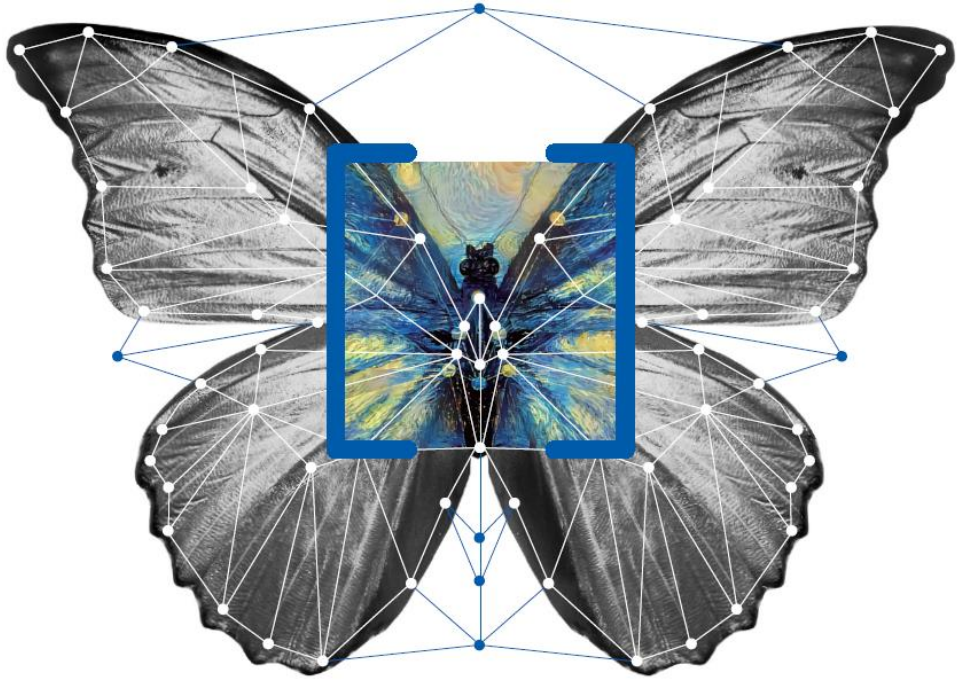
한빛아카데미 Hanbit Academy, Inc.



# MACHINE LEARNING 기계 학습

오일석 지음

## 9. 강화 학습

# PREVIEW

- 강화 학습의 과업 사례
  - [그림 9-1]의 과업은 상태 변화와 행동을 번갈아 수행하면서 목적을 달성
  - 지도 학습이나 비지도 학습으로는 과업 달성 불가능 → 강화 학습으로 해결

(a) 사람의 걷기 학습 과정

(b) 2족 로봇의 걷기

**그림 9-1 걷기 과업**

# PREVIEW

- 강화 학습의 응용
  - 스타크래프트와 같은 온라인 게임
  - 장기나 바둑, 윷놀이 같은 게임
  - 엘리베이터 최적 제어, 자동차의 자율 주행, 정유 생산 라인의 제어, 로켓 발사 제어 등
  - 바둑(알파고와 알파고 제로)

# 각 절에서 다루는 내용

**9.1절_** 강화 학습의 기본 원리와 특성을 기술한다.

**9.2절_** 가치함수를 이용하여 최적 정책을 찾아내는 방식을 설명한다.

**9.3절_** 초창기 알고리즘인 동적 프로그래밍을 소개한다.

**9.4절_** 샘플을 이용하여 학습하는 몬테카를로 방법을 소개한다.

**9.5절_** 시간차 학습의 기본 개념을 소개하고 널리 쓰이는 Sarsa와 Q-학습 알고리즘을 설명한다.

**9.6절_** 신경망을 이용하여 근사해를 구하는 방법을 간략히 소개한다.

**9.7절_** 강화 학습의 성공적인 응용 사례로서 TD-gammon과 DQN을 소개하며 인간 지능에 한발 다가섰다고 평가되는 이유를 설명한다.

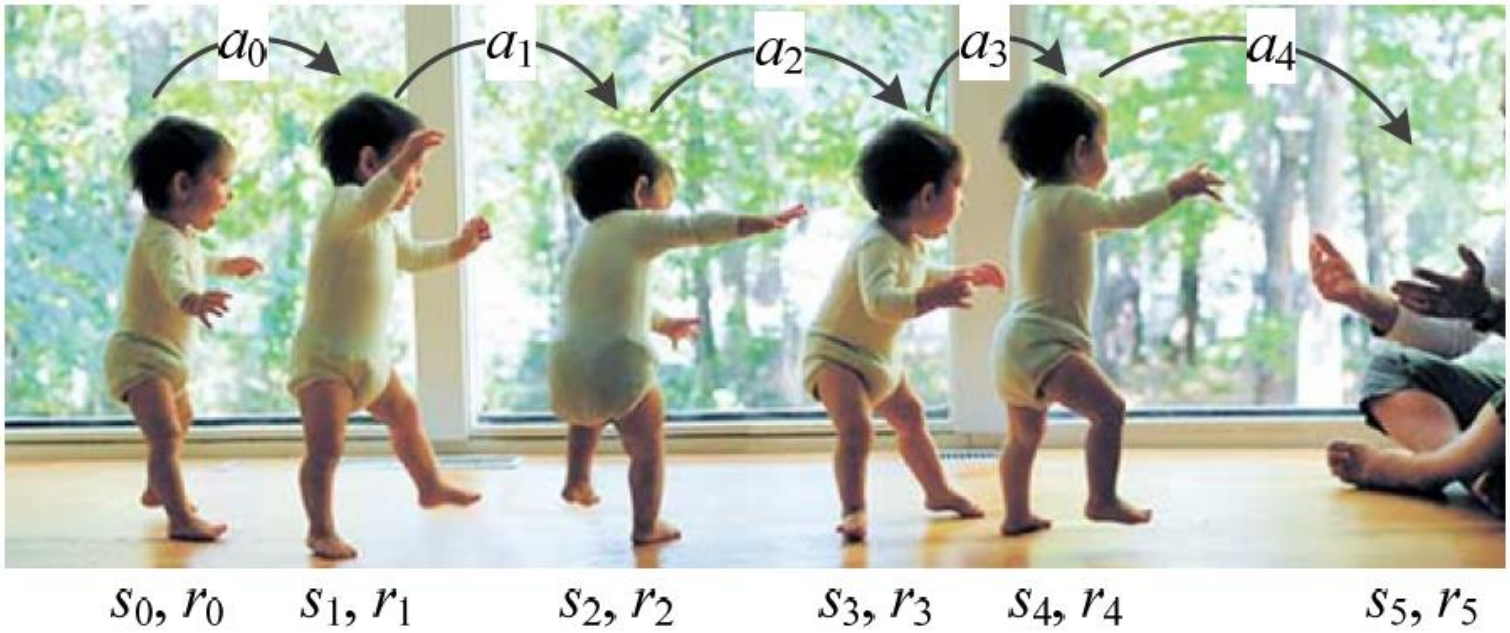
# 9.1 강화 학습의 원리와 성질

- 9.1.1 계산 모형
- 9.1.2 탐험과 탐사
- 9.1.3 마르코프 결정 프로세스

# 9.1.1 계산 모형

- 강화 학습의 핵심 연산
  - 시간에 따른 상태state, 행동action, 보상reward 의 순차적 처리 $t$=0, 1, 2, 3, …, $T$
    - $s_0, r_0 \xrightarrow{a_0} s_1, r_1 \xrightarrow{a_1} s_2, r_2 \xrightarrow{a_2} \cdots\cdots \xrightarrow{a_{T-1}} s_T, r_T$
  - 식으로 쓰면,

강화 학습의 핵심 연산 $f: (s_t, a_t) \rightarrow (s_{t+1}, r_{t+1})$ (9.1)




[그림 9-2] 강화 학습의 핵심 개념인 상태 $s_i$, 행동 $a_i$, 보상 $r_i$

# 9.1.1 계산 모형

- 보상 책정 방안
  - 매 순간 보상액 책정이 어려우면 중간에는 보상액을 0으로 설정하고 마지막 순간에 보상액 결정
  - 예, [그림 9-2]에서 중간에는 보상액 0이고, 마지막 순간에는 아빠에게 안기면 1, 넘어지면 -1 부여
  - 예, 바둑에서 중간에는 보상액 0, 마지막에 이기면 1, 지면 -1 부여

# 9.1.1 계산 모형

- 에이전트와agent 환경environment
  - 에이전트는 정책을 가지고 행동을 결정(정책은 학습으로 알아내야 함)
  - 환경은 MDP를 가지고 상태 전환과 보상액 결정(MDP는 주어지는 것: 지도 학습의 훈련집합에 비유할 수 있음)

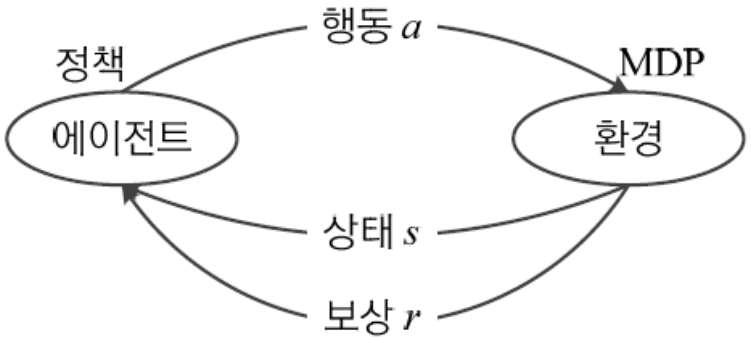



그림 9-3 강화 학습의 계산 모델

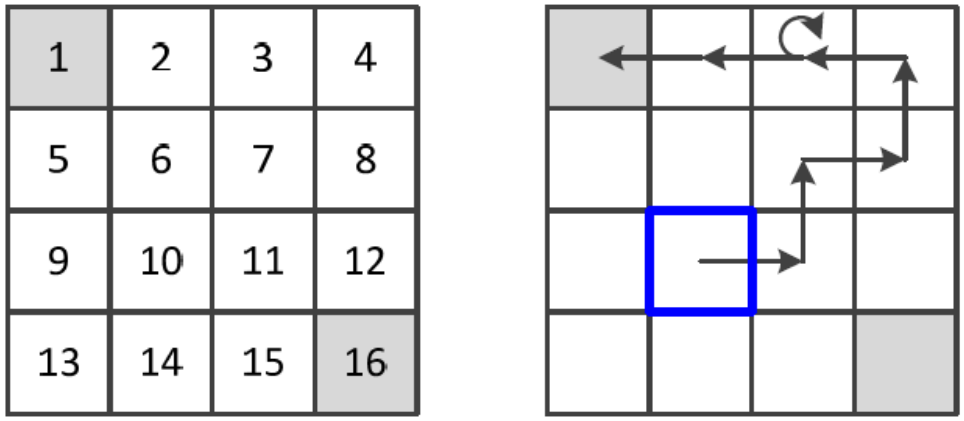
- 강화 학습의 목표는 누적 보상액을 최대화하는 것
  - 이를 달성하려면 매 순간 좋은 행동을 취할 수 있어야 함 → 정책에 따라 행동을 결정하므로 좋은 정책이 필요한 셈
  - 강화 학습은 주어진 MDP를 가지고 최적 정책을 찾아야 함(지도 학습이 훈련집합을 가지고 최적 매개변수를 찾는 것에 비유할 수 있음)

# 9.1.1 계산 모형

- 상태, 행동, 보상의 표현
  - 대부분 응용에서는 이산 값을 가짐
  - 상탯값 집합 $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$, 행동값 집합 $\mathcal{A} = \{a_1, a_2, \cdots, a_m\}$, 보상값 집합 $\mathcal{R} = \{r_1, r_2, \cdots, r_l\}$으로 표기
  - 예, $P(S_{t+1} = s_{58} | S_t = s_{122})$는 $t$ 순간에 $s_{122}$ 상태에 있다가 다음 순간에 $s_{58}$ 상태로 바뀔 확률
  - 예, $P(S_{t+1} = s_{58} | S_t = s_{122}, a_t = a_2)$는 $t$ 순간에 $s_{122}$ 상태에서 행동 $a_2$를 취했을 때 $s_{58}$ 상태로 바뀔 확률

# 9.1.1 계산 모형

**예제 9-1** 격자 세계




(a) 예제 격자 보드 (b) 이동 경로의 예

**그림 9-4 격자 세계**

[그림 9-4(a)]는 로봇이 돌아다닐 수 있는 4*4 격자 보드이다. 회색으로 표시된 1과 16번 칸이 종료 지점이다. 로봇은 현재 위치에서 동서남북 네 방향으로 이동할 수 있다. 가장자리 칸에서 보드 밖으로 나가는 방향을 선택하면 원래 자리에 머물며 보상 -1을 받는다. 밖으로 나가지 않는 방향을 선택하면 그 방향으로 이동하며 보상 0을 받는다. 종료 지점에 들어가면 보상 5를 받는다.

[그림 9-4(b)]를 살펴보자. 시작 위치가 파란색으로 표시된 10번 칸인데, 이 칸에서 동쪽으로 이동하면 보상은 0이고 다음 위치는 11이 된다. 반면, 여섯 번째 이동은 3번 칸에서 북쪽을 선택하였는데 보드 바깥으로 나가게 되므로 보상은 -1이고 원래 위치에 머문다.

# 9.1.1 계산 모형

이 문제를 위한 상태집합, 행동집합, 보상집합을 앞서 정의한 표기법으로 표현하면 다음과 같다. 로봇의 위치가 상태에 해당하며, 로봇의 이동 방향이 행동에 해당한다. 상태는 칸의 번호가 $i$라면 $s_i$라고 표기해야 하는데, 간결하게 그냥 $i$로 표기하자. 행동집합 $\mathcal{A}$와 보상집합 $\mathcal{R}$도 마찬가지로 직관적으로 이해하기 쉬운 표기를 사용한다.

$$\mathcal{S} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$$
$$\mathcal{A} = \{\text{동}, \text{서}, \text{남}, \text{북}\}$$
$$\mathcal{R} = \{0, 5, -1\}$$

[그림 9-4(b)]는 예제의 이동 경로를 보여 주며, 이 경로는 다음과 같이 표현할 수 있다.

$$\text{이동 경로 } (10,0) \xrightarrow{\text{동}} (11,0) \xrightarrow{\text{북}} (7,0) \xrightarrow{\text{동}} (8,0) \xrightarrow{\text{북}} (4,0) \xrightarrow{\text{서}} (3,0) \xrightarrow{\text{북}} (3,-1) \xrightarrow{\text{서}} (2,0) \xrightarrow{\text{서}} (1,5)$$

시작 순간 $t = 0$에서는 $s_0 = 10$, $r_0 = 0$이다. $a_0$으로 동쪽이라는 행동을 선택하였는데, 이 행동으로 다음 순간에는 $s_1 = 11$이 되고 $r_1 = 0$을 받았다. $t = 5$ 순간에는 상태 $s_5 = 3$에 있는데, $a_5$ = 북쪽을 선택하여 $s_6 = 3$이 되고 보상으로 $r_6 = -1$을 받았다. 결과적으로 이 이동 경로의 누적 보상, 즉 보상 총액은 $0+0+0+0+0+0-1+0+5=4$이다.

처음 위치에서 출발한 로봇은 '누적 보상을 최대화'하면서 회색으로 표시된 종료 지점에 도달해야 한다. 이제부터 이러한 목적을 달성하기 위한 방법을 고안해 보자.

# 9.1.1 계산 모형

- 정책policy
  - 에이전트는 확률 규칙으로 행동을 결정하는데, 이 규칙을 정책이라 부름
  - [예제 9-1]이 동, 서, 남, 북이라는 4가지 행동에 대해 동일 확률을 사용한다고 가정하면, 이 정책을 다음과 같이 쓸 수 있음

[예제 9-1]의 정책:

$$P(a = \text{동}|s = i) = P(a = \text{서}|s = i) = P(a = \text{남}|s = i) = P(a = \text{북}|s = i) = \frac{1}{4},\ i = 2,3,\cdots,15$$

  - 동일 확률을 쓰는 이 정책이 좋은 정책인가?

# 9.1.1 계산 모형

## 더 좋은 정책

- 밖으로 나가는 행동을 배제하면 누적 보상액이 항상 5가 됨

[예제 9-1]을 개선한 정책(첫 행에 있는 칸 2, 3, 4에 대해서만 기술):

$$P(a = \text{동}|s = i) = \frac{1}{3}, P(a = \text{서}|s = i) = \frac{1}{3}, P(a = \text{남}|s = i) = \frac{1}{3}, P(a = \text{북}|s = i) = 0, \quad i = 2,3,4$$

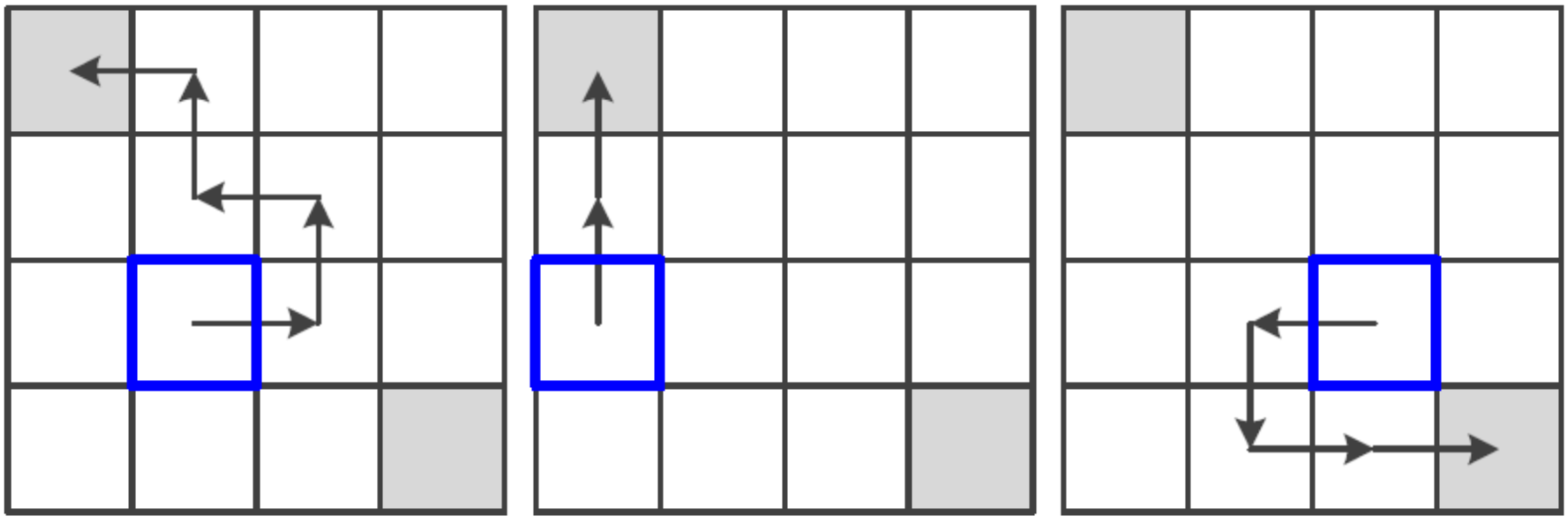


그림 9-5 보드 밖으로 나가는 상황을 배제한 효율적인 정책

# 9.1.2 탐험과 탐사

- 탐험과 탐사 갈등 현상
  - 탐험은 전체 공간을 골고루 찾아보는 전략
  - 탐사는 특정한 곳 주위를 집중적으로 찾아보는 전략
    - [알고리즘 2-2](무작위 탐색 알고리즘)은 극단적인 탐험 방식, [알고리즘 2-3]과 지금까지 사용한 경사 하강법은 탐사 방식
    - 탐험은 시간이 너무 오래 걸리고, 탐사는 지역 최적해에 머무는 문제
- 강화 학습에서는 탐사로 치우치지 않게 주의를 기울여야 함
  - 예, *k*-손잡이 밴딧 문제


그림 9-6 *k*-손잡이 밴딧 문제

# 9.1.2 탐험과 탐사

- 처음에 탐험 방식으로 게임을 하다가, 12번 만에 2번 손잡이에서 잭팟이 터졌다고 가정하면,

  3, 4, 1, 2, 5, 4, 3, 1, 2, 5, 1, **2**

- 극단적인 탐사와 극단적인 탐험을 예시하면,

  3, 4, 1, 2, 5, 4, 3,1, 2, 5, 1, **2**, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, … 탐사 방식

  3, 4, 1, 2, 5, 4, 3, 1, 2, 5, 1, **2**, 3, 5, 4, 3, 1, 2, 5, 3, 1, 2, … 탐험 방식

- 균형 방식을 예시하면,

  3, 4, 1, 2, 5, 4, 3, 1, 2, 5, 1, **2**, 3, 4, 2, 3, 2, 1, 5, 2, 2, 4, … 균형 방식

- 강화 학습은 균형 방식을 사용함(9.4~9.5절)

# 9.1.3 마르코프 결정 프로세스

## 마르코프 성질

- 행동을 결정할 때 이전 이력이 중요하지 않다면 마르코프 성질을 만족함
  - [예제 9-1]의 격자 세계와 [그림 9-7]의 바둑은 마르코프 성질을 만족

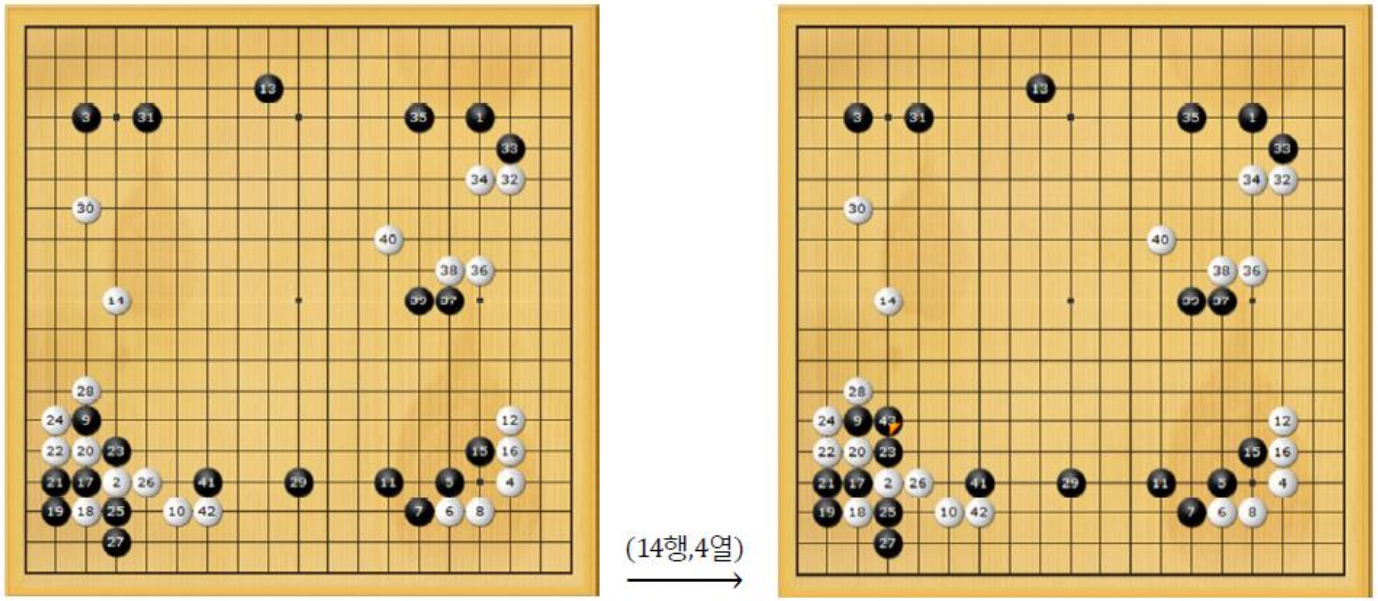



그림 9-7 바둑[1]

# 9.1.3 마르코프 결정 프로세스

- 마르코프 성질을 수식으로 표현하면,

$$P(s_{t+1}, r_{t+1} | s_0, a_0, r_1, s_1, a_1, \cdots, r_{t-1}, s_{t-1}, a_{t-1}, r_t, s_t, a_t) = P(s_{t+1}, r_{t+1} | s_t, a_t) \quad (9.2)$$

  - 첨자를 생략하여 간략히 표기하면,

$$P(s_{t+1}, r_{t+1} | s_t, a_t) = P(s', r | s, a) \quad (9.3)$$

- 마르코프 성질을 만족하지 못하는 문제
  - 크게 벗어나면 강화 학습 적용할 수 없음
  - 근사하게 만족하도록 상태 표현 설계 가능
    - 예, 날씨에 따른 행동 결정 문제에서는 이틀 날씨를 상태로 표현

$$S = \{\text{맑은후맑음}, \text{비온후맑음}, \text{눈온후맑음}, \cdots, \text{눈온후눈}\}$$

    - 예, 로켓 발사 제어 문제에서 위치 정보만 사용하는 대신 위치, 속도, 가속도 정보로 상태를 표현

# 9.1.3 마르코프 결정 프로세스

- 환경 모델로서 MDP(Markov decision process)
  - 환경은 MDP 확률분포를 가지고 다음 상태와 보상을 정함 ([그림 9-3])

$$\text{MDP 확률분포: } P(s', r|s, a), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}, \forall s' \in \mathcal{S}, \forall r \in \mathcal{R} \tag{9.4}$$

  - 결정론적 MDP
    - 예, [예제 9-1]의 경우(상태 10에서 동쪽이라는 행동을 취하면 항상 11상태로 전환하고 보상은 0)

$$P(11,0|10, \text{동}) = 1, \; P(12,0|10, \text{동}) = 0, \; P(11,-1|10, \text{동}) = 0, \; \cdots$$

  - 스토캐스틱 MDP
    - 예, 가끔 돌풍이 불어 동쪽 행동을 취했는데 일정한 확률로 11이 아니라 7이나 15로도 갈 수 있는 경우

# 9.2 정책과 가치함수

- 9.2.1 정책
- 9.2.2 가치함수
- 9.2.3 최적 가치함수

# 9.2 정책과 가치함수

- 강화 학습의 핵심은 좋은 정책을 찾는 것
  - 좋은 정책이 있으면 누적 보상액을 최대로 만들 최적 행동을 매 순간 선택할 수 있음
  - 좋은 정책을 찾는 일은 지도 학습이 목적함수의 최적해 찾는 일에 비유할 수 있음

- 좋은 정책
  - 누적 보상을 최대화하려고 일부러 함정에 빠지는 행동까지 추론 가능해야 함
  - 예, 바둑에서 작은 말을 희생하여 대마를 잡는 상황

# 9.2.1 정책

- 정책 $\pi$는 상태 $s$에서 행동 $a$를 취할 확률을 명시한 것

$$\pi(a|s) = P(a|s), \forall s, \forall a \tag{9.5}$$

  - 예, [예제 9-1]의 정책 $\pi_1$([그림 9-4])

$$\pi_1: \quad P(\text{동}|i) = P(\text{서}|i) = P(\text{남}|i) = P(\text{북}|i) = \frac{1}{4}, \qquad i = 2, \cdots, 15 \tag{9.6}$$

**예제 9-2** **격자 세계를 위한 최적 정책**

[예제 9-1]의 MDP는 산책하는 로봇을 만들 때 적합하다. 보드 바깥으로 나가지 않는다면 아무리 먼 길을 돌아와도 같은 누적 보상 5를 받기 때문이다. 특정 임무를 맡고 빨리 종료 상태에 도달해야 하는 로봇을 만들어야 한다면 MDP를 수정해야 한다. 이제 종료 상태로 이동을 제외한 모든 이동은 -1, 바깥으로 나가면 -2, 종료 상태로 이동은 5라는 보상을 받는 MDP를 사용한다고 가정하자. 로봇은 짧은 시간 안에 종료 지점에 도달해야 한다.

[그림 9-8]은 여러 가지 정책을 예시한다. 가장 왼쪽은 식 (9.6)의 동일 확률 정책 $\pi_1$이다. $\pi_1$에 따라 바로 아래에 있는 예제 해가 생성되었다고 가정한다.

# 9.2.1 정책

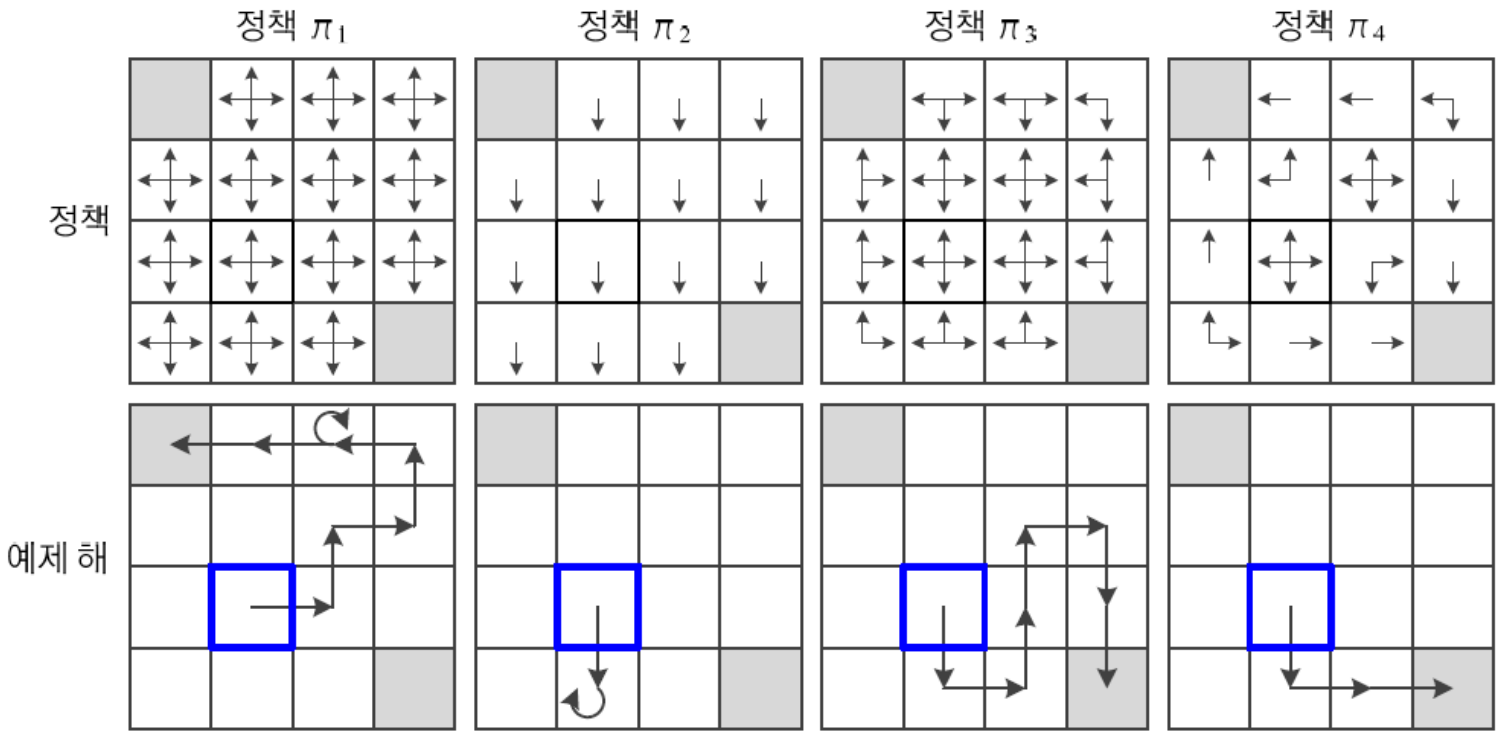



**그림 9-8 여러 가지 정책($\pi_4$는 최적 정책임)**

남으로만 갈 수 있는 $\pi_2$라는 정책을 생각해 보자. $\pi_2$는 다음과 같은 식으로 정의된다. 이 정책은 4, 8, 12 중 한 곳에서 시작하지 않는 한 종료 상태에 도달할 수 없다. 따라서 해의 품질을 떠나 아예 채택할 수 없는 정책이라 할 수 있다.

$$\pi_2: \ P(\text{동}|i) = P(\text{서}|i) = P(\text{북}|i) = 0, \ P(\text{남}|i) = 1, \text{ 상태 } i = 2, \cdots, 15$$

# 9.2.1 정책

정책 $\pi_3$은 $\pi_1$을 개선한 것이다. 가장자리의 칸에서 바깥으로 나가는 방향을 허용하지 않음으로써 −2라는 보상을 피하는 정책이다. $\pi_3$ 수식은 다음과 같다.

$$\pi_3 \begin{cases} P(\text{동}|i) = P(\text{서}|i) = P(\text{남}|i) = P(\text{북}|i) = \frac{1}{4}, & i = 6, 7, 10, 11 \\ P(\text{서}|i) = P(\text{남}|i) = \frac{1}{2}, & i = 4 \\ P(\text{동}|i) = P(\text{북}|i) = \frac{1}{2}, & i = 13 \\ P(\text{동}|i) = P(\text{남}|i) = P(\text{북}|i) = \frac{1}{3}, & i = 5, 9 \\ P(\text{서}|i) = P(\text{남}|i) = P(\text{북}|i) = \frac{1}{3}, & i = 8, 12 \\ P(\text{동}|i) = P(\text{서}|i) = P(\text{남}|i) = \frac{1}{3}, & i = 2,3 \\ P(\text{동}|i) = P(\text{서}|i) = P(\text{북}|i) = \frac{1}{3}, & i = 14, 15 \end{cases}$$

# 9.2.1 정책

[그림 9-8]의 맨 오른쪽 정책에서는 2나 3 상태에서 항상 서쪽을 향한다. 다른 방향으로 가면 헤맬 수 있고, 서쪽을 향하면 가장 짧은 시간에 종료 상태 1에 도달할 수 있기 때문이다. 6번 상태에서는 서로 가든 북으로 가든 경로의 길이가 같으므로 어느 것을 선택하든 무방하다. 이 정책을 수식으로 쓰면 다음과 같다.

$$\pi_4 \begin{cases} P(\text{서}|i) = 1, & i = 2,3 \\ P(\text{서}|i) = P(\text{남}|i) = \frac{1}{2}, & i = 4 \\ P(\text{북}|i) = 1, & i = 5,9 \\ P(\text{서}|i) = P(\text{북}|i) = \frac{1}{2}, & i = 6 \\ P(\text{동}|i) = P(\text{서}|i) = P(\text{남}|i) = P(\text{북}|i) = \frac{1}{4}, & i = 7,10 \\ P(\text{남}|i) = 1, & i = 8,12 \\ P(\text{동}|i) = P(\text{남}|i) = \frac{1}{2}, & i = 11 \\ P(\text{동}|i) = P(\text{북}|i) = \frac{1}{2}, & i = 13 \\ P(\text{동}|i) = 1, & i = 14,15 \end{cases}$$

조금만 생각해 보면 $\pi_4$를 사용할 때 항상 가장 짧은 시간에 목표 상태에 도달하여 최고의 누적 보상액을 받을 수 있음을 알 수 있다. 즉, 새로운 MDP에서는 $\pi_4$가 최적 정책이다. 강화 학습은 학습 알고리즘을 이용하여 $\pi_4$를 찾아내야 한다.

# 9.2.1 정책

- 학습 알고리즘이 할 일

학습 알고리즘이 해야 할 일은 최적 정책 $\hat{\pi}$을 찾는 것이다.
즉, $\hat{\pi} = \underset{\pi}{\operatorname{argmax}}\, goodness(\pi)$ (9.7)

  - 정책 공간이 방대하여 모든 정책을 일일이 평가하고 그중 제일 좋은 것을 선택하는 방법은 현실성이 없음 → 따라서 가치함수를 이용하는 전략 사용

# 9.2.2 가치함수

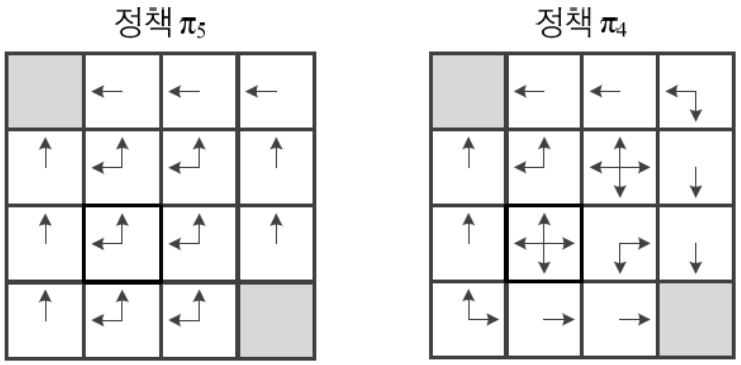
- 가치함수는 특정 정책의 좋은 정도를 측정하는 함수
  - 좋은 정도는 상태 $s$로부터 종료 상태에 이르기까지 누적 보상액의 추정치
  - 특정 정책 $\pi$에서 추정하며 상태 $s$의 함수이므로 $v_\pi(s)$로 표기

**예제 9-3** **가치함수로 정책 평가**

여기서는 지름길이 더 좋은 보상을 받는 [예제 9-2]의 MDP를 사용한다고 가정한다.

[그림 9-9(a)]의 $\pi_5$ 정책을 살펴보자. 이 정책에서는 모든 상황이 종료 상태 1로 향한다. 이 정책을 확률로 표기하면 다음과 같다.

$$\pi_5 \begin{cases} P(\text{서}|i) = 1, & i = 2,3,4 \\ P(\text{북}|i) = 1, & i = 5,8,9,12,13 \\ P(\text{서}|i) = P(\text{북}|i) = \frac{1}{2}, & i = 6,7,10,11,14,15 \end{cases}$$




(a) 종료 상태 1로만 향하는 정책 $\pi_5$ (b) 최적 정책

**그림 9-9** 두 가지 정책 예제와 가치함수

# 9.2.2 가치함수

상태 2에 대한 가치함수의 값, 즉 $v_{\pi 5}(2)$를 계산해 보자. 상태 2에서는 서쪽으로 가는 길만 있고 바로 종료 상태에 도달하여 누적 보상 5를 얻으므로 $v_{\pi 5}(2) = 5$이다. 상태 3은 서서라는 외길이 있고 누적 보상은 -1+5이므로 $v_{\pi 5}(3) = 4$이다.

상태 6으로 관심을 옮겨보자. 상태 6에서는 0.5 확률로 서 또는 북으로 향할 수 있다. 종료 상태로 이어지는 경로는 서북과 북서 2개이고 둘은 0.5라는 확률로 발생할 것이다. 따라서 $v_{\pi 5}(6)$은 다음과 같이 계산할 수 있다.

$$v_{\pi 5}(6) = \frac{1}{2}(-1+5) + \frac{1}{2}(-1+5) = 4$$

상태 11에서는 북북서서, 북서북서, 북서서북, 서북북서, 서북서북, 서서북북의 여섯 가지 길이 있으므로 다음과 같이 계산하여 $v_{\pi 5}(11) = 2$를 얻는다.

$$v_{\pi 5}(11) = \frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5)$$
$$+ \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}(-1-1-1+5) = 2$$

[그림 9-9(b)]의 정책 $\pi_4$를 살펴보자. 이 정책은 [그림 9-8]의 $\pi_4$와 같다. 이 정책에서는 상태 11에서 종료 상태에 도달하는 길이 동남과 남동 두 가지이다. 따라서 $v_{\pi 4}(11)$은 다음과 같이 계산하여 4가 된다. $v_{\pi 5}(11) = 2$인 점을 고려하면 $\pi_4$가 $\pi_5$보다 좋다. 다른 상태에서도 계산하면 모든 상태에서 $\pi_4$가 $\pi_5$보다 좋다는 사실을 알 수 있다.

$$v_{\pi 4}(11) = \frac{1}{2}(-1+5) + \frac{1}{2}(-1+5) = 4$$

# 9.2.2 가치함수

- 식 (9.7)을 다시 쓰면,

$$\hat{\pi} = \underset{\pi}{\operatorname{argmax}}\, v_\pi(s), \forall s \in \mathcal{S} \qquad (9.8)$$

- 가치함수 $v_\pi(s)$를 구하는 식
  - $P(z)$는 경로 $z$의 발생 확률, $\mathbb{r}(z)$는 경로 $z$의 누적 보상액

$$v_\pi(s) = \sum_{s\text{에서 출발하는 모든 경로 } z} P(z)\mathbb{r}(z) \qquad (9.9)$$

- 누적 보상액을 계산하는 식

$$\mathbb{r}(z) = r_{t+1} + r_{t+2} + \cdots + r_T \qquad (9.10)$$

  - 예, [예제 9-3]의 정책 $\pi_5$에 따라 상태 11에서 출발하여 북북서서를 거쳐 상태 1에 도달하는 경로는 -1-1-1+5=2라는 누적 보상액을 받음

# 9.2.2 가치함수

- 에피소드 과업과 영구 과업
  - 아래와 같이 유한 경로 (파란 부분)가 발생하는 과업을 에피소드 과업이라 부름

유한 경로(에피소드 과업) $z: (s_t, r_t) \xrightarrow{a_t} (s_{t+1}, r_{t+1}) \xrightarrow{a_{t+1}} (s_{t+2}, r_{t+2}) \xrightarrow{a_{t+2}} \cdots \xrightarrow{a_{T-1}} (s_T, r_T)$

  - 아래와 같이 무한 경로 (파란 부분)가 발생하는 과업을 영구 과업이라 부름

무한 경로(영구 과업) $z: (s_t, r_t) \xrightarrow{a_t} (s_{t+1}, r_{t+1}) \xrightarrow{a_{t+1}} (s_{t+2}, r_{t+2}) \xrightarrow{a_{t+2}} (s_{t+3}, r_{t+3}) \xrightarrow{a_{t+3}} (s_{t+4}, r_{t+4}) \cdots$

- 영구 과업이 사용하는 할인 누적 보상액

$$\mathbb{r}(z) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots = \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \qquad (9.11)$$

  - 할인율 $\gamma$는 $0 \le \gamma \le 1$
  - 0이면 $r_{t+1}$만 남으므로 매우 근시안적 보상액(탐욕 방법이 됨), 1이면 식 (9.10)과 같음
  - 현재 순간에서 멀어질수록 보상을 할인하여 공헌도를 낮추는 셈(예, $\gamma = 0.9$이면 50만큼 지난 순간은 $0.9^{50} = 0.52\%$만 남음)

# 9.2.2 가치함수

- 가치함수 추정을 위한 순환식
  - 식 (9.9)는 모든 경로를 나열하는 방식이라 그대로 적용은 현실성이 없는데, 순환 구조를 이용하여 순환식으로 바꾸어 쓸 수 있음
  - 예, 정책 $\pi_5$에서 상태 11의 가치함숫값 $v_{\pi 5}(11)$은

$$v_{\pi 5}(11) = \frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5)$$
$$+\frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}\frac{1}{2}(-1-1-1+5) + \frac{1}{2}\frac{1}{2}(-1-1-1+5)$$

$$\Downarrow$$

$$v_{\pi 5}(11) = \frac{1}{2}\left(-1 + v_{\pi 5}(7)\right) + \frac{1}{2}\left(-1 + v_{\pi 5}(10)\right)$$

  - 일반화하면, ($s$에서 $a$를 실행했을 때 $s'$로 전환)

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} P(a|s)(r + v_\pi(s')),\ \forall s \in \mathcal{S} \qquad (9.12)$$

벨만 수식

# 9.2.2 가치함수

- 현재까지는 결정론적 프로세스를 다룸
  - 어떤 상태에서 특정 행동을 취하면 항상 같은 상태로 전환하고 같은 보상을 받음
  - 예, 격자 세계 예제, 바둑, 장기 등

- 스토캐스틱 프로세스
  - 어떤 상태에서 특정 행동을 취하면 확률에 따라 다른 상태로 전환하고 다른 보상을 받음
  - 예, 격자 세계에서 6번 지점은 가끔 바람이 불어 동쪽이라는 행동을 선택했을 때 0.9 확률로 상태 7, 0.05 확률로 11, 0.05 확률로 10으로 전환

# 9.2.2 가치함수

- 스토캐스틱 프로세스에서 가치함수 추정

벨만 수식

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} P(a|s) \sum_{s'} \sum_{r} P(s', r|s, a)\big(r + v_\pi(s')\big), \forall s \in \mathcal{S} \quad (9.13)$$

  - $P(s', r|s, a)$는 스토캐스틱 정보를 표현(이 확률분포는 식 (9.4)의 MDP)

- 영구 과업인 경우는,

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} P(a|s) \sum_{s'} \sum_{r} P(s', r|s, a)\big(r + \gamma v_\pi(s')\big), \forall s \in \mathcal{S} \quad (9.14)$$

- 상태 가치함수 $v$와 상태-행동 가치함수 $q$
  - 식 (9.13)과 식 (9.14)는 상태의 함수이므로 상태 가치함수라 부름
  - 식 (9.15)는 상태와 행동의 함수이므로 상태-행동 가치함수라 부름

$$q_\pi(s, a) = \sum_{s'} \sum_{r} P(s', r|s, a)\big(r + \gamma v_\pi(s')\big), \forall s \in \mathcal{S}, \forall a \in \mathcal{A} \quad (9.15)$$

  - 두 가치함수 사이에 $v_\pi(s) = \sum_{a \in \mathcal{A}(s)} p(a|s) q_\pi(s, a)$가 성립

# 9.2.3 최적 가치함수

- 최적 정책을 찾는 식 (9.8)을 가치함수를 이용하여 다시 쓰면,

$$\hat{v}(s) = v_{\hat{\pi}}(s) = \max_{\pi} v_{\pi}(s), \forall s \in \mathcal{S} \tag{9.16}$$

  - 이들 식은 선언적 의미일 뿐 실제 계산 방법은 제시하지 않음

- 계산이 가능하도록 바꿔 쓰면,

$$\hat{v}(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \sum_{r} P(s', r|s, a)\left(r + \hat{v}(s')\right), \forall s \in \mathcal{S} \tag{9.17}$$

  - 식 (9.17)은 식 (9.13)의 평균 연산자를 max 연산자로 바꾼 것

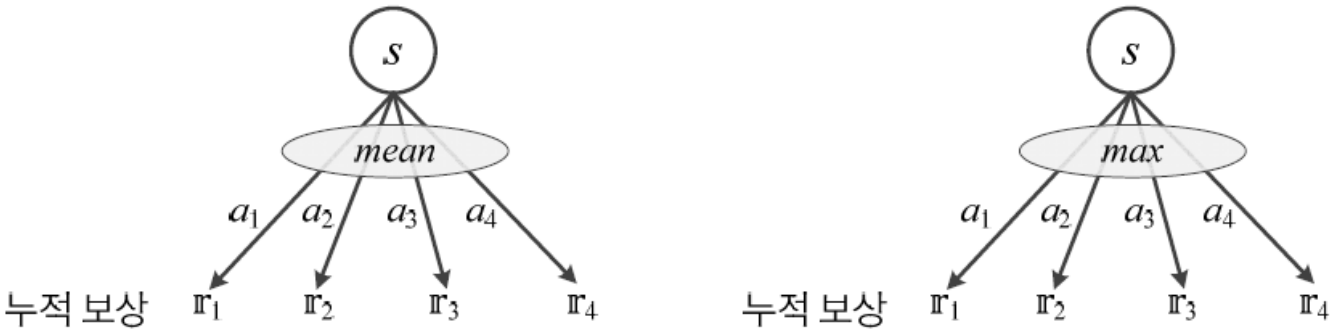



**그림 9-10 가치함수 계산과 최적 가치함수의 계산**

# 9.2.3 최적 가치함수

- 영구 과업으로 바꿔 쓰면,

$$\hat{v}(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \sum_{r} P(s', r|s, a)\left(r + \gamma \hat{v}(s')\right), \quad \forall s \in \mathcal{S} \tag{9.18}$$

- 식 (9.17)과 식 (9.18)을 어떻게 계산하나?
  - 상태들이 순환적으로 연결되어 있는데, 어느 상태도 값이 정해져 있지 않음
  - 이들을 푸는 보편적 방법
    - 정책을 임의로 설정하고 출발 → 가치함수 계산 → 정책 개선 → 가치함수 계산 → …
    - 9.3절(동적 프로그래밍), 9.4절(몬테카를로 방법), 9.5절(시간차 학습)은 이 방식을 사용

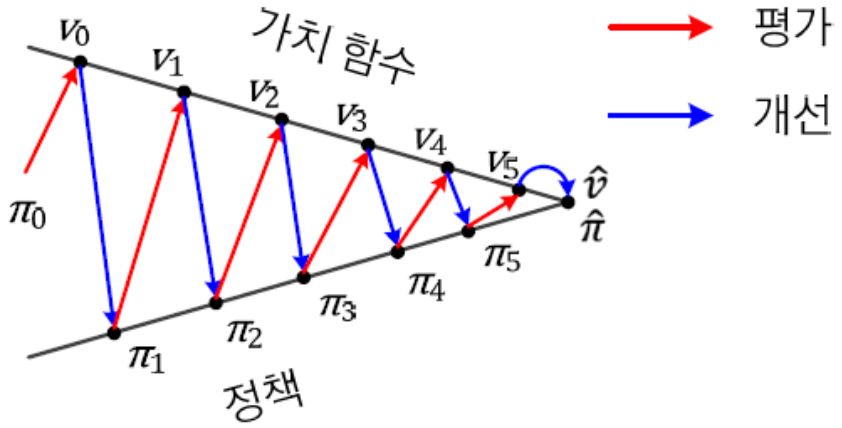



그림 9-11 강화 학습이 최적 정책을 찾아가는 보편적 과정

# 9.3 동적 프로그래밍

- 9.3.1 정책 반복 알고리즘
- 9.3.2 가치 반복 알고리즘

- 강화 학습에서 동적 프로그래밍
  - 초창기 강화 학습이 사용한 고전적 방법
  - MDP 확률분포가 주어져야 하며, 상태와 행동의 개수가 적어야 한다는 강한 제약 조건 하에서 작동
  - 현대적 알고리즘의 토대를 이루는 핵심 개념과 아이디어 제공

# 9.3 동적 프로그래밍

## 문제해결 방법론으로서 동적 프로그래밍

- 모든 쌍 최단 경로 찾기all-pair shortest path, 행렬 곱셈 순서matrix chain multiplication 문제 등을 효율적으로 푸는 문제해결 방법론(상향식 방법론)
- 예, 행렬 곱셈 순서 문제
  - 가장 작은 단위의 부분 문제로 분해한 후, 2개 행렬의 답을 구하여 표에 기록, 표를 보고 3개 조합에 대한 답을 구하여 표에 기록, 표를 보고 4개 조합에 대한 답을 구하여 표에 기록하는 과정을 반복
- $i$번째 단계의 답을 가지고 $i+1$번째 답을 구하기 위한 순환식 필요

## 강화 학습에서는 스토캐스틱 동적 프로그래밍

- 식 (9.13)이나 식 (9.17)과 같은 확률을 포함한 순환식을 사용

# 9.3.1 정책 반복 알고리즘

- 정책 반복 알고리즘이 사용하는 평가-개선 사이클
  - 평가 단계: 현재 정책에서 가치함수를 계산
  - 개선 단계: 가치함수를 이용하여 정책을 갱신


그림 9-12 정책 반복 알고리즘

**알고리즘 9-1 정책 반복**

**입력:** MDP, 즉 $P(s', r|s, a), \forall s \in \mathcal{S}, \forall s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}$, 임곗값 $\varepsilon$
**출력:** 최적 정책 $\hat{\pi}$과 최적 가치함수 $\hat{v}$

1 $t=0$
2 정책 $\pi_t$를 임의값으로 초기화한다.
3 repeat
4 　　정책 $\pi_t$에서 가치함수 $v_t$를 계산한다. // 평가
5 　　$v_t$로 $\pi_t$를 개선하여 $\pi_{t+1}$이라 한다. // 개선
6 　　라인 5에서 개선된 양을 Δ라 한다.
7 　　$t$++
8 until($\Delta < \varepsilon$)
9 $\hat{\pi} = \pi_t, \hat{v} = v_{t-1}$

# 9.3.1 정책 반복 알고리즘

- 정책 반복 알고리즘의 과다한 시간 소요
  - 라인 4의 복잡성 등이 요인

라인 4:

repeat

for 상태 각각에 대해

정책 $\pi_t$에 따라 가치함수 $v_t$를 갱신한다.

until (개선된 양이 충분히 작음)

# 9.3.2 가치 반복 알고리즘

- 가치 반복 알고리즘의 전략
  - MDP를 보고 최적 가치함수를 찾은 후, 최적 가치함수로부터 최적 정책을 찾음

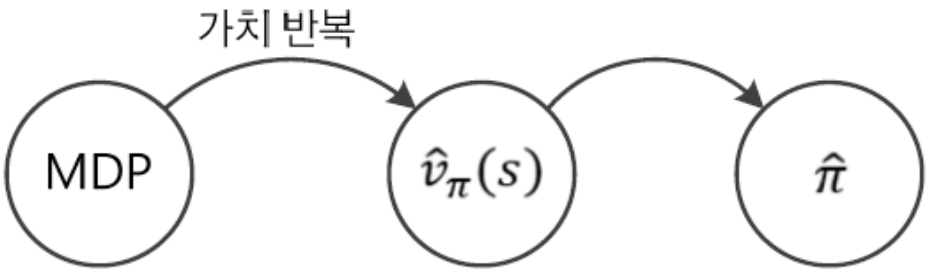



그림 9-13 가치 반복 알고리즘

- MDP에 따라 가치함수를 더 좋은 가치함수로 갱신하는 식 (9.19)
  - 식 (9.17)을 개조하여 얻음

$$v^{(t+1)}(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \sum_{r} P(s', r|s, a)(r + v^{(t)}(s')), \forall s \in \mathcal{S} \qquad (9.19)$$

# 9.3.2 가치 반복 알고리즘

**알고리즘 9-2 가치 반복**

**입력:** MDP, 즉 $P(s',r|s,a), \forall s \in \mathcal{S}, \forall s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}$, 임곗값 $\varepsilon$
**출력:** 최적 정책 $\hat{\pi}$과 최적 가치함수 $\hat{v}$

1 for $(s \in \mathcal{S})$ $v^{(0)}(s) = 0$ // 초기화 (0 또는 임의값으로 설정함)
2 $t$=0
3 repeat
4 $max\_current$=0
5 for $(s \in \mathcal{S})$
6 $v^{(t+1)}(s) = \max_{a \in \mathcal{A}(s)} \sum_{s'} \sum_{r} P(s',r|s,a)(r + v^{(t)}(s'))$ // 식 (9.19)
7 라인 6에서 max였던 행동집합을 $A(s)$에 대입한다.
8 $max\_current$=max($max\_current$, $v^{(t+1)}(s)$-$v^{(t)}(s)$)
9 $t$++
10 until($max\_current < \varepsilon$)
11 $\hat{v} = v^{(t)}$
12 for $(s \in \mathcal{S})$ // $A(s)$ 정보를 가지고 최적 정책 구축
13 $\sum_{a \in A(s)} P(a|s) = 1$이 되도록 $A(s)$에 속한 행동 $a$에 확률을 배분한 후 확률을 $\hat{\pi}$에 저장한다.

# 9.3.2 가치 반복 알고리즘

**예제 9-4** **격자 세계 문제에 가치 반복 알고리즘 적용**

[예제 9-2]와 같은 MDP를 사용한다고 가정한다. [그림 9-14]의 $t$=0은 [알고리즘 9-2]의 라인 1에 의한 초기화 결과이다. 모든 상태를 0으로 초기화한다고 가정하자. 첫 번째 repeat 반복에서 라인 6을 상태 2와 상태 6에 적용해 보자.

$$
\begin{aligned}
v^{(1)}(2) &= \max_{a\in\{동,서,남,북\}} \sum_{s'}\sum_{r} P(s',r|2,a)(r+v^{(0)}(s')) \\
&= \max\{P(3,-1|2,동)(-1+v^{(0)}(3)), P(1,5|2,서)(5+v^{(0)}(1)), \\
&\quad P(6,-1|2,남)(-1+v^{(0)}(6)),\ P(2,-2|2,북)(-2+v^{(0)}(2))\} \\
&= \max\{1*(-1+0), 1*(5+0), 1*(-1+0), 1*(-2+0)\} = 5
\end{aligned}
$$

$$
\begin{aligned}
v^{(1)}(6) &= \max_{a\in\{동,서,남,북\}} \sum_{s'}\sum_{r} p(s',r|6,a)(r+v^{(0)}(s')) \\
&= \max\{P(7,-1|6,동)(-1+v^{(0)}(7)), P(5,-1|6,서)(-1+v^{(0)}(5)), \\
&\quad P(10,-1|6,남)(-1+v^{(0)}(10)), P(2,-1|6,북)(-1+v^{(0)}(2))\} \\
&= \max\{1*(-1+0), 1*(-1+0), 1*(-1+0), 1*(-1+0)\} = -1
\end{aligned}
$$

결국, $v^{(1)}(2)=5$와 $v^{(1)}(6)=-1$이 된다. 이 두 상태에 라인 7을 적용하면 $A(2)=\{서\}$와 $A(6)=\{동, 서, 남, 북\}$이 된다. 비슷하게 나머지 상태도 계산하면 [그림 9-14]의 $t$=1이 된다.

# 9.3.2 가치 반복 알고리즘

두 번째 반복, 즉 $t$=2를 계산하자. 상태 2와 6에 라인 6을 수행하면 다음과 같다. 결국, $v^{(2)}(2) = 5$와 $v^{(2)}(6) = 4$가 된다. 이 두 상태에 라인 7을 적용하면 $A(2) = \{서\}$와 $A(6) = \{서, 북\}$이 된다. 비슷하게 나머지 상태도 계산하면 [그림 9-14]의 $t$=2가 된다.

$$
\begin{aligned}
v^{(2)}(2) &= \max_{a \in \{동,서,남,북\}} \sum_{s'} \sum_{r} P(s', r|2, a)(r + v^{(1)}(s')) \\
&= \max\{ P(3, -1|2, 동)(-1 + v^{(1)}(3)), P(1,5|2, 서)(5 + v^{(1)}(1)), \\
&\quad P(6, -1|2, 남)(-1 + v^{(1)}(6)), P(2, -2|2, 북)(-2 + v^{(1)}(2))\} \\
&= \max\{1 * (-1 - 1), 1 * (5 + 0), 1 * (-1 - 1), 1 * (-2 + 5)\} = 5
\end{aligned}
$$

$$
\begin{aligned}
v^{(2)}(6) &= \max_{a \in \{동,서,남,북\}} \sum_{s'} \sum_{r} P(s', r|6, a)(r + v^{(1)}(s')) \\
&= \max\{ P(7, -1|6, 동)(-1 + v^{(1)}(7)), P(5, -1|6, 서)(-1 + v^{(1)}(5)), \\
&\quad P(10, -1|6, 남)(-1 + v^{(1)}(10)), P(2, -1|6, 북)(-1 + v^{(1)}(2))\} \\
&= \max\{1 * (-1 - 1), 1 * (-1 + 5), 1 * (-1 - 1), 1 * (-1 + 5)\} = 4
\end{aligned}
$$

세 번째 반복을 수행하면 [그림 9-14]의 맨 오른쪽, 즉 $t$=3이 된다. $t$=2일 때에 비해 변화가 없으므로 수렴했음을 알 수 있고 알고리즘은 멈춘다. 결국, $t$=2가 최적 가치함수와 최적 정책이 된다.

# 9.3.2 가치 반복 알고리즘

라인 12~13을 수행하여 최적 정책을 구축하자. $A(s)$에 속한 행동 $a$에 같은 확률을 배분하면 [예제 9-2]의 $\pi_4$가 된다. 예를 들어, $A(4) = \{서, 남\}$인데, 같은 확률을 부여하므로 $P(서|4) = P(남|4) = \frac{1}{2}$이다.

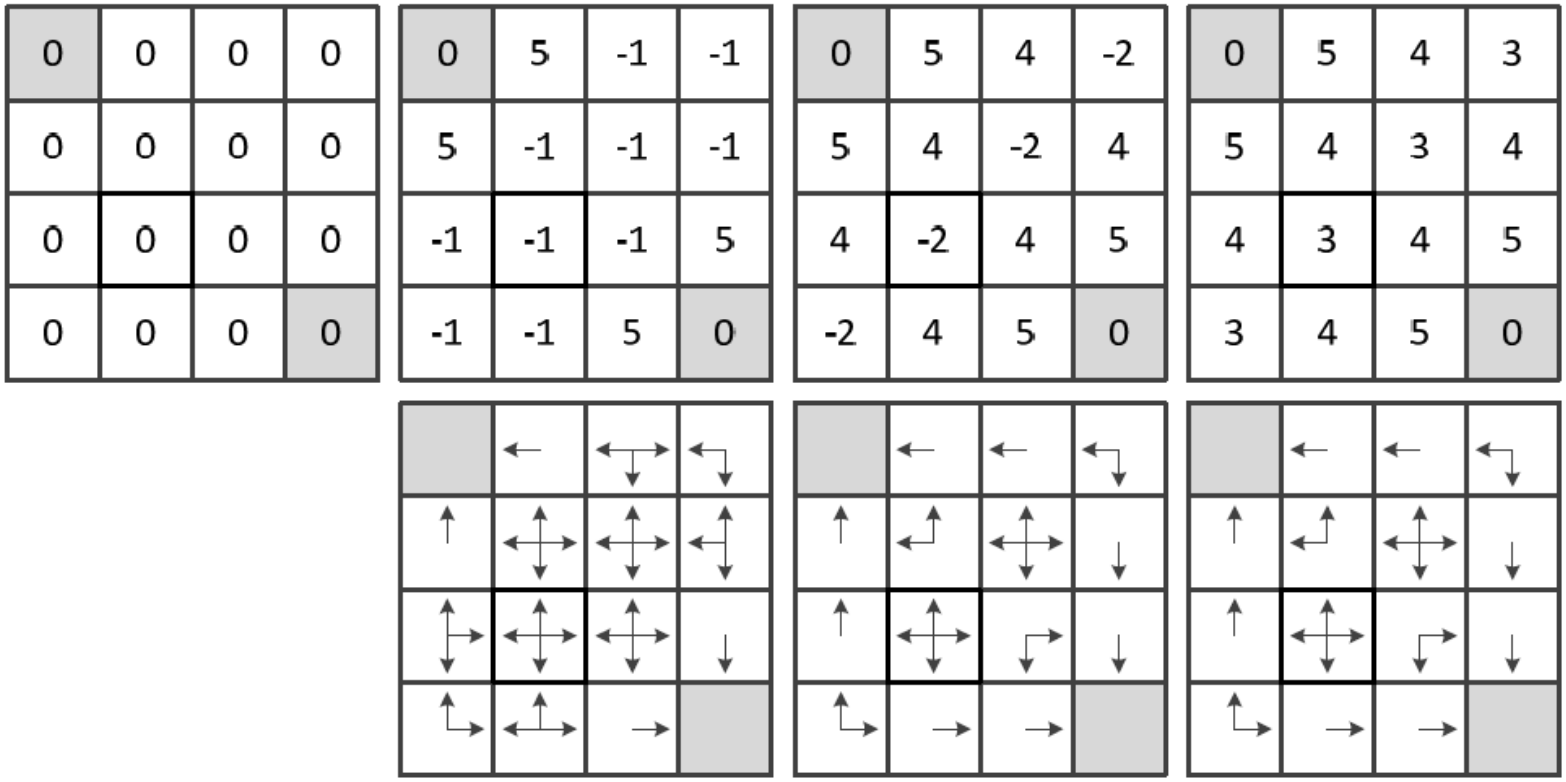



그림 9-14 가치 반복 알고리즘의 적용 예제

# 9.3.2 가치 반복 알고리즘

- [알고리즘 9-1]과 [알고리즘 9-2]는 부트스트랩 방식
  - 수렴하기 이전에는 모든 상태가 불완전한 상황 → 다른 상태의 불완전한 추측값으로 자신의 추측값을 정하는 셈
  - [그림 9-14]를 보면, 목표 지점에 인접한 상태(2, 5, 12, 15번 칸)가 가장 먼저 가치함숫값을 확정. 이 값이 이웃 상태로 전파되는 과정이 동적 프로그래밍의 핵심 원리
- 제자리 연산으로 구현
  - 부트스트랩 방식은 모든 상태가 이전 값을 이용하여 현재 값을 계산해야 공평 → 배열 하나만 써서 갱신하는 제자리 연산이 불가능하여, 두 개 배열을 번갈아 사용
  - 그럼에도 제자리 연산으로 구현하면 성능 저하 없음

# 9.3.2 가치 반복 알고리즘

- 제자리 버전
  - [알고리즘 9-2]에서 반복 수를 제어하는 변수 $t$를 제거함

**알고리즘 9-3 가치 반복(제자리 버전)**

**입력:** MDP, 즉 $P(s',r|s,a), \forall s \in \mathcal{S}, \forall s' \in \mathcal{S}, \forall a \in \mathcal{A}, \forall r \in \mathcal{R}$, 임곗값 $\varepsilon$
**출력:** 최적 정책 $\hat{\pi}$과 최적 가치함수 $\hat{v}$

```
1  for (s ∈ S) v(s) = 0   // 초기화 (0 또는 임의값으로 설정함)
2  repeat
3      max_current = 0
4      for (s ∈ S)
5          previous = v(s)
6          v(s) = max_{a∈A(s)} Σ_{s'} Σ_r P(s',r|s,a)(r + v(s'))     // 식 (9.19)
7          라인 6에서 max였던 행동집합을 A(s)에 대입한다.
8          max_current = max(max_current, v(s)-previous)
9  until(max_current < ε)
10 v̂ = v
11 for (s ∈ S)   // A(s) 정보를 가지고 최적 정책 구축
12     Σ_{a∈A(s)} P(a|s) = 1이 되도록 A(s)에 속한 행동 a에 확률을 배분한 후 확률을 π̂에 저장한다.
```

# 9.3.2 가치 반복 알고리즘

- 동적 프로그래밍의 한계
  - 상태 공간이 방대한 경우 계산 불가능
  - 예, 백가몬은 $10^{20}$ 가량의 상태가 있음
    - 상태 하나 처리하는 데 1초가 걸린다면, 모든 상태를 처리하는 데 300만 년 가량 걸림

# 9.4 몬테카를로 방법

- 9.4.1 훈련집합의 수집과 정책 평가
- 9.4.2 최적 정책 탐색

- 환경 모델에 대한 정보가 부족한 상황에서는,
  - 즉 MDP가 없는 상황(확률분포 $P(s', r|s, a)$가 주어지지 않은 상황)
  - MDP를 요구하는 [알고리즘 9-1]과 [알고리즘 9-2]의 동적 프로그래밍은 적용 불가능
  - 다행히 불완전한 모델로부터 샘플을 생성하거나 수집할 수 있는 경우 많음
    - 예, 바둑에서 이전 기보를 샘플로 활용 또는 프로그램과 프로그램을 대결시켜 샘플을 만들어 냄
  - 몬테카를로 방법은 샘플을 훈련집합으로 사용하여 최적 정책을 알아내는 '학습' 기반

# 9.4.1 훈련집합의 수집과 정책 평가

- 에피소드로부터 훈련집합 구축
  - 에피소드 예, [그림 9-4]에서 이동 경로의 사례, 바둑에서 기보 하나
  - 에피소드는 식 (9.20)처럼 표기

$$e = [s_0, r_0]a_0[s_1, r_1]a_1[s_2, r_2]a_2 \cdots [s_t, r_t]a_t \cdots [s_T, r_T] \quad (9.20)$$

  - 에피소드로부터 샘플 수집(첫 방문과 모든 방문 방식)
    - 예, 상태 10이 두 군데서 발생하는데 첫 방문은 앞에 것만 수집, 모든 방문은 둘 다 수집

$$e = [10,0]동[11,0]북[7,0]남[11,0]서[10,0]서[9,0]북[5,0]북[1,5]$$

  - 모든 방문의 수집 예

$$\begin{cases} Z(5) = \{[5,0]북[1,5]\} \\ Z(7) = \{[7,0]남[11,0]서[10,0]서[9,0]북[5,0]북[1,5]\} \\ Z(9) = \{[9,0]북[5,0]북[1,5]\} \\ Z(10) = \{[10,0]동[11,0]북[7,0]남[11,0]서[10,0]서[9,0]북[5,0]북[1,5], [10,0]서[9,0]북[5,0]북[1,5]\} \\ Z(11) = \{[11,0]북[7,0]남[11,0]서[10,0]서[9,0]북[5,0]북[1,5], [11,0]서[10,0]서[9,0]북[5,0]북[1,5]\} \end{cases}$$

# 9.4.1 훈련집합의 수집과 정책 평가

- 상태-행동 샘플을 수집하면,

$$
\begin{cases}
Z(5, \text{북}) = \{[5,0]\text{북}[1,5]\} \\
Z(7, \text{남}) = \{[7,0]\text{남}[11,0]\text{서}[10,0]\text{서}[9,0]\text{북}[5,0]\text{북}[1,5]\} \\
Z(9, \text{북}) = \{[9,0]\text{북}[5,0]\text{북}[1,5]\} \\
Z(10, \text{동}) = \{[10,0]\text{동}[11,0]\text{북}[7,0]\text{남}[11,0]\text{서}[10,0]\text{서}[9,0]\text{북}[5,0]\text{북}[1,5]\} \\
Z(10, \text{서}) = \{[10,0]\text{서}[9,0]\text{북}[5,0]\text{북}[1,5]\} \\
Z(11, \text{북}) = \{[11,0]\text{북}[7,0]\text{남}[11,0]\text{서}[10,0]\text{서}[9,0]\text{북}[5,0]\text{북}[1,5]\} \\
Z(11, \text{서}) = \{[11,0]\text{서}[10,0]\text{서}[9,0]\text{북}[5,0]\text{북}[1,5]\}
\end{cases}
$$

- 실제 수집 방법
  - 현장에서 수집
    - 예, 바둑 기보 수집(알파고는 16만 개 기보에서 대략 3천만 개 샘플을 수집) 또는 두 개 프로그램을 대결시켜 기보 생성
  - 시뮬레이션
    - MDP가 있는 경우 확률분포에 따라 에피소드 생성

# 9.4.1 훈련집합의 수집과 정책 평가

- 훈련집합으로 정책 평가
  - 훈련집합 $Z$로 가치함수를 추정하는 식

$$v_\pi(s) = \frac{1}{|Z(s)|} \sum_{z \in Z(s)} \mathrm{r}(z), \quad \forall s \in \mathcal{S} \tag{9.21}$$

$$q_\pi(s,a) = \frac{1}{|Z(s,a)|} \sum_{z \in Z(s,a)} \mathrm{r}(z), \quad \forall s \in \mathcal{S}, \quad \forall a \in \mathcal{A} \tag{9.22}$$

**알고리즘 9-4 정책 평가를 위한 몬테카를로 방법(상태-행동 가치함수 버전)**

**입력:** 정책 $\pi$, 에피소드 발생기
**출력:** 가치함수 $q_\pi$

```
1  for (s ∈ S와 a ∈ A) q_π(s,a) = 0  // 가치함수를 0으로 초기화
2  repeat
3     π에 따라, 에피소드 e를 생성한다.
4     e에서 상태-행동 샘플을 수집하여 Z(s,a)에 추가한다.
5     for (라인 4에서 Z(s,a)가 변경된 s,a 쌍에 대해)
6        q_π(s,a) = 1/|Z(s,a)| Σ_{z∈Z(s,a)} r(z)      // 식 (9.22)
7  until(멈춤 조건)
```

# 9.4.1 훈련집합의 수집과 정책 평가

- 몬테카를로 방법과 동적 프로그래밍의 비교

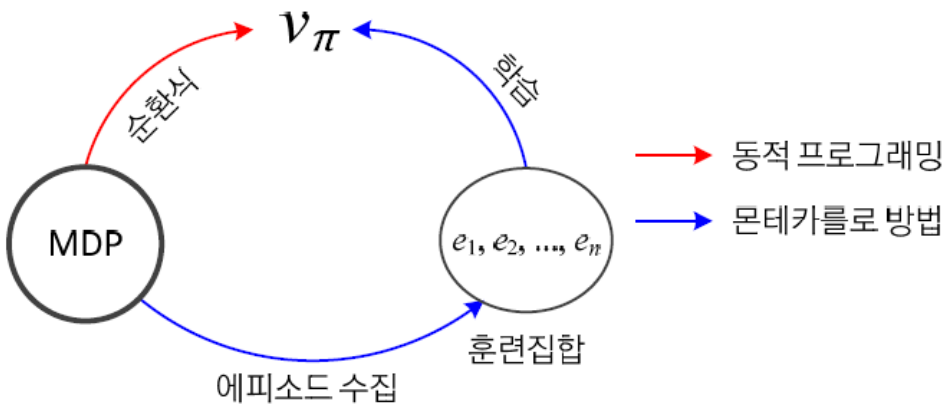



그림 9-15 동적 프로그래밍과 몬테카를로 방법의 동작 비교

  - 몬테카를로는 훈련집합을 사용하므로 특정 상태에 대해서만 가치함수 계산이 가능

# 9.4.2 최적 정책 탐색

## 탐험과 탐사 조절

- 학습 과정에서 탐사에 치우칠 위험
  - 어떤 상태에서 우연히 열등한 행동이 발생하였다면 이후 그 행동만 선택할 가능성([그림 9-6]의 $k$-손잡이 밴딧 문제의 사례)
- 탐사에 따른 조기 수렴을 피하는 2가지 방법
  - 탐험형 시작exploring starts: 상태-행동 쌍이 골고루 발생하도록 배려
  - $\epsilon$-소프트: 주류에서 벗어난 상태 행동에 일정한 확률을 배정하여 선택 가능성 열어둠

# 9.4.2 최적 정책 탐색

**알고리즘 9-5 최적 정책 탐색을 위한 몬테카를로 방법(탐험형 시작)**

**입력:** 에피소드 발생기

**출력:** 최적 정책 $\hat{\pi}$, 최적 상태-행동 가치함수 $\hat{q}$

1 for ($s \in S$ and $a \in A$) $Z(s,a) = \emptyset$, $q(s,a) = 0$ // 샘플집합을 공집합, 가치함수를 0 초기화
2 정책 $\pi$를 임의로 초기화한다.
3 repeat
4 탐험형 시작을 적용하여, 상태 $s_0$과 행동 $a_0$을 발생한다.
5 $\pi$에 따라, $s_0$과 $a_0$에서 시작하는 에피소드 $e$를 생성한다.
6 $e$에서 샘플을 수집하여 훈련집합 $Z(s,a)$에 추가한다.
7 for (라인 6에서 $Z(s,a)$가 변한 $s$, $a$ 쌍에 대해) // 정책 평가
8 $q(s,a) = \frac{1}{|Z(s,a)|}\sum_{z \in Z(s,a)} r(z)$ // 식 (9.22)
9 for ($e$에 있는 $s$에 대해) // 정책 개선
10 $A(s) = \arg\max_a q(s,a)$
11 until(멈춤 조건)
12 $\hat{q} = q$
13 for ($s \in S$) // $A(s)$ 정보를 가지고 최적 정책 구축
14 $\sum_{a \in A(s)} P(a|s) = 1$이 되도록 $A(s)$에 속한 행동 $a$에 확률을 배분한 후 확률을 $\hat{\pi}$에 저장한다.

# 9.4.2 최적 정책 탐색

**알고리즘 9-6 최적 정책 탐색을 위한 몬테카를로 방법($\varepsilon$-소프트)**

**입력:** 에피소드 발생기, 확률 배분 비율 $\epsilon$
**출력:** 최적 정책 $\hat{\pi}$, 최적 상태-행동 가치함수 $\hat{q}$

1 for($s \in \mathcal{S}$ and $a \in \mathcal{A}$) $Z(s,a) = \emptyset$, $q(s,a) = 0$ // 샘플집합을 공집합, 가치함수를 0 초기화
2 정책 $\pi$를 임의로 초기화한다.
3 repeat
4 $\pi$에 따라 에피소드 $e$를 생성한다.
5 $e$에서 샘플을 수집하여 훈련집합 $Z(s,a)$에 추가한다.
6 for(라인 5에서 $Z(s,a)$가 변한 $s$, $a$ 쌍에 대해) // 정책 평가
7 $q(s,a) = \frac{1}{|Z(s,a)|}\sum_{z \in Z(s,a)} r(z)$ // 식 (9.22)
8 for($e$에 있는 $s$에 대해)
9 $a' = \arg\max_a q(s,a)$
10 $A(s) = a'$
11 for ($a \in \mathcal{A}(s)$)
12 $$P(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}, & a = a' \text{이면} \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & a \neq a' \text{이면} \end{cases}$$ // $\varepsilon$-소프트
13 until(멈춤 조건)
14 $\hat{q} = q$
15 for ($s \in \mathcal{S}$) // $A(s)$ 정보를 가지고 최적 정책 구축
16 $\sum_{a \in A(s)} P(a|s) = 1$이 되도록 $A(s)$에 속한 행동 $a$에 확률을 배분한 후 확률을 $\hat{\pi}$에 저장한다.

# 9.4.2 최적 정책 탐색

- 몬테카를로 방법의 장점
  - 환경 모델이 없어도 적용 가능(에피소드를 수집할 수 있다면)
  - 부트스트랩 방식이 아니므로 관심 있는 상태에 대해서만 최적 가치와 최적 정책을 추정할 수 있음
  - 마르코프 성질에서 크게 벗어나도 적용 가능(에피소드가 문제의 특성을 반영함)

# 9.5 시간차 학습

- 9.5.1 정책 평가
- 9.5.2 Sarsa
- 9.5.3 Q-학습

- 시간차 학습은 동적 프로그래밍과 몬테카를로 방법의 중간
  - 동적 프로그래밍은 MDP 확률분포를 사용하며 부트스트랩 방식
  - 몬테카를로는 MDP 확률분포 대신 훈련집합을 사용하며 부트스트랩 방식이 아님

  → 시간차 학습은 훈련집합을 사용하며 부트스트랩 방식(동적 프로그래밍과 몬테카를로의 장점을 겸비한 셈)

# 9.5.1 정책 평가

- 새로운 샘플로 가치함수를 갱신하는 수식 유도
  - 에피소드 $e = [s_0, r_0]a_0[s_1, r_1]a_1 \cdots [s_t, r_t]a_t \cdots [s_T, r_T]$ 에서 샘플 $z_t = [s_t, r_t]a_t \cdots [s_T, r_T]$를 처리하는 절차
    - 이 샘플을 $Z(s_t)$에 추가한 다음
    - 다음 식을 적용하여 가치함수를 갱신

$$v_\pi(s_t) = \frac{1}{|Z(s_t)|} \sum_{z \in Z(s_t)} \mathbb{r}(z)$$

  - 샘플 $z_t$가 $k$번째로 추가되었다면 다음 식이 성립
    - $v_\pi^{(old)}$는 직전까지 가치함숫값, 즉 먼저 추가된 $k$-1개 샘플의 평균

$$v_\pi^{(new)}(s_t) = \frac{v_\pi^{(old)}(s_t) * (k-1)}{k} + \frac{\mathbb{r}(z_t)}{k} = v_\pi^{(old)}(s_t) + \frac{1}{k}\left(\mathbb{r}(z_t) - v_\pi^{(old)}(s_t)\right)$$

# 9.5.1 정책 평가

- 간략한 표기로 식을 다시 쓰면,

$$v_\pi(s_t) = v_\pi(s_t) + \rho\left(\mathbb{r}_{new} - v_\pi(s_t)\right) \tag{9.23}$$

  - 식 (9.21)의 몬테카를로 방법은 종료 상태 $s_T$에 도달해야만 갱신이 일어나는데, 식 (9.23)을 이용하여 매 순간 갱신할 수 있나? ← 시간차 학습의 핵심 아이디어
  - 식 (9.23)에서는 $\mathbb{r}_{new}$ 때문에 매 순간 갱신 불가능
  - 그런데 식 (9.12)~(9.15)에서 사용한 순환 관계를 적용하면 식 (9.24)로 바꿔 쓸 수 있음(매 순간 갱신하는 시간차 아이디어 실현됨)

$$v_\pi(s_t) = v_\pi(s_t) + \rho\left(\left(r_{t+1} + \gamma v_\pi(s_{t+1})\right) - v_\pi(s_t)\right) \tag{9.24}$$

  - 상태-행동 버전으로 바꿔 쓰면,

$$q_\pi(s_t, a_t) = q_\pi(s_t, a_t) + \rho\left(\left(r_{t+1} + \gamma q_\pi(s_{t+1}, a_{t+1})\right) - q_\pi(s_t, a_t)\right) \tag{9.25}$$

# 9.5.1 정책 평가

**알고리즘 9-7 정책 평가를 위한 시간차 학습(상태 가치함수 버전)**

**입력:** 정책 $\pi$, 에피소드 발생기

**출력:** 가치함수 $v_\pi$

```
1  for (s ∈ S) v_π(s) = 0  // 가치함수를 0으로 초기화
2  repeat
3      에피소드 e를 생성한다.  // 식 (9.20)
4      for (t=0 to T-1)
5          v_π(s_t) = v_π(s_t) + ρ(r_{t+1} + γv_π(s_{t+1}) − v_π(s_t))   // 식 (9.24)
6  until(멈춤 조건)
```

- 라인 5는 라인 3이 만든 샘플을 사용하므로 학습 기반이고 다른 상태의 값을 사용하므로 부트스트랩 방식 → 동적 프로그래밍과 몬테카를로 방법의 장점을 겸비

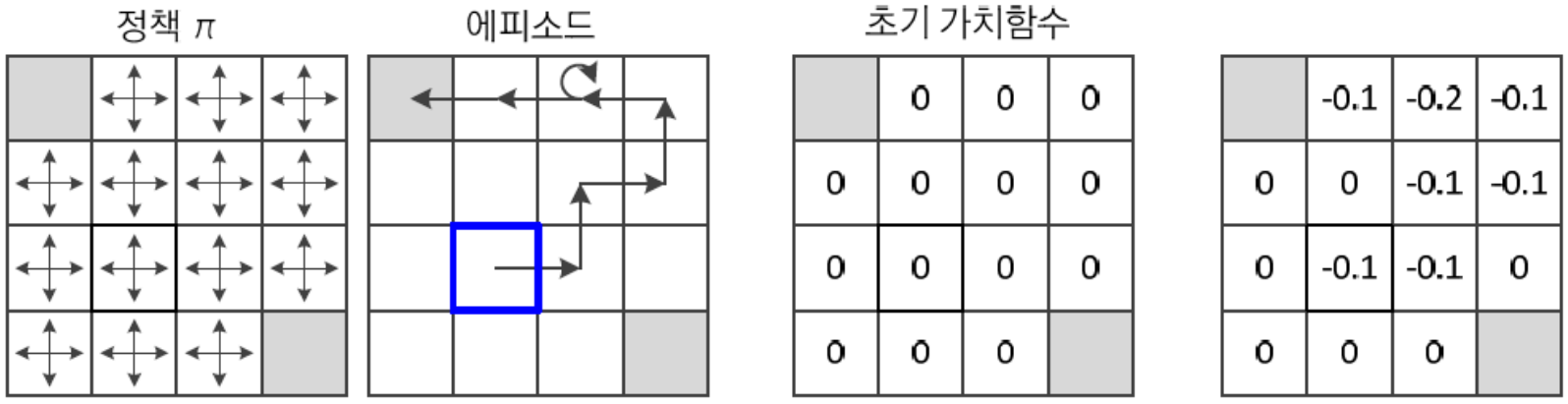
# 9.5.1 정책 평가

**예제 9–5** **격자 세계 문제에 시간차 학습 적용**

[그림 9-16]은 격자 세계의 예제이다. 모든 방향으로 동일 확률로 전이하고 보상 -1을 받는 정책을 사용한다고 가정하자. 1번과 16번 칸은 보상 5를 받고 종료하는 상태이다. [그림 9-16(a)]는 이러한 정책 $\pi$를 보여 준다. 학습률 $\rho = 0.1$이고 할인율 $\gamma = 1.0$이라고 가정하자.

[알고리즘 9-7]에서 루프를 형성하는 라인 2~6의 첫 번째 반복에서 다음 에피소드가 발생했다고 하자. [그림 9-16(a)]는 이 에피소드를 보여 주고, [그림 9-16(b)]는 라인 1이 초기화한 가치함수를 보여 준다.

$$e = [10,0]동[11,-1]북[7,-1]동[8,-1]북[4,-1]서[3,-1]북[3,-1]서[2,-1]서[1,5]$$




(a) 정책과 예제 에피소드 (b) 가치함수 초기화 (c) 시간차 적용

**그림 9–16 격자 세계에 시간차 학습을 적용한 예제**

# 9.5.1 정책 평가

라인 4에서 $t = 0$일 때 라인 5를 수행하면 다음과 같다.

$$v_\pi(10) = v_\pi(10) + 0.1\big(-1 + 1 * v_\pi(11) - v_\pi(10)\big) = -0.1$$

$t = 1$일 때 라인 5를 수행하면 다음과 같다.

$$v_\pi(11) = v_\pi(11) + 0.1\big(-1 + 1 * v_\pi(7) - v_\pi(11)\big) = -0.1$$

에피소드가 종료될 때까지 적용하면 [그림 9-16(c)]가 된다. 아직 충분히 수렴하지 않았으므로 다른 에피소드를 생성하여 같은 과정을 반복한다.

# 9.5.2 Sarsa

- 최적 정책을 찾는 Sarsa 알고리즘

**알고리즘 9-8 최적 정책을 탐색하는 Sarsa**

**입력:** 에피소드 발생기
**출력:** 최적 상태-행동 가치함수 $\hat{q}$

```
1  for(s ∈ S and a ∈ A) 임의값으로 q(s,a)를 설정한다. s가 종료 상태이면 q(s,a) = 0
2  repeat
3      초기 상태 s를 생성한다.
4      q에 따라, 상태 s에서 행동 a를 생성한다. 필요하다면 ε-소프트를 적용한다.
5      repeat
6          a를 실행하여 다음 상태 s'와 보상 r을 얻는다.  // q가 MDP의 역할 대행함
7          q에 따라, 상태 s'에서 행동 a'를 생성한다. 필요하다면 ε-소프트를 적용한다.
8          q(s,a) = q(s,a) + ρ(r + γq(s',a') − q(s,a))    // 식 (9.25)
9          s = s', a = a'
10     until(s가 종료 상태)
11 until(멈춤 조건)
12 q̂ = q
```

# 9.5.3 Q-학습

- Q-학습이 사용하는 수식

$$q_\pi(s_t, a_t) = q_\pi(s_t, a_t) + \rho\left((r_{t+1} + \gamma \max_a q_\pi(s_{t+1}, a)) - q_\pi(s_t, a_t)\right) \quad (9.26)$$

  - 식 (9.25)와 다른 점은 $q_\pi(s_{t+1}, a_{t+1})$이 $\max_a q_\pi(s_{t+1}, a)$로 바뀐 것

- 켜진 정책 방식과 꺼진 정책 방식
  - Sarsa는 정책을 사용하므로 켜진 정책 방식이라 부르고, Q-학습은 정책 대신 max 연산을 사용하므로 꺼진 정책 방식이라 부름

# 9.5.3 Q-학습

**알고리즘 9-9 Q-학습(상태-행동 버전 TD 제어)**

**입력:** 에피소드 발생기
**출력:** 최적 정책 $\hat{\pi}$, 최적 상태-행동 가치함수 $\hat{q}$

```
1  for(s ∈ S and a ∈ A) 임의값으로 q(s,a)를 설정한다. s가 종료 상태이면 q(s,a) = 0
2  repeat
3      초기 상태 s를 생성한다.
4      repeat
5          q에 따라, 상태 s에서 행동 a를 생성한다. 필요하다면 ε-소프트를 적용한다.
6          a를 실행하여 다음 상태 s'와 보상 r을 얻는다.    // MDP의 역할
7          q(s,a) = q(s,a) + ρ(r + γ max_a' q(s',a') − q(s,a))    // 식 (9.26)
8          s = s'
9      until(s가 목표 상태)
10 until(멈춤 조건);
11 q̂ = q
```

# 9.6 근사 방법

## 참조표 방식

- 9.3~9.5절의 알고리즘은 모두 가치함수 $v$ 또는 $q$를 표에 저장하는 참조표 방식([알고리즘 9-1~9.9]의 출력은 $\hat{v}$ 또는 $\hat{q}$)
- $v$와 $q$를 저장하려면 각각 $|\mathcal{S}|$와 $|\mathcal{S}| \times |\mathcal{A}|$ 크기의 배열이 필요
  - $|\mathcal{S}|$는 상태의 개수로서 백가몬 게임의 경우 $10^{20}$개 가량 → 참조표 방식이 현실적으로 불가능한 문제가 많음

## 근사 방식

- 선형 방정식(식 (9.27)) 또는 신경망([그림 9-17])을 사용하여 가치 함수를 근사 추정
- 신경망을 주로 사용(9.7.1절의 TD-gammon에서 구체적인 입출력 사례 제시)

$$v(s;\boldsymbol{\theta}) = \sum_{i=1}^{k} \theta_i \phi_i(s) \qquad (9.27)$$

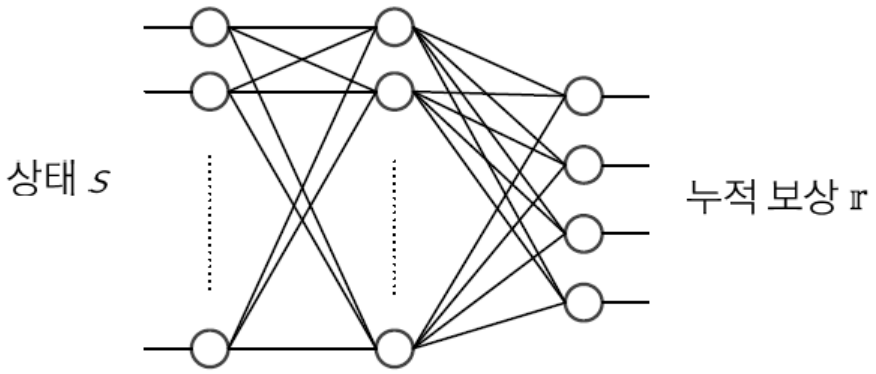



그림 9-17 근사 방법을 구현하는 신경망

# 9.7 응용 사례

- 9.7.1 TD-gammon
- 9.7.2 DQN: 아타리 비디오 게임

- 인공지능에서 이 둘의 중요한 의미
  - TD-gammon: 자가 플레이로 학습한 최초 사례
  - DQN: 같은 신경망으로 서로 다른 수십 종의 게임에서 높은 성능을 얻음

# 9.7.1 TD-gammon

- 백가몬 게임 규칙

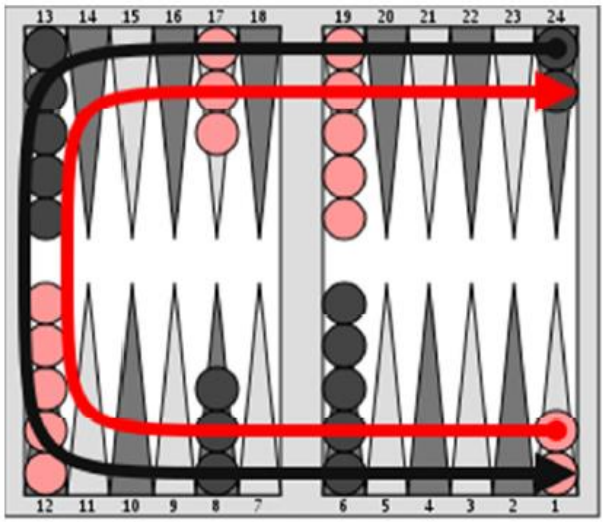



그림 9-18 백가몬 게임

[그림 9-18]은 백가몬backgammon 게임이다. 검은 말과 빨간 말이 대결하는데, 검은 말은 낮은 번호로 이동하고 빨간 말은 높은 번호로 이동한다. 15개의 검은 말이 모두 1~6번으로 이동한 다음 밖으로 빠져 나와야 한다. 빨간 말은 19~24로 이동한 후 빠져 나와야 한다. 먼저 빠져 나오는 말이 이긴다. 게임은 주사위 2개를 던져 진행된다. 현재 검은 말 차례인데 1과 5가 나왔다고 하면, 검은 말 2개를 각각 1과 5만큼 시계 반대 방향으로 이동할 수 있다. 예를 들어, 24번에 있는 말을 각각 23과 19로 옮길 수 있다. 그런데 19번 위치에 빨간 말이 5개 있으므로 19번으로 이동할 수 없다. 빨간 말이 이동을 방해하고 있다고 볼 수 있는데, 상대편을 방해하는 것은 일종의 게임 전략이다. 만약 19번에 빨간 말이 하나뿐이라면 검은 말이 빨간 말을 잡을 수 있다. 잡힌 말은 바깥에 두었다가 처음 위치에서 다시 시작해야 한다. 더 자세한 게임 규칙은 [Tesauro1995]를 참조하라.

# 9.7.1 TD-gammon

- 백가몬 프로그램
  - Tesauro가 1990년대 초에 신경망을 사용하여 NeuroGammon 제작(낮은 성능)
  - 이후 강화 학습을 사용한 TD-gammon으로 발전
- TD-gammon
  - 은닉층이 하나인 다층 퍼셉트론으로 가치함수를 근사
    - 입력층은 198개 노드
      - 검은 말과 빨간 말이 놓인 위치를 표현하는 데 2*24*4=192개 노드
      - 말의 차례를 표시하는 데 2개 노드
      - 빠져나온 말의 개수를 표현하는 데 2개 노드
      - 잡힌 말의 개수를 표현하는 데 2개 노드
    - 출력층은 4개 노드
      - 승리한 측을 표시하는 데 2개 노드
      - 대승한 측을 표시하는 데 2개 노드
  - 학습은 시간차 학습으로 해결
  - 자가 플레이로self-play 학습(사람이 사용하는 전략을 알려주지 않음)

# 9.7.1 TD-gammon

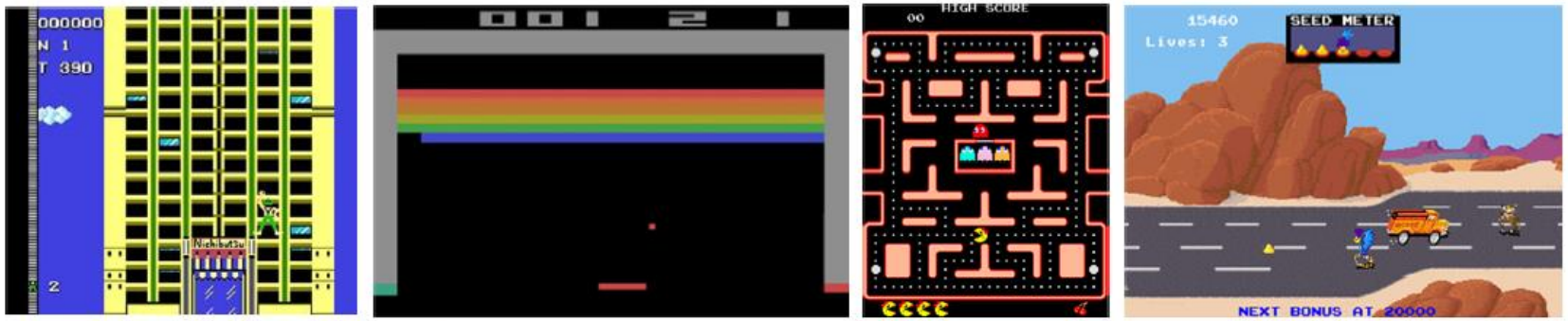
- 강화 학습과 보드게임

그림 9-19 여러 가지 보드게임

- 세계 챔피언 수준의 보드 게임 프로그램을 제작하려 많은 노력을 기울임. 그 결과 가장 어려운 보드 게임인 바둑에서 이세돌을 이기는 성과 이룩
- 기계 학습을 게임에 활용한 역사를 살피려면 [Kurenkov2016] 참조
- 틱텍토: 상태의 수가 $3^9$=19683가지이므로 간단한 추론 프로그램으로 최적 정책을 알아낼 수 있음
- 체커: 사무엘이 1950년대 컴퓨터에서 고전적인 알고리즘으로 제작[Samuel1959]
- 체스: 수를 평가하는 특수 목적용 VLSI 개발. 발전된 기계 학습 알고리즘보다는 특수 목적용 슈퍼 컴퓨터로 세계 챔피언 물리침[Campbell2002]
- 바둑: 알파고는 강화 학습과 딥러닝을 결합하여 이세돌 물리침[Silver2016]

# 9.7.2 DQN: 아타리 비디오 게임

- 아타리 비디오 게임 프로그램
  - Atari2600 기종에서 실행되는 49종 게임을 골라 자동으로 플레이하는 프로그램 개발
    - 게임마다 상태 표현과 게임 전략, 입력 방식이 크게 다름




Crazy climber Breakout Ms. Pacman Road runner

그림 9-20 여러 가지 아타리 비디오 게임

- 강화 학습을 사용한 프로그램은 동일한 입력, 동일한 신경망 구조, 동일한 하이퍼 매개변수를 사용했음에도 거의 모든 게임에서 전문 플레이어 수준을 달성
- 이전에는 사람이 게임마다 적합한 신경망 구조를 따로따로 설계한 점에 비추면, 과업간 일반화 능력이 크게 발전한 셈(사람은 몇 가지 과업에 익숙해지면 유사한 다른 과업도 잘함) ← 인공지능 발전에 기여

# 9.7.2 DQN: 아타리 비디오 게임

- DQN(deep Q-network)
  - 컨볼루션 신경망과 Q-학습을 결합한 모델
  - 입력층은 게임 스크린 영상(최근 4장의 영상 84*84*4 텐서) ← 이 텐서가 상태를 표현
  - 출력층은 18개 노드(게임마다 4~18개 조이스틱 행동이 있는데 게임마다 필요한 만큼 사용)