀까?

(1) 우리가 구하고 싶은것: 각 상태의 가치함수(i.e. $v_\pi(s_n)$)

(2) 이때, 모든 상태에 대해 가치함수를 구하고 iteration을 돌며 각 상태에 대한 가치함수를 업데이트 한다.

	s_2	s_3	s_4	s_5
s_6	s_7	s_8	s_9	s_{10}
s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
s_{21}	s_{22}	s_{23}	s_{24}	

각각의 모든 상태에 대해 진짜 가치함수 $v_\pi(s)$ 를 구하는 것이 목표!

Dynamic Programming

3. 가치함수가 구해지는 과정 (벨만방정식 푸는 과정)

	s_2	s_3	s_4	s_5
s_6	s_7	s_8	s_9	s_{10}
s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
s_{21}	s_{22}	s_{23}	s_{24}	

iteration	가치함수 (v_π)
1	$v_0(s_1), \ldots, v_0(s_{25})$

$$v(s) = E[G_t \mid S_t = s] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$$

Dynamic Programming

3. 가치함수가 구해지는 과정 (벨만방정식 푸는 과정)

	s_2	s_3	s_4	s_5
s_6	s_7	s_8	s_9	s_{10}
s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
s_{21}	s_{22}	s_{23}	s_{24}	

iteration	가치함수 (v_π)
1	$v_0(s_1), \dots, v_0(s_{25})$
2	$v_1(s_1), \dots, v_1(s_{25})$

$$v(s) = E[G_t \mid S_t = s] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

Dynamic Programming

3. 가치함수가 구해지는 과정 (벨만방정식 푸는 과정)

	s_2	s_3	s_4	s_5
s_6	s_7	s_8	s_9	s_{10}
s_{11}	s_{12}	s_{13}	s_{14}	s_{15}
s_{16}	s_{17}	s_{18}	s_{19}	s_{20}
s_{21}	s_{22}	s_{23}	s_{24}	

iteration	가치함수 (v_π)
1	$v_0(s_1), \ldots, v_0(s_{25})$
2	$v_1(s_1), \ldots, v_1(s_{25})$
…	…
k	$v_\pi(s_1), \ldots, v_\pi(s_{25})$

$$v(s) = E[G_t \mid S_t = s] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$$

Dynamic Programming

4. 정책 이터레이션: 정책 평가와 정책 발전

- 핵심은 다이나믹 프로그래밍으로 벨만방정식을 풀어서 가치함수를 구하는 과정을 이해하는것!
- 이 과정을 정책 이터레이션이라고 함
- 처음에는 무작위 행동을 한 후 이터레이션을 돌며 가치함수를 최적화 시켜나감
- 위와 같은 최적화 과정은 정책 평가(Policy Evaluation)와 정책 발전(Policy Improvement)으로 나누어짐

순차적인 행동을 결정해야되는 문제

State
Action
Reward
Transition
Discount Factor

MDP

정책 이터레이션 → 벨만 기대 방정식

가치 이터레이션 → 벨만 최적 방정식

Dynamic Programming

5. 정책 평가 (from 정책 이터레이션)

- 어떤 정책(Policy)이 있을때 그 정책을 정책 평가를 통해 얼마나 좋은지 판단하고 그 결과를 기준으로 더 좋은 정책으로 발전시킴
- 그렇다면 정책을 어떻게 평가할 수 있을까?
- 그 근거는 바로 다이나믹 프로그래밍으로 구한 각각의 상태에 대한 가치함수가 된다!

$$v(s) = E[G_t \mid S_t = s] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s]$$

- 위 수식은 각각의 상태에 대한 아주 먼 미래까지 고려해야하기 때문에 계산량이 급격하게 늘어남
- 하지만 다이나믹 프로그래밍을 통해 이터레이션을 돌며 가치함수를 최적화 할 수 있음!

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s]$$

Dynamic Programming

5. 정책 평가 (from 정책 이터레이션)

- 벨만 “기대”방정식은 가치함수에 대한 기댓값 형태(확률곱)으로 나타낼 수 있다.

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

$$= \sum_{a \in A} \pi(a|s)(R_{t+1} + \gamma v_\pi(s'))$$

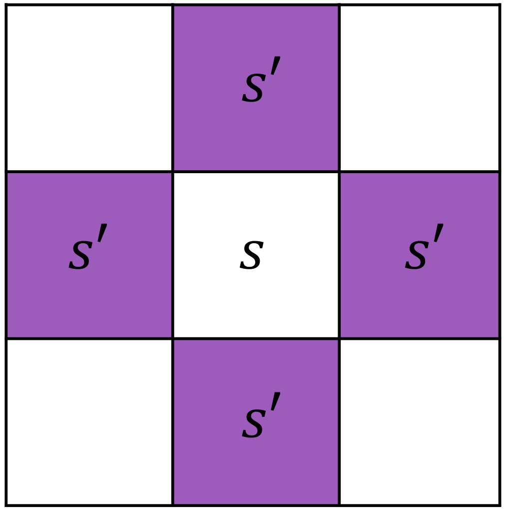
$$\therefore \; v_\pi^{n+1}(s) = \sum_{a \in A} \pi(a|s)(R_{t+1} + \gamma v_\pi^{n}(s'))$$

Dynamic Programming

5. 정책 평가 (from 정책 이터레이션)

(1) 현재 상태 s 에서 갈 수있는 다음 상태 s' 에 대한 가치함수를 불러옴 ($v_\pi^{n+1}(s')$, 보라색 부분 중 하나)

(2) 가치함수에 감가율을 곱하고 그 상태로 가는 것에 대한 보상을 더함 ($R_{t+1} + \gamma v_\pi^n(s')$)



Iteration n

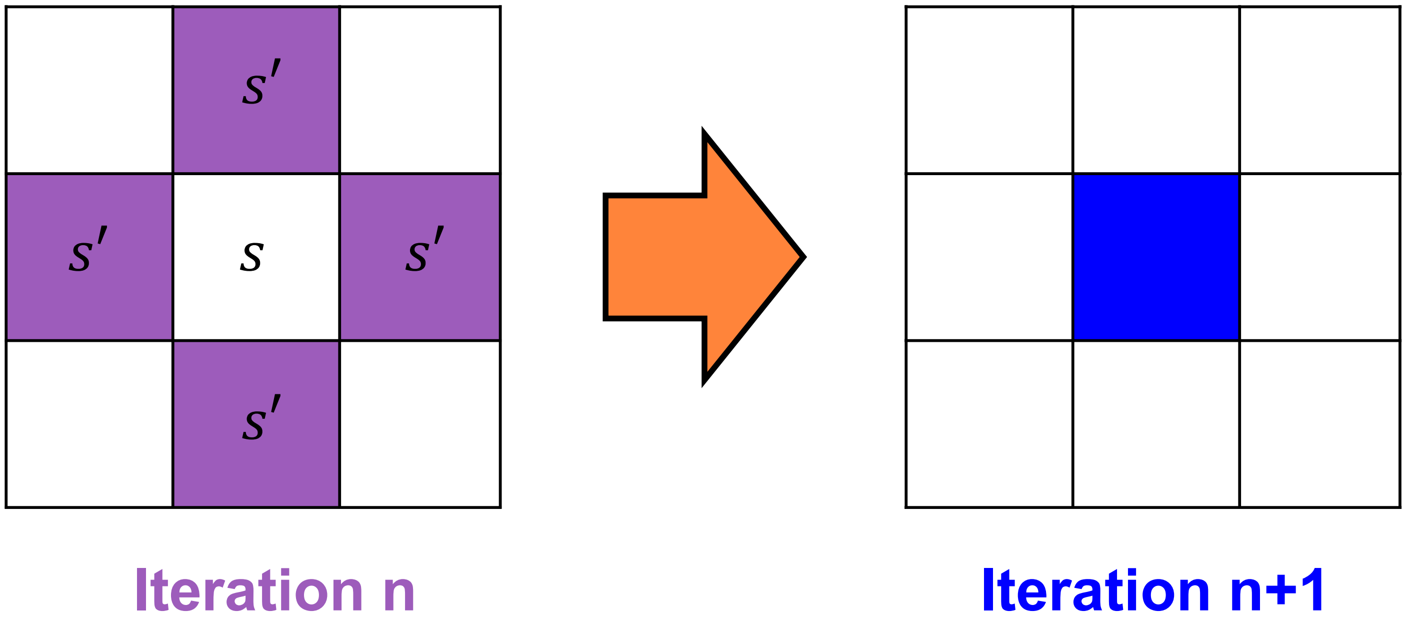
Dynamic Programming

5. 정책 평가 (from 정책 이터레이션)

(3) (2)에서 구한 값에 그 행동을 취할 확률(정책)을 곱하여 기댓값 형태로 나타냄 ($\pi(a|s)(R_{t+1} + \gamma v_\pi^n(s'))$)

(4) (3)을 모든 가능한 행동에 대해 반복하고 그 값을 더함

($\sum_{a \in A} \pi(a|s)(R_{t+1} + \gamma v_\pi^n(s'))$). 결과를 n+1 가치함수로 사용



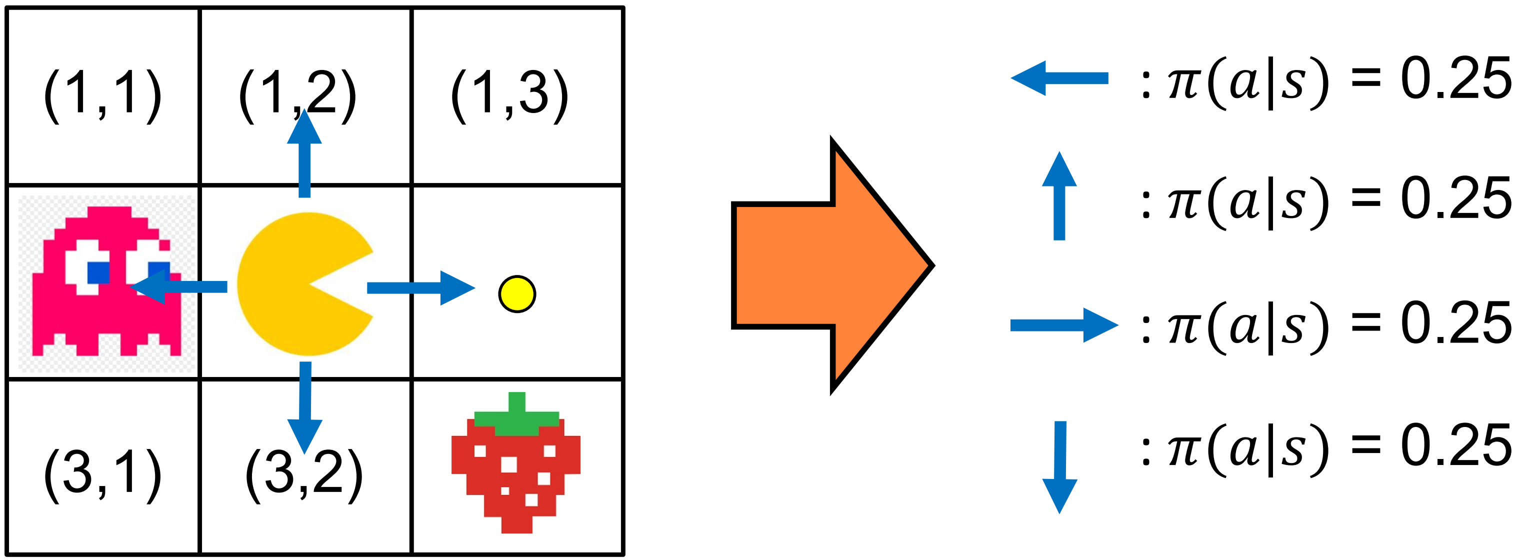
Dynamic Programming

6. 정책 발전 (from 정책 이터레이션)

(1) 정책 평가를 바탕으로 어떻게 정책을 발전시킬 수 있을까?

(2) 가치함수 최적화 전에는 무작위 행동을 하고 점점 가치함수가 높은 행동을 더 많이 하도록 학습함

(3) 가장 유명한 방법중 하나인 탐욕 정책 발전(Greedy Policy Improvement)를 사용하려고 함



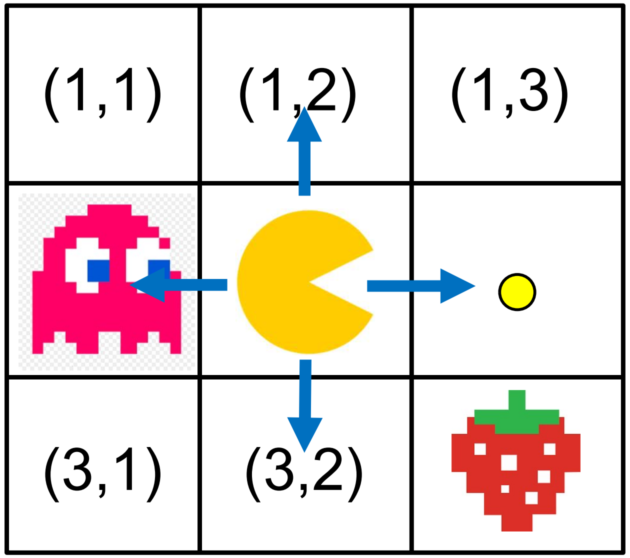
Dynamic Programming

6. 정책 발전 (from 정책 이터레이션)

(4) 정책에 대한 평가를 거치면 큐함수(Q-function)을 이용하여 행동에 대한 가치함수를 알 수 있음

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

$$= R_{t+1} + \gamma v_\pi(S_{t+1})$$



← : $\pi(a|s)$ = 0.25

↑ : $\pi(a|s)$ = 0.25

→ : $\pi(a|s)$ = 0.25

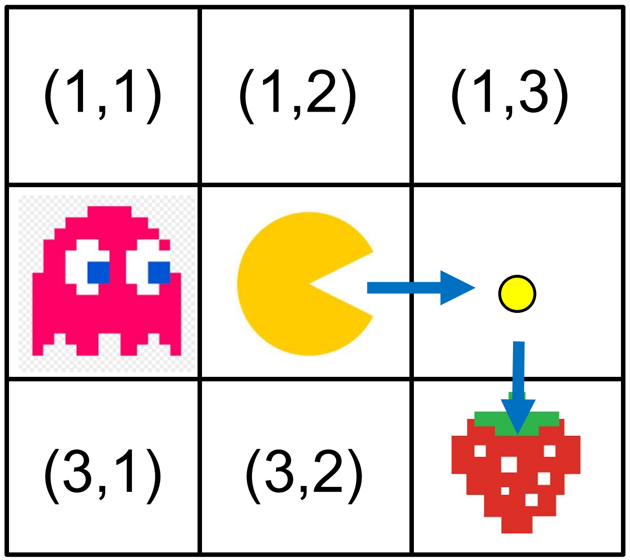
↓ : $\pi(a|s)$ = 0.25

Dynamic Programming

6. 정책 발전 (from 정책 이터레이션)

(5) 상태 s에서 큐함수들을 비교하여 가장 큰 큐함수를 가지는 행동, 즉 행동 가치함수값이 가장 높은 행동을 선택함. 따라서 더 높은 보상을 주는 행동을 반복하도록 학습됨.

$$\pi'(s) = \underset{a \in A}{argmax}\, q_\pi(s, a)$$



← : $\pi(a|s)$ = 0.01

↑ : $\pi(a|s)$ = 0.05

→ : $\pi(a|s)$ = 0.7

↓ : $\pi(a|s)$ = 0.24

Dynamic Programming

7. 가치 이터레이션

(1) 가치 이터레이션(Value iteration)이란 현재의 가치함수가 최적은 아니지만 최적이라는 전제하에 각 상태에 대한 가치함수를 업데이트 하는 방법 (벨만최적방정식 사용)

(2) 벨만 기대방정식은 기댓값 형태이기 때문에 정책을 고려했었음. 하지만 벨만 최적방정식에서는 현재 상태에서 가능한 최고의 가치함수 값을 고려하면 됨.

$$v_{n+1}(s) = \max_{a \in A}(R_{t+1} + \gamma v_n(s'))$$

Dynamic Programming

8. 동적 프로그래밍의 한계

(1) 계산 복잡도 (i.e. 5x5 그리드 월드가 아니라 nxn이라면?)

(2) 차원의 저주 (i.e. 그리드 월드처럼 2차원이 아니라 n차원이라면?)

(3) 환경에 대한 완벽한 정보를 알아야 한다 (우리는 실제로 세상을 탑뷰로 바라보며 모든 환경에 대한 정보를 인지하고 있는가?)

Dynamic Programming

진단 평가

1) 동적 프로그래밍이란? 장점은?
2) 동적 프로그래밍으로 풀고자 하는 문제는?
3) 가치함수가 업데이트 되는 과정을 설명할 수 있는가?
4) 정책 이터레이션이란? 정책 평가와 정책 발전은 각각 무엇을 의미하는가?
5) 정책 이터레이션과 가치 이터레이션의 차이는?
6) 동적 프로그래밍의 한계는?

Outline

- Overview
- Basics
- Dynamic Programming
- Q-Learning
- Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Applications

Q-Learning

1. 학습 방법

• 사람은 바둑을 어떻게 둘까? 다이나믹 프로그래밍때 처럼 한칸한칸 모든 경우의 수를 매 턴마다 생각하면서 둘까?

•많은 경우가 일단 해보고 복기를 하고 복기 내용을 바탕으로 학습하여 더욱 잘해지는 과정을 반복한다.

• 강화학습은 사람의 학습방법처럼 겪은 경험으로 부터 가치함수를 업데이트 함

2016: Google AlphaGo beats
Go Champion Lee Se-dol

Q-Learning

2. 예측과 제어

- 에이전트는 환경과 상호작용을 통해 주어진 정책에 대한 가치함수를 학습할 수 있음. 이를 예측(prediction)이라고 함

(앞에서 얘기했던 정책 평가에 해당)

e.g.) 몬테카를로 예측, 시간차 예측

- 또한 가치함수를 토대로 정책을 끊임없이 발전시켜 나가 최적의 정책을 학습할 수 있음. 이를 제어(control)라고 함

(앞에서 얘기했던 정책 발전에 해당)

e.g.) SARSA

Q-Learning

2. 예측과 제어

- 즉, 예측은 현재 정책을 따랐을 때 참 가치함수를 구하는 과정.

= '정책 평가' 라고도 함. 왜냐하면 이 정책을 따랐을때 보상의 합인 가치함수가 얼마인지 나오고 그거에 따라 정책이 좋은지 나쁜지 평가하기 때문.

- 예측의 결과로 정책을 발전 시키는 것을 제어라고 함.

= '정책 발전' 이라고도 함.

- 정책 평가와 정책 발전을 번갈아 가며 진행하는 것을 통해 학습함.

- 그렇다면 예측(가치함수를 구하는 과정)은 어떻게 이루어질까?

Q-Learning

3. 몬테카를로 예측

(1) 몬테카를로 한줄 요약: 일단 해보자

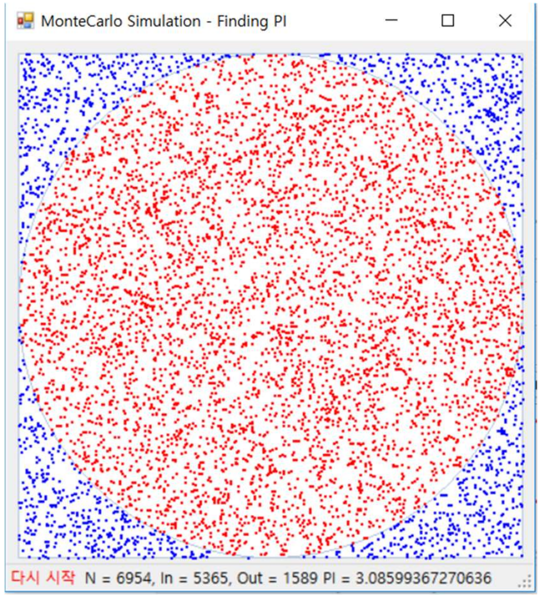
ex) 원의넓이 = πr^2

하지만, 원의 넓이 공식을 모른다면?

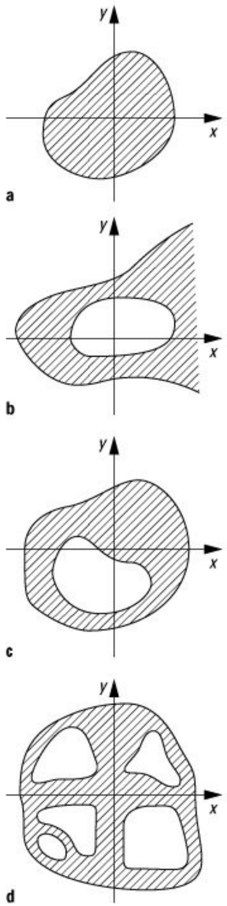
Q-Learning

3. 몬테카를로 예측

(1) 몬테카를로 한줄 요약: 일단 해보자



ex) 원의넓이 $= \pi r^2$



하지만, 원의 넓이 공식을 모른다면?

(혹은 방정식이 원이아니라면?)

→ 점들을 샘플링, 붉은 점의 갯수를 전체 점의 갯수로 나눔

→ 예를들어 전체 점이 6954개 이고 빨간색 점이 5365개라면 원의 넓이는 5365/6954 = 0.7714로 계산

→ 샘플링 한 점의 갯수가 무한대로 가면 원의 넓이에 수렴

Q-Learning

3. 몬테카를로 예측

(2) 샘플링과 몬테카를로 예측

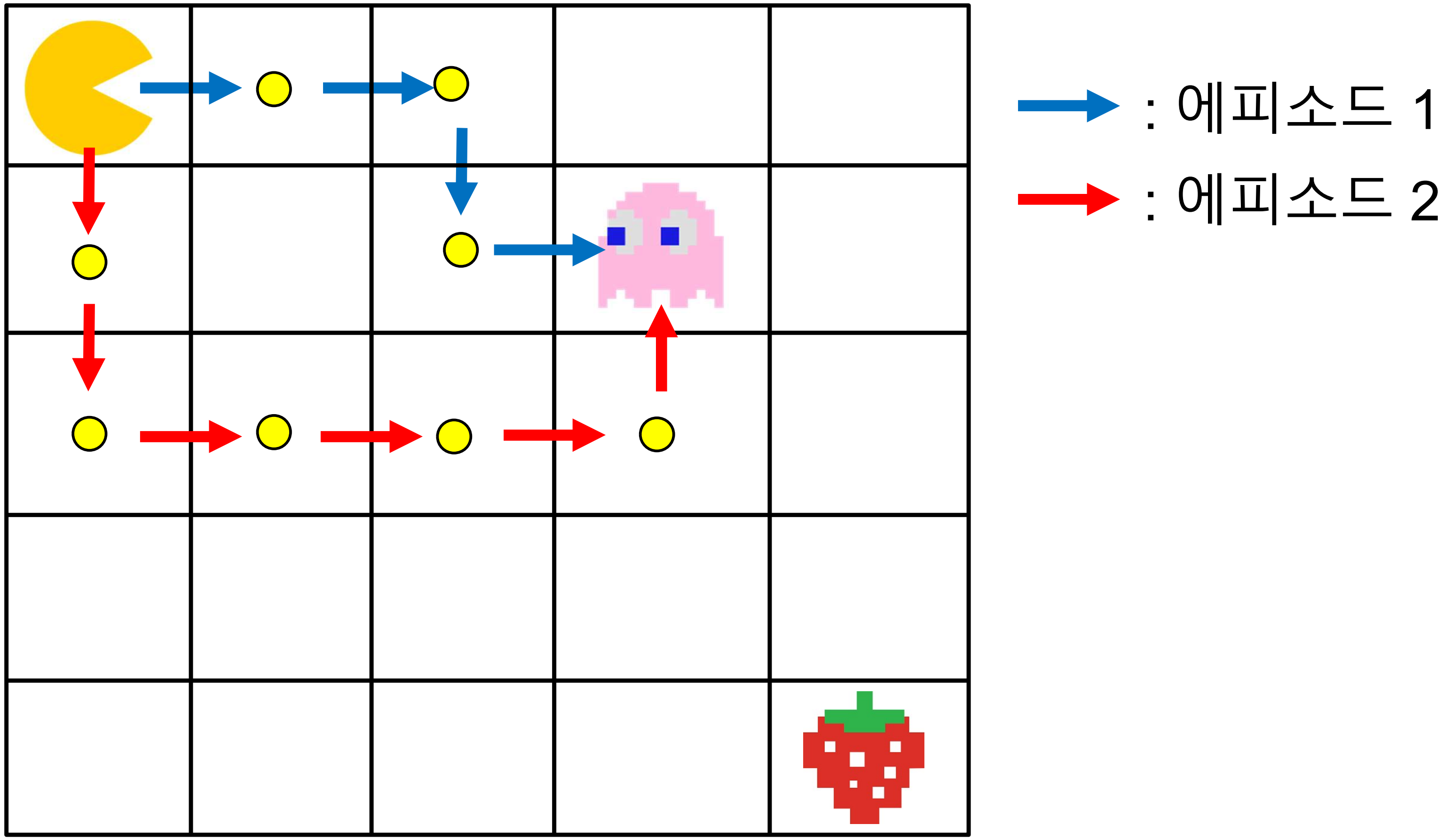
- 앞에서 본 것처럼 가치함수에 대한 모델을 모르는 경우에도 몬테카를로 예측을 통해 가치함수를 추정하는 것이 가능함
- 가치함수를 추정할때 에이전트가 환경과 상호작용한 한 에피소드를 샘플링함 (점 하나 뿌리기)
- 여러번의 샘플링을 통해 가치함수의 기댓값을 계산하지 않고 샘플들의 평균으로 가치함수를 최적화 하려면 (i.e. 예측하려면) 어떻게 해야할까?

$$v(s) = E[G_t \mid S_t = s] = E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s]$$

Q-Learning

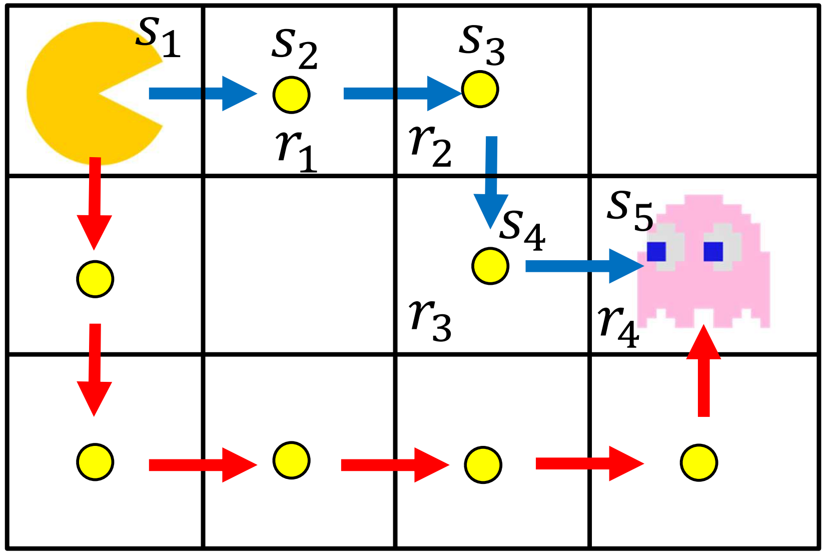
3. 몬테카를로 예측

(3) 가치함수를 추정할 때는 에이전트가 환경에서 한 에피소드를 진행 한 것을 샘플링함. (원의 넓이 구할 때 점 하나 뿌리기)



Q-Learning

3. 몬테카를로 예측



$$G(s_1) = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4$$

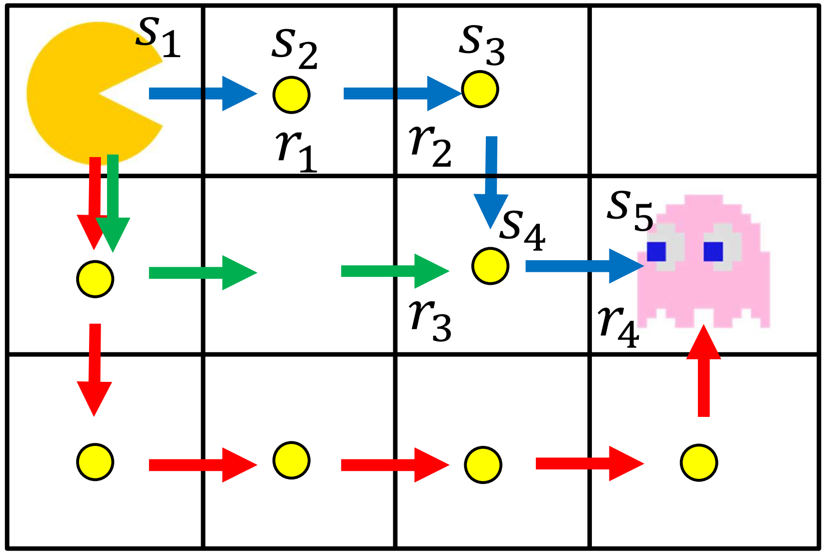
$$G(s_2) = r_2 + \gamma r_3 + \gamma^2 r_4$$

$$G(s_3) = r_3 + \gamma r_4$$

$$G(s_4) = r_4$$

Q-Learning

3. 몬테카를로 예측



$$G(s_1) = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4$$
$$G(s_2) = r_2 + \gamma r_3 + \gamma^2 r_4$$
$$G(s_3) = r_3 + \gamma r_4$$
$$G(s_4) = r_4$$

$$v_\pi(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$$

$N(s)$ 는 상태 s를 여러번의 에피소드 동안 방문한 횟수

$G_i(s)$ 는 그 상태를 방문한 i번째 에피소드에서 s의 반환값

Q-Learning

3. 몬테카를로 예측

$$v_\pi(s) \sim \frac{1}{N(s)} \sum_{i=1}^{N(s)} G_i(s)$$

$$v_{n+1}(s) = \frac{1}{n}\sum_{i=1}^{n} G_i = \left(G_n + \sum_{i=1}^{n-1} G_i \right)$$

$$= \frac{1}{n}\left(G_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} G_i \right)$$

$$= \frac{1}{n}(G_n + (n-1){\color{red}v_n})$$

$$= v_n + \frac{1}{n}(G_n - v_n)$$

$$\therefore V(s) \leftarrow V(s) + \alpha\big(G(s) - V(s)\big) \qquad \alpha: learning\ rate$$

Q-Learning

3. 몬테카를로 예측

(4) 몬테카를로 예측에서 에이전트는 이 업데이트 식을 통해
에피소드 동안 경험한 모든 상태에 대해 가치함수를 업데이트 함

(5) 샘플 수 가 많아질수록 더 정확한 가치함수 최적화가 이루어짐

(6) 단점은 없을까???

Q-Learning

4. 시간차 예측 (Temporal Difference Prediction)

(1) 몬테카를로 예측의 가장 큰 단점은 실시간이 아니라는 점이다. 다시말해, 한 에이전트의 한 에피소드가 끝나기 전까지는 가치함수의 업데이트를 할 수없다.

(2) 만약 한 에피소드가 정말 길어진다거나 끝이 없다면 몬테카를로 예측은 사용할 수 없게된다.

Q-Learning

4. 시간차 예측 (Temporal Difference Prediction)

(3) 시간차 예측이란 매 타임스텝마다 가치함수를 업데이트 하는 방법

$$V(s) \leftarrow V(s) + \alpha\big(G(s) - V(s)\big)$$

(4) 위의 몬테카를로 예측 기반 가치함수 예측식에서 반환값 $G(s)$ 는 한 에피소드가 끝나야 알 수 있음

(5) 시간차 예측에서는 다음 스텝의 보상과 가치함수를 샘플링 하여 현재 상태의 가치함수를 업데이트 한다.

$$\boxed{R + \gamma V(s_{t+1}) - V(s_t)}$$

Q-Learning

4. 시간차 예측 (Temporal Difference Prediction)

$$V(s) \leftarrow V(s) + \alpha\big(G(s) - V(s)\big)$$

$$\begin{aligned} v(s) &= E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ \dots \ | S_t = s] \\ &= E[R_{t+1} + \gamma(R_{t+2} + \gamma^1 R_{t+3} + \ \dots)| \, S_t = s] \\ &= E[R_{t+1} + \gamma(G_{t+1})| \, S_t = s] \\ &= E[R_{t+1} + \gamma v(S_{t+1})| \, S_t = s] \end{aligned}$$

반환값이긴 하지만 사실 에이전트가
실제로 받은 보상이 아직은 아님.
따라서 가치함수 형태로 나타낼 수 있음

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R + \gamma V(S_{t+1}) - V(s_t)\big)$$

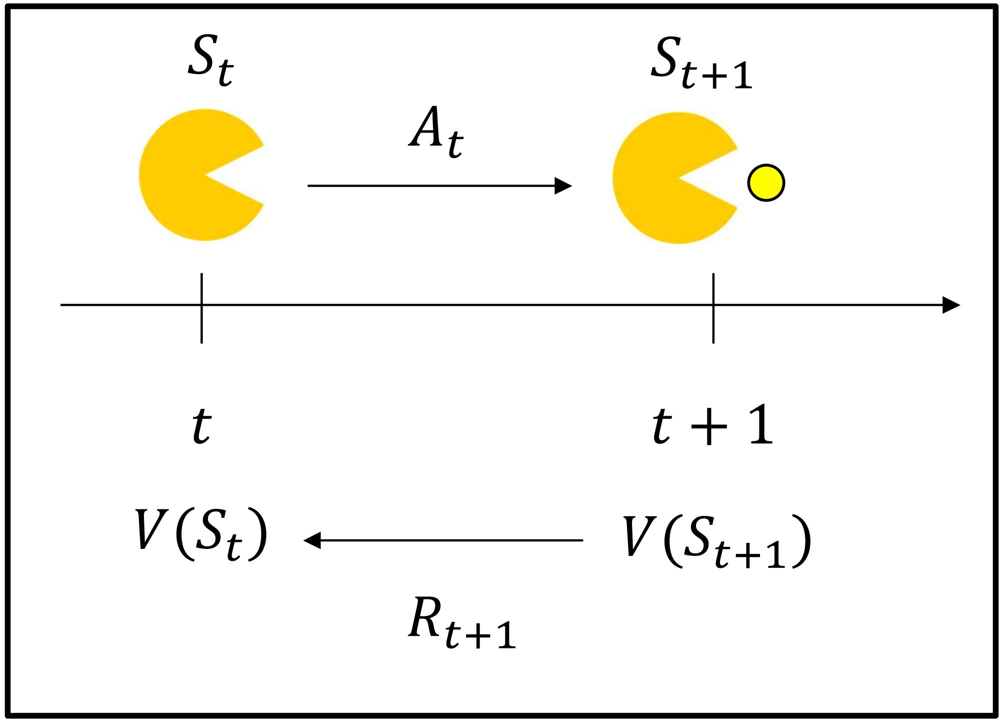
여기서 $R + \gamma V(S_{t+1})$ 를 시간차 에러(Temporal-difference error) 라고 함

Q-Learning

4. 시간차 예측 (Temporal Difference Prediction)

- 따라서 시간차 예측은 어떤 상태에서 행동을 하면 보상을 받고 다음 상태를 알게되고 다음 상태의 가치함수와 알게된 보상을 더해 그 값을 업데이트의 목표로 삼는다는 것. 이 과정을 반복

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R + \gamma V(S_{t+1}) - V(s)\big)$$



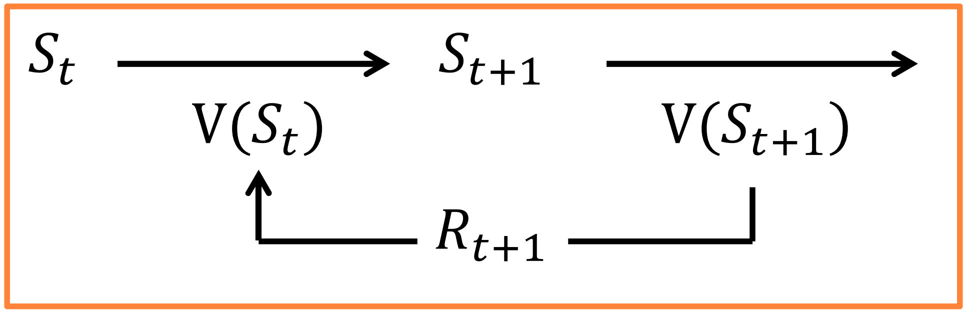
Q-Learning

4. 시간차 예측 (Temporal Difference Prediction)

(6) 시간차 예측은 매 타임스텝마다 현재 상태에서 하나의 행동을 하고 환경으로 부터 보상을 받고 다음 상태를 알게 됨

(7) 다음 상태의 예측값을 통해 현재의 가치함수를 업데이트 하는 방식을 강화학습에서는 부트스트랩(Bootstrap)이라고 함. 즉, 목표가 정확하지 않은 상태에서 현재의 가치함수를 업데이트 함

(8) 단점은 없을까?



Q-Learning

5. 살사 (SARSA)

(1) 한줄요약: 살사 = 정책 이터레이션 + 가치 이터레이션

(2) 정책 이터레이션 = 정책 평가(예측) + 정책 발전(제어)

(3) 예측: 가치함수 학습

(4) 제어: 예측을 기반으로 정책을 발전 시킴

(5) 시간차 예측의 문제점은 가치함수를 현재상태에서만 업데이트함. 즉, 모든 상태에서의 정책을 발전시키기 어렵다.

(6) 이 문제를 살사에서는 가치 이터레이션을 통해 해결함

Q-Learning

5. 살사 (SARSA)

(7) 즉 살사 = 시간차 예측 + 탐욕정책(ε-greedy)

(8) 살사에서 업데이트 하는 대상은 가치함수가 아닌 큐함수임. 왜냐하면 현재상태의 정책을 발전시키려면 $R_{t+1} + \gamma v_n(s')$의 최댓값을 알아야하는데 그러려면 정책에 대한 정보 (환경에 대한 정보)를 알야아 하기 때문

$$v_{n+1}(s) = \underset{a \in A}{\operatorname{argmax}}(R_{t+1} + \gamma v_n(s'))$$

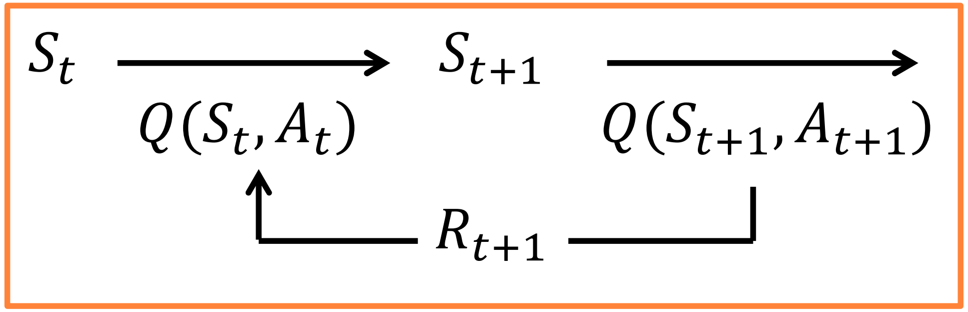
$$\pi'(s) = \underset{a \in A}{argmax}\, q_\pi(s, a)$$

Q-Learning

5. 살사 (SARSA)

(9) 큐함수를 업데이트 하려면 샘플이 필요함

$$Q(s,a) <- Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$



(10) 따라서 살사에서는 $[S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}]$를 샘플로 사용함

(11) 즉, 샘플의 상태 S_t에서 탐욕정책에 따라 A_t로 행동하고 그 다음스텝의 보상 R_{t+1}을 받음. 여기서 에이전트가 한번 더 S_{t+1}에서 행동 A_{t+1}을 하면 샘플이 생성되고 이 샘플로 큐함수를 학습시킴

Q-Learning

5. 살사 (SARSA)

(12) 살사는 큐함수를 토대로 샘플을 탐욕 정책으로 모으고 그 샘플로 방문한 큐함수를 업데이트하는 과정을 반복함

(13) 기존의 탐욕정책으로 살사 알고리즘을 실행하면 문제가 없을까?

$$S_t \xrightarrow[Q(S_t, A_t)]{} S_{t+1} \xrightarrow[Q(S_{t+1}, A_{t+1})]{}$$

$$R_{t+1}$$

Q-Learning

5. 살사 (SARSA)

(14) 초기에 무작위로 행동하는게 맞지만 계속 그렇게 할 경우 잘못된 학습을 할 가능성이 매우 큼

(15) 따라서 앞에서 사용한 탐욕정책을 개선한 ε-탐욕정책을 사용함

$$\pi(s) = \begin{cases} a^* = \underset{a \in A}{arg\max}\, Q(s,a) & ,[1-\varepsilon] \\ a \neq a^* & ,[\varepsilon] \end{cases}$$

(16) 하지만 이 방법은 최적의 큐함수를 찾아도 일정 확률(ε)로 무작위 행동을 하며 탐험한다는 단점이 있음

(17) 이는 초기에 설정한 ε값을 학습 시간이 흐름에 따라 점점 감소시키는 방법을 통해 해결할 수 있음

Q-Learning

5. 살사 (SARSA)

(18) 살사 요약

- ε-greedy policy (exploration)

$$\pi(s) = \begin{cases} a^* = \underset{a \in A}{argmax}\, Q(s,a) & ,[1-\varepsilon] \\ a \neq a^* & ,[\varepsilon] \end{cases}$$

- Sampling

$$[S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}]$$

- Update Q-function

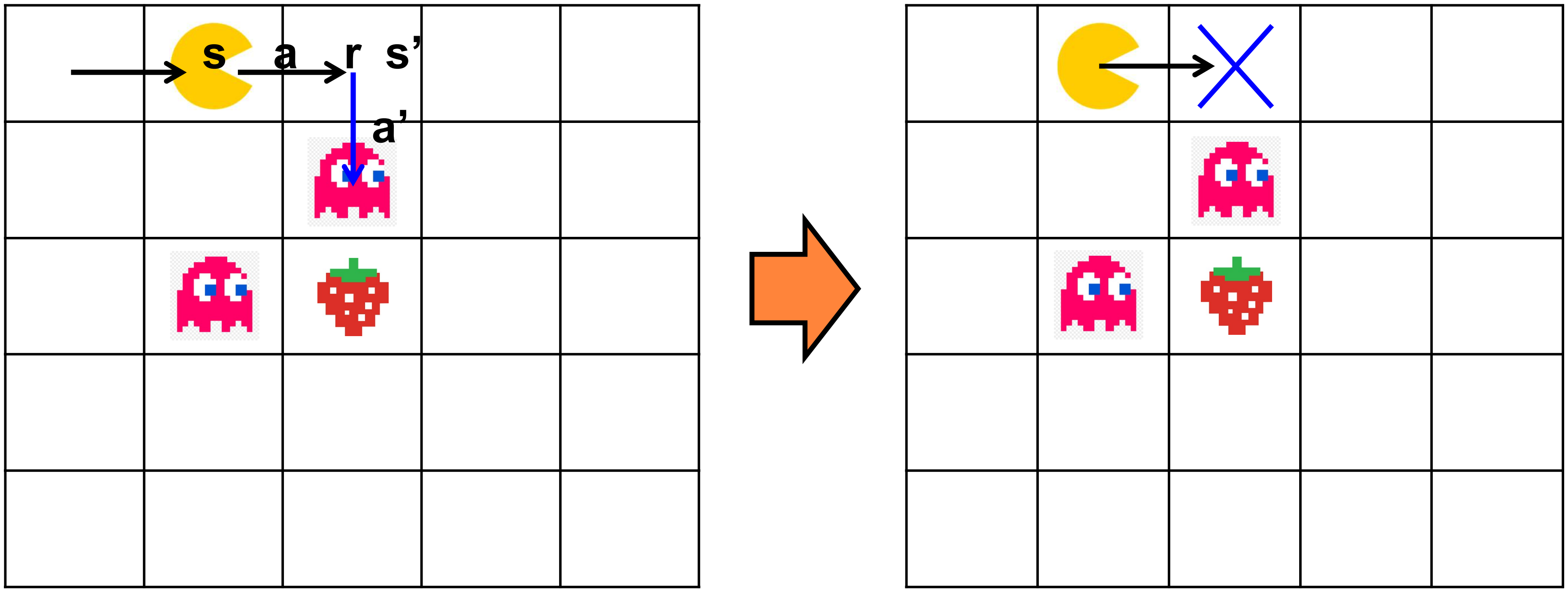
$$Q(s,a) <- Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

Q-Learning

5. 살사 (SARSA)

(19) 살사의 한계(a.k.a On-Policy)

(20) 온폴리시(On-Policy)란 행동 정책과 학습 정책이 같은걸 의미



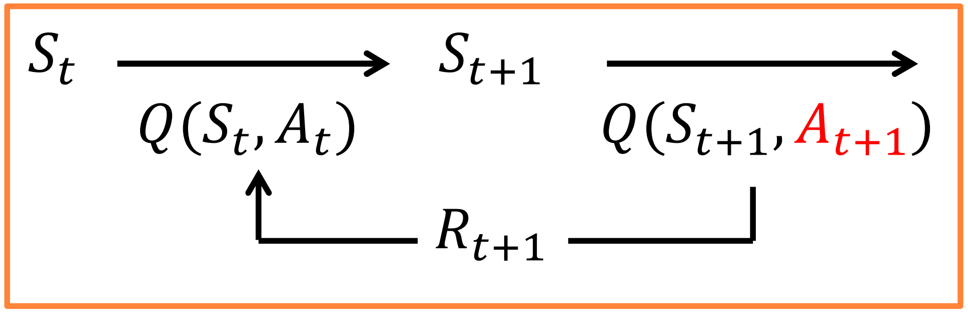
Q-Learning

6. 큐러닝 (Q-Learning)

(1) 온폴리시 학습의 경우 탐험에서 문제점이 발생함

(2) 강화학습에서 탐험은 필수적인데 그렇다고 안할 수도 없고...

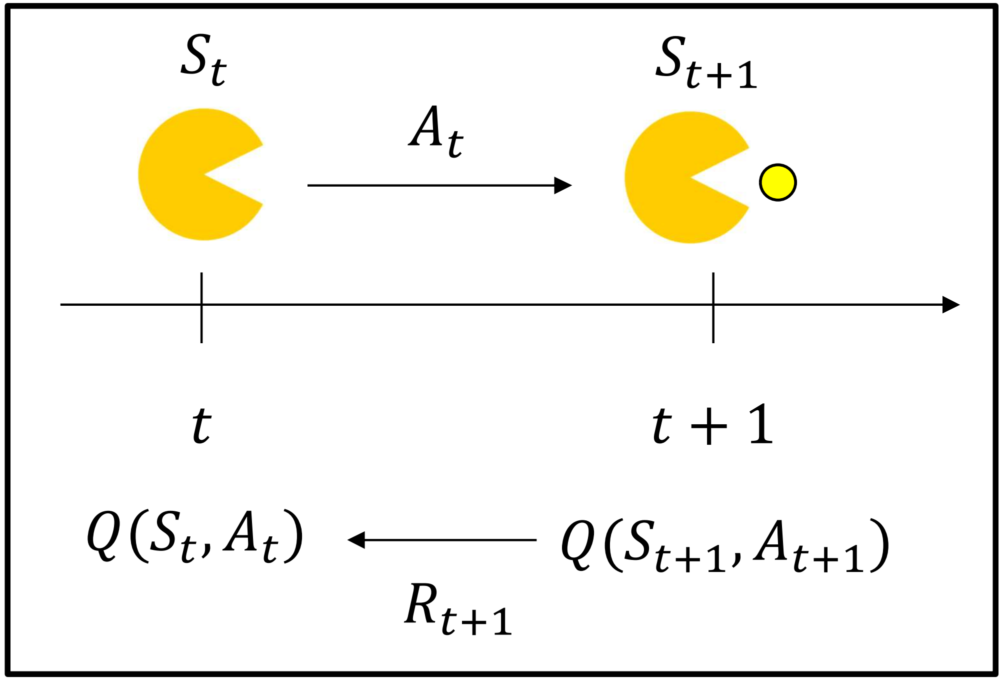
(3) 온폴리시에서 문제가 발생하는 이유가 행동 정책과 학습 정책이 같아서니까 그 둘이 다르면 되지 않을까???



Q-Learning

6. 큐러닝 (Q-Learning)

- 큐러닝은 에이전트가 다음상태를 알게되면 그 상태에서 가장 큰 큐함수를 현재 큐함수의 업데이트에 사용함.



$$\mathrm{Q}(S_t, A_t) \leftarrow \mathrm{Q}(S_t, A_t) + \alpha\left(R + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$

Q-Learning

6. 큐러닝 (Q-Learning)

(4) 이와 같은 방식을 오프폴리시(Off-Policy)라고 함

(5) 큐러닝은 살사의 딜레마를 해결하기 위해 행동 선택은 ε-탐욕정책으로, 업데이트는 벨만 최적 방정식으로 진행한다.

Q-Learning

6. 큐러닝 (Q-Learning)

(6) 큐러닝 요약

- ε-greedy policy (exploration)

$$\pi(s) = \begin{cases} a^* = \underset{a \in A}{argmax}\, Q(s,a) \;, [1-\varepsilon] \\ a \neq a^* \;, [\varepsilon] \end{cases}$$

- Sampling

$$[S_t\,, A_t\,, R_{t+1}, S_{t+1}\,, A_{t+1}]$$

- Update Q-function

$$Q(s,a) <- Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

Q-Learning

6. 큐러닝 (Q-Learning)

(6) 큐러닝 요약

- ε-greedy policy (exploration)

$$\pi(s) = \begin{cases} a^* = \underset{a \in A}{argmax}\, Q(s,a) & ,[1-\varepsilon] \\ a \neq a^* & ,[\varepsilon] \end{cases}$$

- Sampling

$$[S_t \,, A_t \,, R_{t+1}, S_{t+1} \,]$$

- Update Q-function

$$Q(s,a) <- Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

Q-Learning

6. 큐러닝 (Q-Learning)

(6) 큐러닝 요약

- ε-greedy policy (exploration)

$$\pi(s) = \begin{cases} a^* = \underset{a \in A}{argmax}\, Q(s,a) & ,[1-\varepsilon] \\ a \neq a^* & ,[\varepsilon] \end{cases}$$

- Sampling

$$[S_t\ , A_t\ , R_{t+1}, S_{t+1}\ ]$$

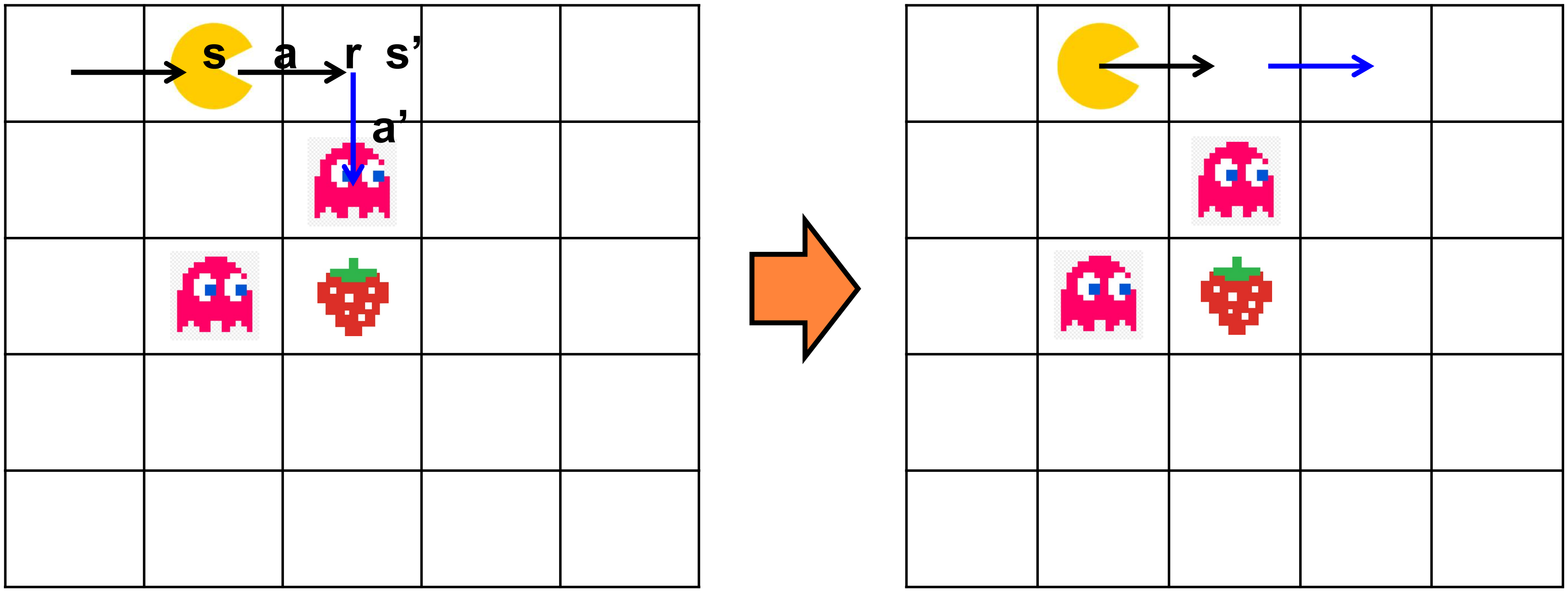
- Update Q-function

$$\mathrm{Q}(S_t, A_t) \leftarrow \mathrm{Q}(S_t, A_t) + \alpha\left(R + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$

Q-Learning

6. 큐러닝 (Q-Learning)

아까와 같이 구석에 갇히는 경우가 발생하지 않음



Q-Learning

진단 평가

1) 예측과 제어에 대해서 설명하시오
2) 몬테카를로 예측이란? 문제점은?
3) 시간차 예측이란? 문제점은?
4) 살사란? 문제점은?
5) 큐러닝이란? 문제점은?

Outline

- Overview
- Basics
- Dynamic Programming
- Q-Learning
- Deep Reinforcement Learning
- Deep Q-Networks (DQN)
- Advantage Actor-Critic (A2C)
- Asynchronous Advantage Actor-Critic (A3C)
- Applications

Q & A

About me

Jeiyoon Park

I'm a master's student in the Department of Computer Science and Engineering at Korea University, advised by the professor Heuiseok Lim.

My research interests are dialog systems, reinforcement learning and meta-learning.

Email / Github / Google Scholar / LinkedIn

https://jeiyoon.github.io/

https://www.youtube.com/channel/UC5dx094F-Se1DMI1vvVJyRw