강화 학습
Reinforcement Learning

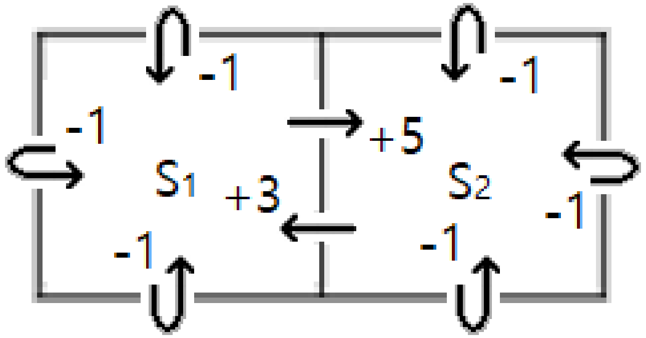
Bellman equation

- Bellman equation for v_π
  - $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$

- action-value function
  - $q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$

- state-value와 action-value의 관계는?

Example - Tiny World



- $\pi(a|s) = 1/4$
- $p(s', r|s, a) = 1$
- Bellman equation
  - $v_\pi(S_1) = \frac{1}{4} \cdot 3 \cdot [-1 + \gamma v_\pi(S_1)] + \frac{1}{4} \cdot [5 + \gamma v_\pi(S_2)]$
  - $v_\pi(S_2) = \frac{1}{4} \cdot [3 + \gamma v_\pi(S_1)] + \frac{1}{4} \cdot 3 \cdot [-1 + \gamma v_\pi(S_2)]$
- Solution
  - $v_\pi(S_1) = \frac{4-3\gamma}{4(1-\gamma)(2-\gamma)}$, $v_\pi(S_2) = \frac{\gamma}{4(1-\gamma)(2-\gamma)}$
  - $\gamma = 0.9 \Longrightarrow v_\pi(S_1) = 2.95$, $v_\pi(S_2) = 2.05$

Bellman Equation

- Bellman equation을 풀어 v_π(s)를 모두 구할 수 있다
  - 형태는 $Av + b = 0$ 꼴
  - 시간 문제
  - 메모리 문제

Optimal Policy and Optimal Value Function

- optimal value function
  - 상태의 최대 가치
  - $v_*(s) = \max_\pi v_\pi(s)$

- optimal policy
  - 가장 우수한 정책 π_*
  - $v_{\pi_*}(s) = v_*(s)$

Policy Evaluation

- 주어진 policy π에 대하여 최적의 v_π를 찾아라
- Bellman equation 풀기
  - $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')]$
- algorithm
  - $v_0 \longrightarrow v_1 \longrightarrow v_2 \longrightarrow \cdots$
  - v_0 arbitrary
  - $v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$

Policy Evaluation

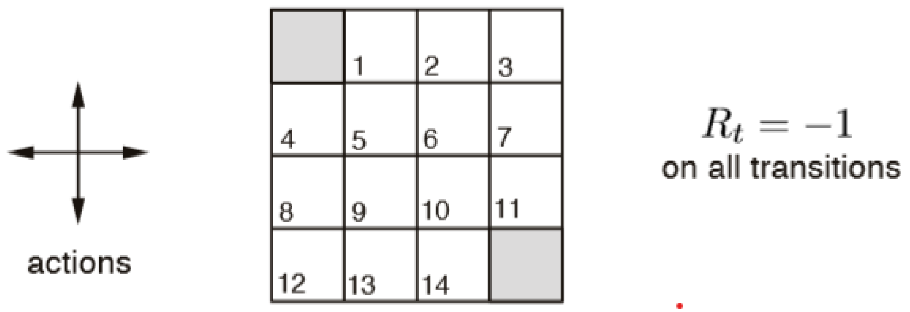
- 수학적 배경
- solve the linear equation $Ax + b = 0$
    - $x = (I + \alpha A)x + \alpha b$
    - iteration
        - x_0 arbitrary
        - $x_{k+1} = (I + \alpha A)x_k + \alpha b$
        - $x_n \longrightarrow A^{-1}b$ if $(I + \alpha A)^n \longrightarrow 0$

Policy Evaluation

- Algorithm
  - input π, the policy to be evaluated
  - initialize $v(s) = 0$ for all state s
  - repeat
    - $v_{copy} \leftarrow v$
    - for each $s \in \mathrm{S}$
      - $v(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma v_{copy}(s')\right]$
    - $\Delta \leftarrow \max_s \left|v(s) - v_{copy}(s)\right|$
  - until $\Delta < \theta$
  - output v
- Output $v \approx v_\pi$

Policy Evaluation

- Example - 4×4 gridworld



v_k for the random policy

$k = 0$

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0

$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0

$k = 3$

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

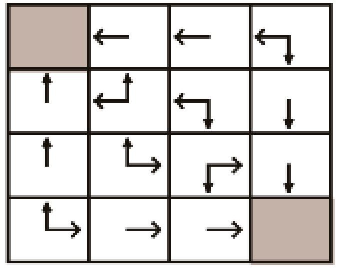
0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

Policy Improvement

- 주어진 state-value function에 대하여 가장 우수한 정책을 찾아라
- $$\pi'(s) = \underset{a}{\operatorname{argmax}}\, q_\pi(s,a)$$
  $$= \underset{a}{\operatorname{argmax}}\, \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

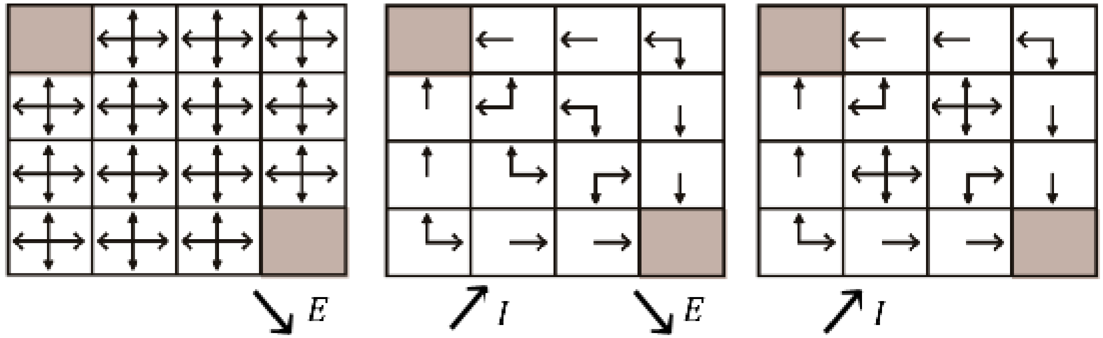
v_π



π'

Policy Iteration

- policy evaluation(E)과 policy improvement(I)를 반복
- $\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$



0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0

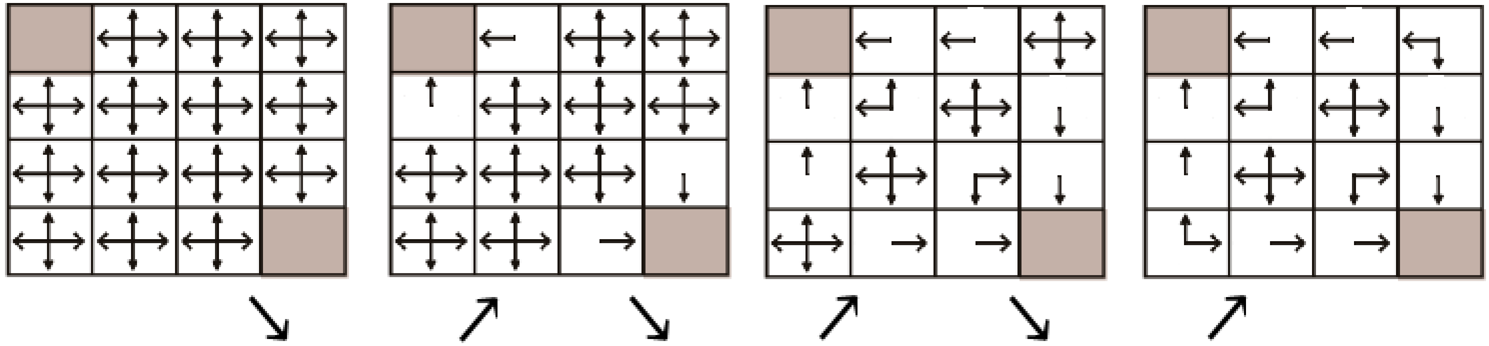
Value Iteration

- combination of policy evaluation and policy improvement
- $v_0 \longrightarrow v_1 \longrightarrow v_2 \longrightarrow \cdots$
  - $v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$

- 이론적 배경???

Value Iteration

- 4×4 gridworld



0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0

0.0	-1.0	-2.0	-2.0
-1.0	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.0
-2.0	-2.0	-1.0	0.0

0.0	-1.0	-2.0	-3.0
-1.0	-2.0	-3.0	-2.0
-2.0	-3.0	-2.0	-1.0
-3.0	-2.0	-1.0	0.0

Dynamic Programming

- 최적화 문제를 해결하는 것
  - 가장 우수한 정책(optimal policy) 찾기
  - policy iteration이나 value iteration 사용
- 장점
  - 거의 정확한 값을 계산할 수 있다
- 단점
  - 모든 경우의 수를 생각해야 한다
    - 바둑의 state의 수 ≈ 361!
    - 시간의 한계
    - 메모리의 한계

Monte Carlo Methods

- problem
  - $q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$
    - $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$
      $= R_{t+1} + \gamma G_{t+1}$
    - γ - discount rate
- compute $q_\pi(s, a)$?
  - impossible, in general, e.g. 바둑
- approximate $q_\pi(s, a)$!
  - sampling
  - Monte Carlo method

Monte Carlo Methods

- policy π에 따라 episode를 생성한다
- G_t를 계산한다
- 각 (s, a)에 대하여 $q_\pi(s, a)$의 근사값을 구한다
- policy를 다시 설정한다

- episode 복습
  - $S_0 \xrightarrow{A_0} R_1, S_1 \xrightarrow{A_1} R_2, S_2 \xrightarrow{A_2} \cdots \xrightarrow{A_{T-1}} R_T, S_T$
  - $G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-t-1} R_T$

Monte Carlo Methods

- Monte Carlo ES(exploring starts)
  - initialize, for all $s \in S$, $a \in A(s)$
    - $q(s, a) \longleftarrow$ arbitrary
    - $\pi(a|s) \longleftarrow$ arbitrary
    - $returns(s, a) \longleftarrow$ empty list
  - repeat forever
    - choose $S_0 \in S$ and $A_0 \in A(S_0)$
    - generate an episode starting from S_0, A_0, following π
    - for each pair s, a in the episode
      - $G \longleftarrow$ the return at (s, a)
      - append G to $returns(s, a)$
      - $q(s, a) \longleftarrow average(returns(s, a))$
    - $\pi(s) \longleftarrow \arg\max_a q(s, a)$

Policy

- greedy policy
  - 각 state s에서 $q(s,a)$가 가장 큰 action을 선택
  - 처음 선택된 action이 계속 선택될 가능성이 있다
- ϵ-greedy(ϵ-soft greedy) policy
  - $q(s,a)$가 가장 큰 action을 $1-\epsilon+\frac{\epsilon}{|A(s)|}$의 확률로 선택
    - $1-\epsilon$을 주고 나머지의 $\frac{1}{|A(s)|}$을 더 준다
  - 나머지 action을 $\frac{\epsilon}{|A(s)|}$의 확률로 선택

Monte Carlo Methods

- on-policy MC control for ϵ-soft policies
  - initialize, for all $s \in S$, $a \in A(s)$
    - $q(s,a) \longleftarrow$ arbitrary
    - $\pi(a|s) \longleftarrow$ arbitrary ϵ-soft policy
    - $returns(s,a) \longleftarrow$ empty list
  - repeat forever
    - generate an episode following π
    - for each pair s, a in the episode
      - $G \longleftarrow$ the return at (s,a)
      - append G to $returns(s,a)$
      - $q(s,a) \longleftarrow average(returns(s,a))$
    - for each s in the episode
      - $A^* \longleftarrow \arg\max_a q(s,a)$
      - for all $a \in A(s)$
        - $\pi(a|s) = \begin{cases} 1 - \epsilon + \epsilon/|A(s)| & if\ a = A^* \\ \epsilon/|A(s)| & if\ a \neq A^* \end{cases}$

Monte Carlo Methods

- 단점
  - $\text{returns}(s, a)$를 모두 저장해야 한다
    - 개수를 저장하면 이 문제는 해결 가능하다
      - $q(s, a) \leftarrow \frac{q(s,a)\cdot count+G_t}{count+1}$
      - $count \leftarrow count + 1$
  - episode가 끝날 때까지 기다려야 한다
    - temporal-difference learning의 출발점이다