Introduction to Reinforcement Learning

2025. 1st semester

단국대학교
DANKOOK UNIVERSITY

Mobile System Engineering

Evaluation to Control

- *Last time: how good is a specific policy?*
  - *Given no access to the decision process model parameters*
  - *Instead have to estimate from data / experience*
- *Today: how can we learn a good policy?*

Model-free Control

- Generalized policy improvement
- Importance of exploration
- Monte Carlo control
- Model-free control with temporal difference
  - (SARSA, Q-learning)

Model-free Control Examples

Some example problems that can be modelled as MDPs

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Aeroplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein Folding
- Robot walking
- Game of Go

For most of these problems, either:

- MDP model is unknown, but experience can be sampled
- MDP model is known, but is too big to use, except by samples

Model-free control can solve these problems

On and Off-Policy Learning

- *On-policy learning*
  - "*Learn on the job*"
  - *Learn about policy* π *from experience sampled from* π
  - *Learn to estimate and evaluate a policy from experience obtained from following that policy*
- *Off-policy learning*
  - "*Look over someone's shoulder*"
  - *Learn about policy* π *from experience sampled from* μ
  - *Learn to estimate and evaluate a policy using experience gathered from following a different policy*

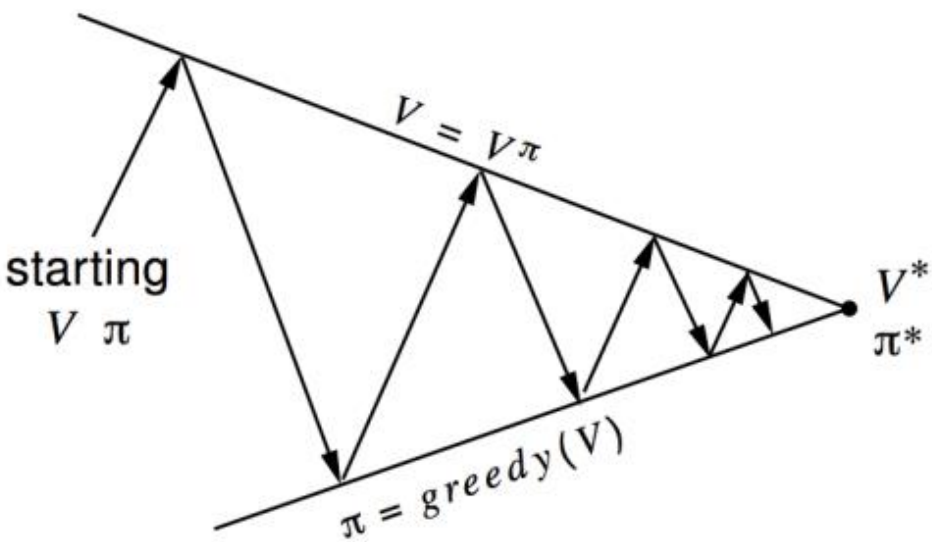
Recall Policy Iteration

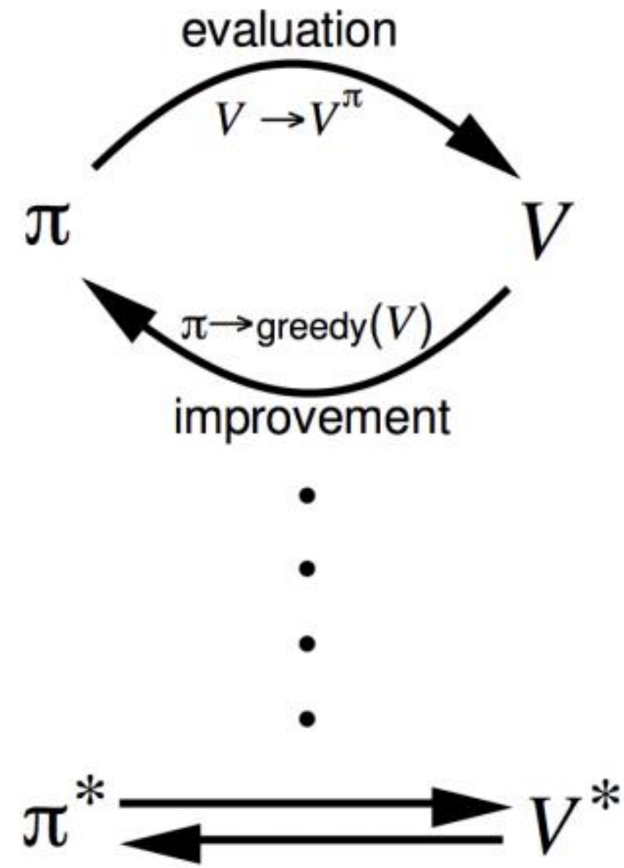
- Initialize policy π
- Repeat:
  - Policy evaluation: compute V^π
  - Policy improvement: update π

$$\pi'(s) = \arg\max_a R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^\pi(s') = \arg\max_a Q^\pi(s,a)$$

- Now want to do the above two steps without access to the true dynamics and reward models
- Last lecture introduced methods for model-free policy evaluation

Generalized Policy Iteration (Refresher)





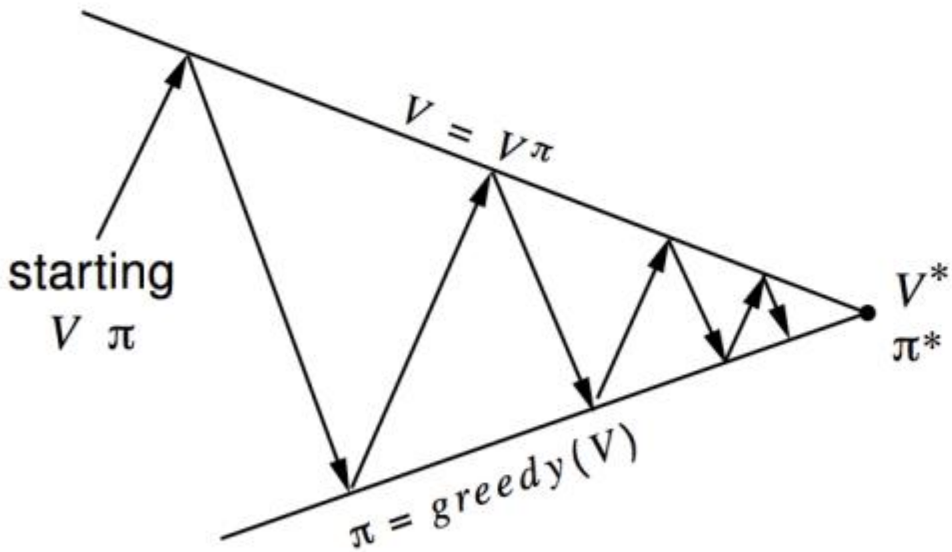
Policy evaluation Estimate v_π

e.g. Iterative policy evaluation

Policy improvement Generate $\pi' \geq \pi$

e.g. Greedy policy improvement

Generalized Policy Iteration With Monte-Carlo Evaluation



Policy evaluation Monte-Carlo policy evaluation, $V = v_\pi$?

Policy improvement Greedy policy improvement?

Model-Free Policy Iteration Using Action-Value Function

- Greedy policy improvement over $V(s)$ requires model of MDP

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \ \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

- Greedy policy improvement over $Q(s,a)$ is model-free

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

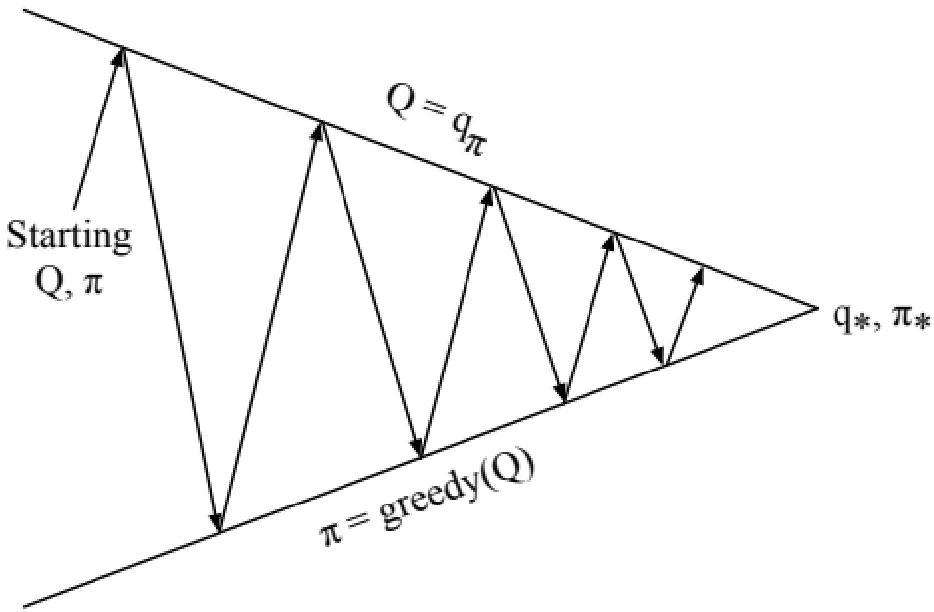
Model Free Policy Iteration

- *Initialize policy* π
- *Repeat:*
  - *Policy evaluation: compute* Q^{π}
  - *Policy improvement: update* π

- Given an estimate $Q^{\pi_i}(s, a)\ \forall s, a$
- Update new policy

$$\pi_{i+1}(s) = \arg\max_a Q^{\pi_i}(s, a)$$

Generalized Policy Iteration with Action-Value Function



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

Policy improvement Greedy policy improvement?

Model-free Policy Iteration

- Initialize policy π
- Repeat:
  - Policy evaluation: compute Q^π
  - Policy improvement: update π given Q^π

- May need to modify policy evaluation:
  - If π is deterministic, can't compute $Q(s, a)$ for any $a \neq \pi(s)$
- How to interleave policy evaluation and improvement?
  - Policy improvement is now using an estimated Q

Example of Greedy Action Selection

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- There are two doors in front of you.
- You open the left door and get reward 0
  $V(left) = 0$
- You open the right door and get reward +1
  $V(right) = +1$
- You open the right door and get reward +3
  $V(right) = +2$
- You open the right door and get reward +2
  $V(right) = +2$

$\vdots$

- Are you sure you've chosen the best door?

Policy Evaluation with Exploration

- Want to compute a model-free estimate of Q^π
- In general seems subtle
  - Need to try all (s, a) pairs but then follow π
  - Want to ensure resulting estimate Q^π is good enough so that policy improvement is a monotonic operator
- For certain classes of policies can ensure all (s,a) pairs are tried such that asymptotically Q^π converges to the true value

ϵ-Greedy Exploration

- Simple idea to balance exploration and exploitation
- Let $|A|$ be the number of actions
- Then an ϵ-greedy policy w.r.t. a state-action value $Q(s,a)$ is $\pi(a|s) = [\arg\max_a Q(s,a)$, w. prob $1-\epsilon$; a w. prob $\frac{\epsilon}{|A|}]$

$$\pi(a \mid s) = \begin{cases} \dfrac{\varepsilon}{|A|} + 1 - \varepsilon, & \text{if } a^* = \underset{a \in A}{\arg\max}\, Q(s,a) \\ \dfrac{\varepsilon}{|A|}, & \text{otherwise} \end{cases}$$

Monotonic ϵ -greedy Policy Improvement

Theorem

For any ϵ-greedy policy π_i, the ϵ-greedy policy w.r.t. Q^{π_i}, π_{i+1} is a monotonic improvement $V^{\pi_{i+1}} \geq V^{\pi}$

$$\begin{aligned} Q^{\pi_i}(s, \pi_{i+1}(s)) &= \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s, a) \\ &= (\epsilon/|A|) \sum_{a \in A} Q^{\pi_i}(s, a) + (1 - \epsilon) \max_a Q^{\pi_i}(s, a) \end{aligned}$$

- Therefore $V^{\pi_{i+1}} \geq V^{\pi}$ (from the policy improvement theorem)

Monotonic ϵ -greedy Policy Improvement

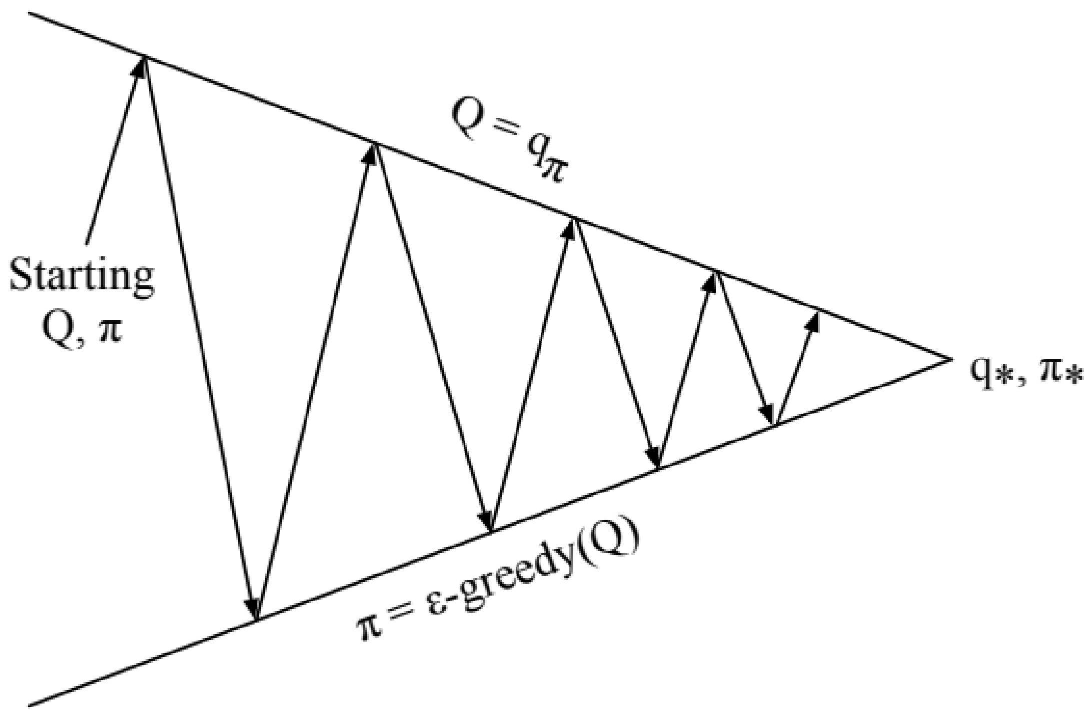
Theorem

For any ϵ-greedy policy π_i, the ϵ-greedy policy w.r.t. Q^{π_i}, π_{i+1} is a monotonic improvement $V^{\pi_{i+1}} \geq V^{\pi}$

$$
\begin{aligned}
Q^{\pi_i}(s, \pi_{i+1}(s)) &= \sum_{a \in A} \pi_{i+1}(a|s) Q^{\pi_i}(s,a) \\
&= (\epsilon/|A|) \sum_{a \in A} Q^{\pi_i}(s,a) + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \\
&= (\epsilon/|A|) \sum_{a \in A} Q^{\pi_i}(s,a) + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \frac{1-\epsilon}{1-\epsilon} \\
&= (\epsilon/|A|) \sum_{a \in A} Q^{\pi_i}(s,a) + (1-\epsilon) \max_a Q^{\pi_i}(s,a) \sum_{a \in A} \frac{\pi_i(a|s) - \frac{\epsilon}{|A|}}{1-\epsilon} \\
&\geq \frac{\epsilon}{|A|} \sum_{a \in A} Q^{\pi_i}(s,a) + (1-\epsilon) \sum_{a \in A} \frac{\pi_i(a|s) - \frac{\epsilon}{|A|}}{1-\epsilon} Q^{\pi_i}(s,a) \\
&= \sum_{a \in A} \pi_i(a|s) Q^{\pi_i}(s,a) = V^{\pi_i}(s)
\end{aligned}
$$

- Therefore $V^{\pi_{i+1}} \geq V^{\pi}$ (from the policy improvement theorem)

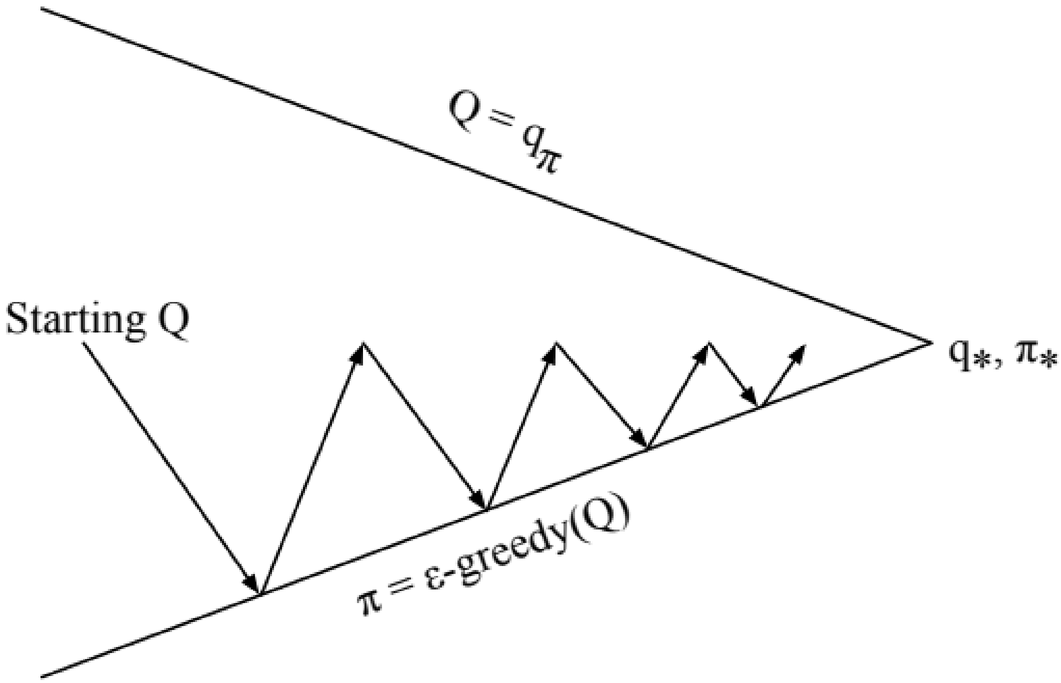
Monte-Carlo Policy Iteration



Policy evaluation Monte-Carlo policy evaluation, $Q = q_\pi$

Policy improvement ϵ-greedy policy improvement

Monte-Carlo Control



Every episode:

Policy evaluation Monte-Carlo policy evaluation, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Greedy in the Limit with Infinite Exploration (GLIE)

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times,

$$\lim_{k \to \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \to \infty} \pi_k(a|s) = \mathbf{1}(a = \operatorname*{argmax}_{a' \in \mathcal{A}} Q_k(s, a'))$$

- For example, ϵ-greedy is GLIE if ϵ reduces to zero at $\epsilon_k = \frac{1}{k}$

GLIE Monte-Carlo Control

- Sample kth episode using π: $\{S_1, A_1, R_2, ..., S_T\} \sim \pi$
- For each state S_t and action A_t in the episode,

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$$
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$
$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

Theorem

GLIE Monte-Carlo control converges to the optimal action-value function, $Q(s, a) \rightarrow q_*(s, a)$

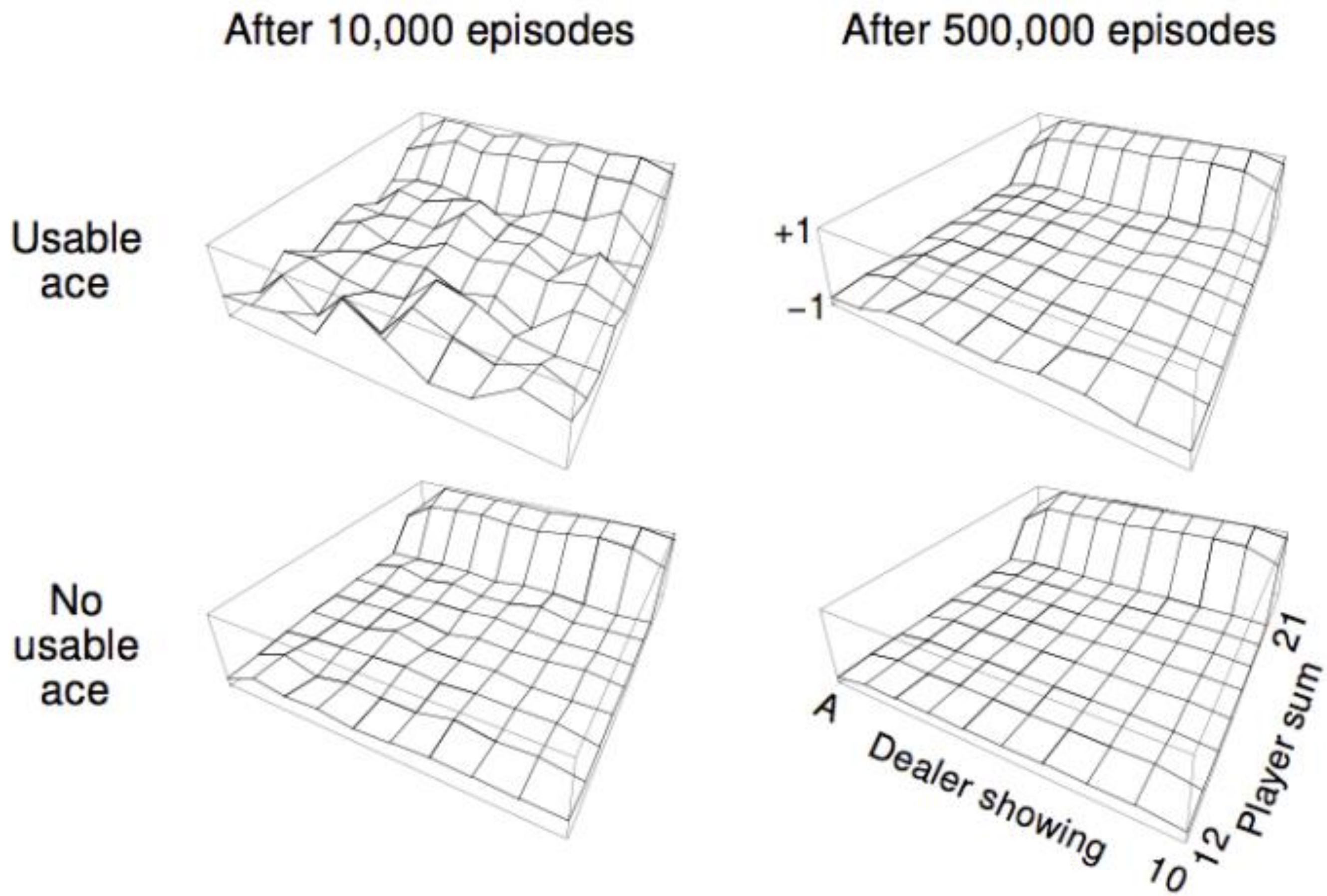
Monte Carlo Online Control / On Policy Improvement

Initialize $Q(s,a) = 0, N(s,a) = 0\ \forall(s,a)$, Set $\epsilon = 1,\ k = 1$
$\pi_k = \epsilon$-greedy(Q) // Create initial ϵ-greedy policy
loop
    Sample k-th episode $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \ldots, s_{k,T})$ given π_k
    $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \cdots \gamma^{T_i - 1} r_{k,T_i}$
    for $t = 1, \ldots, T$ **do**
        if First visit to (s,a) in episode k **then**
            $N(s,a) = N(s,a) + 1$
            $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)}(G_{k,t} - Q(s_t, a_t))$
        end if
    end for
    $k = k + 1,\ \epsilon = 1/k$
    $\pi_k = \epsilon$-greedy(Q) // Policy improvement
end loop

Blackjack Example

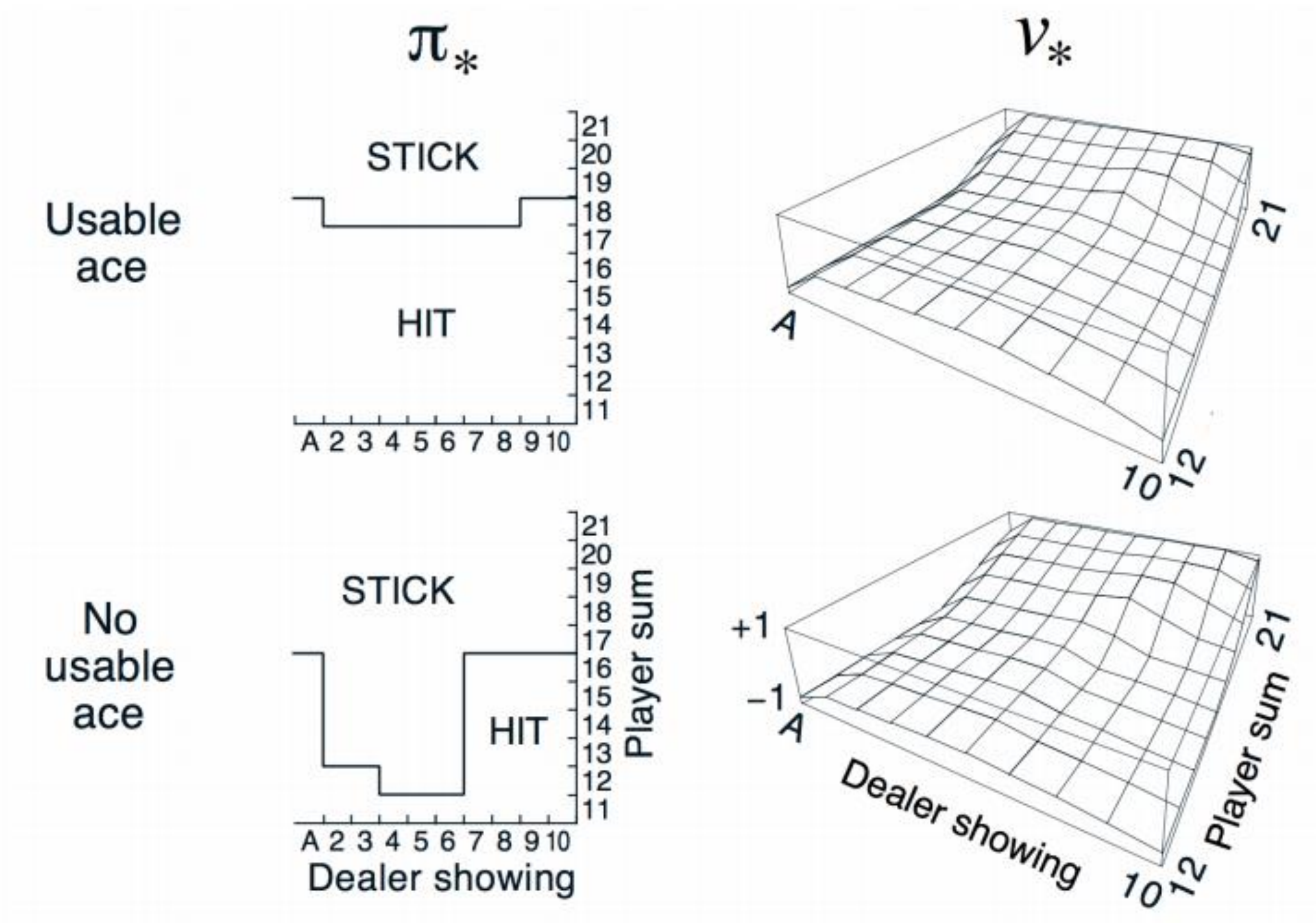
- States (200 of them):
  - Current sum (12-21)
  - Dealer's showing card (ace-10)
  - Do I have a "useable" ace? (yes-no)
- Action stick: Stop receiving cards (and terminate)
- Action twist: Take another card (no replacement)
- Reward for stick:
  - $+1$ if sum of cards $>$ sum of dealer cards
  - 0 if sum of cards $=$ sum of dealer cards
  - -1 if sum of cards $<$ sum of dealer cards
- Reward for twist:
  - -1 if sum of cards $>$ 21 (and terminate)
  - 0 otherwise
- Transitions: automatically twist if sum of cards $<$ 12

Blackjack Value Function after Monte-Carlo Learning

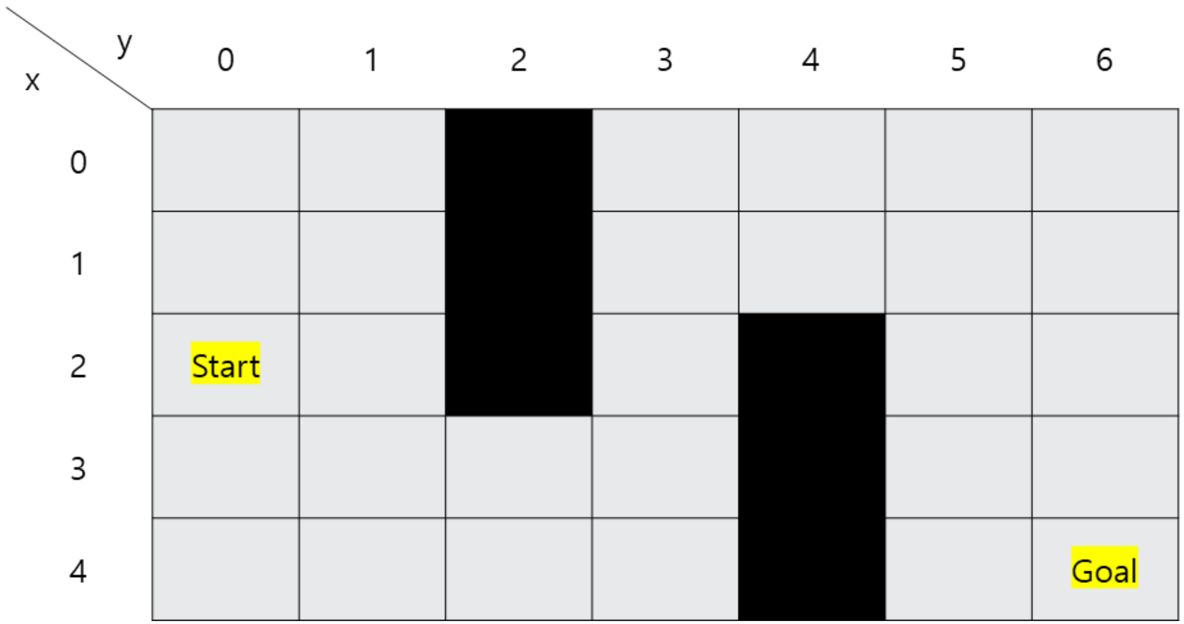


Policy: stick if sum of cards ≥ 20, otherwise twist

Monte-Carlo Control in Blackjack



More example-grid world



More example-grid world

x \ y	0	1	2	3	4	5	6
0	↓	↓		→	→	→	↓
1	↓	↓		↑	↑	→	↓
2	→	↓		↑		←	↓
3	→	↓	↓	↑		↓	↓
4	→	→	→	↑		→	

0: ←
1: ↑
2: →
3: ↓

[[3. 3. 0. 2. 2. 2. 3.]
[3. 3. 0. 1. 1. 2. 3.]
[2. 3. 0. 1. 0. 0. 3.]
[2. 3. 3. 1. 0. 3. 3.]
[2. 2. 2. 1. 0. 2. 0.]]