HSLU Lucerne University of Applied Sciences and Arts

Monte Carlo Methods

Reinforcement Learning
October 27, 2022

FH Zentralschweiz

BLACK JACK PAYS 3 TO 2
Dealer must draw to 16 and stand on all 17's
PAYS 2 TO 1
INSURANCE
4
3
2
1

Learning Objectives

- Understand how value functions are estimated with Monte-Carlo methods
- Understand how Monte-Carlo methods fit into the GPI pattern
- Understand why exploration is needed in MC methods
- Understand the difference between on-policy and off-policy methods
- Unterstand importance sampling
- Understand Monte Carlo prediction and control using off-policy methods

Introduction

- Dynamic Programming require a complete model of the environment and a sweep through all states for an update
- We now look at methods that do not need a model but learn from experience which can be actual or simulated
- While a simulated experience still needs a kind of model, it only needs it to generate sample transitions

Monte Carlo Methods

- Monte Carlo (MC) methods look at whole episodes and then average the complete returns
- The tasks must be episodic, so that all episodes eventually terminate (with the used policy!)
- Value estimation and policies are only changed **on the completion of an episode**
- (While the term is often used for other methods with some random components, we use it here for methods based on averaging complete returns)

Monte Carlo Methods

- The idea of general policy iterations (GPI) from dynamic programming can be adapted to MC methods
- We first consider the prediction problem of estimating the value function(s) for a known policy
- We can then look at the control problem, i.e., finding the best policy

Monte Carlo Prediction

- We want to estimate $v_\pi(s)$, the value of a state s under policy π
- Given is a set of episodes (that pass through s)

- An episode might pass multiple times through s, we will only consider it once, this is known as *first-visit MC*

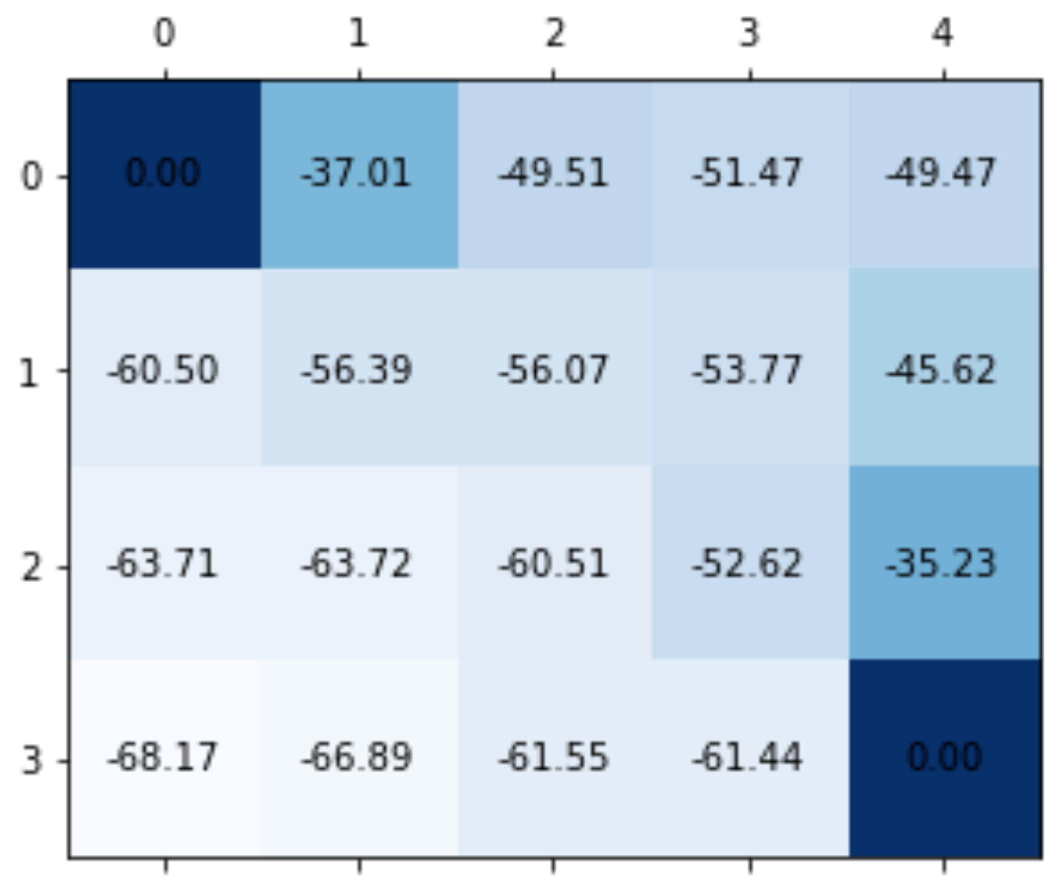
- We average the returns of all episodes

Monte Carlo Prediction (first visit)

Monte Carlo Prediction for estimating v_π

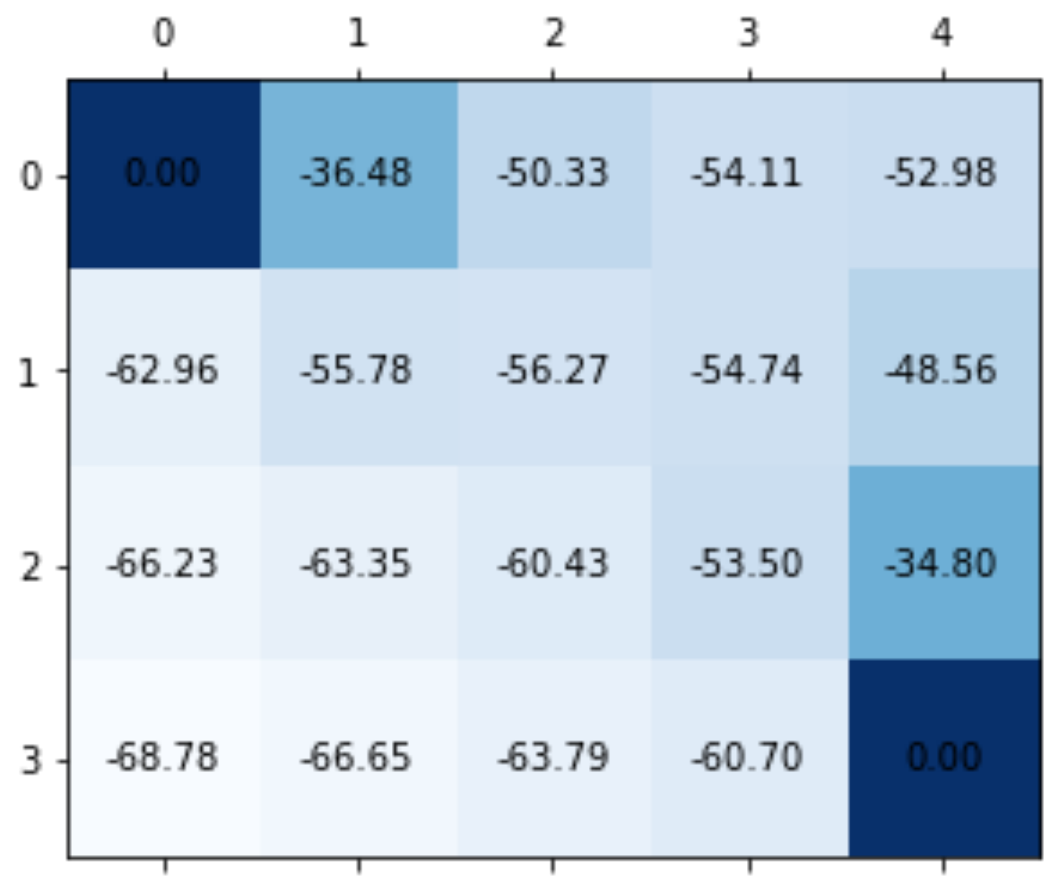
```
Input: a policy $\pi$
Initialize:
    $V(s) \in \mathbb{R}$ (arbitrarily)
    Returns$(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$
Loop forever:
    Generate episode following $\pi$: $S_0, A_0, R_0, S_1, ..., R_T$
    $G \leftarrow 0$
    Loop for each step of the episode, $t = T-1, T-2, ..., 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless $S_t$ appears in $S_{t-1}, ...S_0$:
            Append $G$ to $Returns(S_t)$
            $V(S_t) \leftarrow$ average$(Returns(S_t))$
```

Monte Carlo Prediction



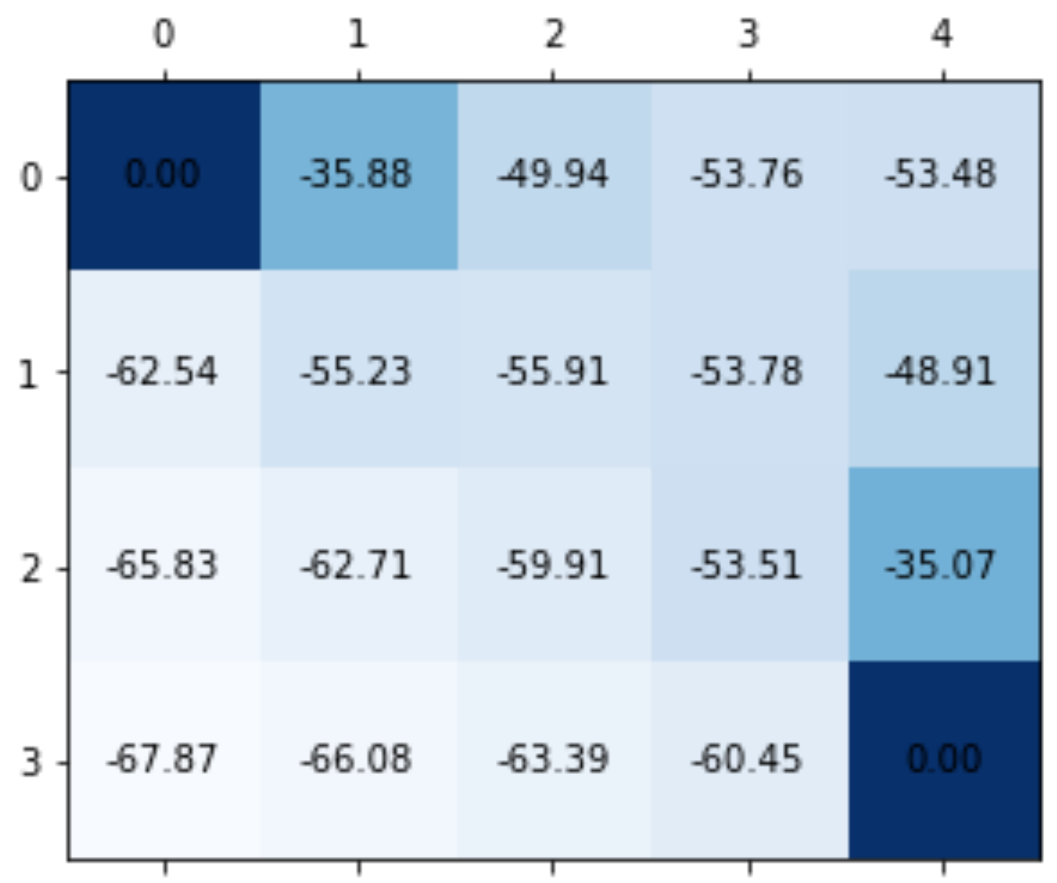
	0	1	2	3	4
0	0.00	-37.01	-49.51	-51.47	-49.47
1	-60.50	-56.39	-56.07	-53.77	-45.62
2	-63.71	-63.72	-60.51	-52.62	-35.23
3	-68.17	-66.89	-61.55	-61.44	0.00

1'000 episodes



	0	1	2	3	4
0	0.00	-36.48	-50.33	-54.11	-52.98
1	-62.96	-55.78	-56.27	-54.74	-48.56
2	-66.23	-63.35	-60.43	-53.50	-34.80
3	-68.78	-66.65	-63.79	-60.70	0.00

10'000 episodes



	0	1	2	3	4
0	0.00	-35.88	-49.94	-53.76	-53.48
1	-62.54	-55.23	-55.91	-53.78	-48.91
2	-65.83	-62.71	-59.91	-53.51	-35.07
3	-67.87	-66.08	-63.39	-60.45	0.00

100'000 episodes

Example on gridworld (with interior walls) with random policy

Black Jack Example

Ace can be 11 or 1, if the ace can be used as 11 it is called useable (and counted as 11)

State of the system (from the point of the player):

- current sum of the player
- card of the dealer
- ace usable or not

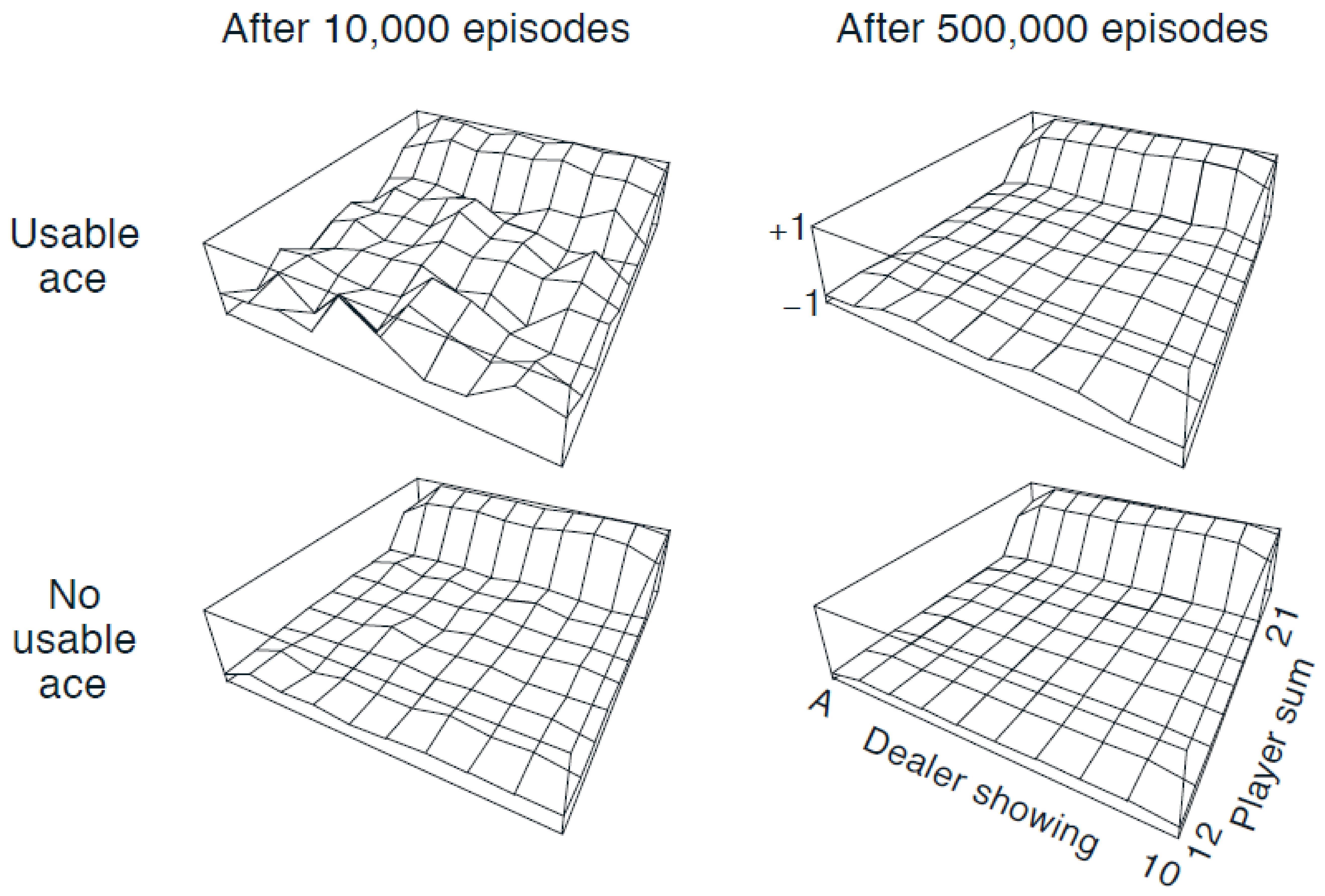
Black Jack Example

- Player vs. Dealer
- 2 cards dealt each at the beginning, one card of the dealer is shown

- Actions: player can request another card (hit) or stop (stick)
- Dealer has fixed strategy: stick with 17 or more, hit otherwise

- Rewards: -1 (loss), 0 (draw), 1 (win) at end
- Episodic tasks

Blackjack



Value function for policy that sticks with 20 or 21 points

Action-Value Function Estimation

- With a model, we can determine the best action by
  - using the state-value function
  - looking one step ahead (for all possible actions)
  - choose the best action (reward and next state value)

- Without a model, we cannot look ahead, so we want to estimate the action-value function instead of the state-value function:

$$q_\pi(s, a)$$

- I.e., the expected return from starting in s with action a and following π

Prediction of Action Values

The prediction algorithm can be adapted for action values

However:
- Some state-action pairs might never be visited

Solutions:
- Exploring Starts: Every state-action pair has a nonzero probability to be selected as start, or
- Maintain exploration by using a policy that has a nonzero probability for every action (this is called an epsilon-soft policy)

Monte Carlo Control

- Generalized policy iteration (GPI) can be used with Monte Carlo Prediction

- Policy Estimation is done as described above
- Policy Improvement is done by choosing a greedy policy with respect to the action-value function

$$\begin{tikzcd}
\pi \arrow[r, bend left=60, "\text{evaluation}", "Q \rightarrow q_\pi"'] & Q \arrow[l, bend left=60, "\text{improvement}", "\pi \rightarrow \text{greedy}(Q)"']
\end{tikzcd}$$

$$\pi_0 \xrightarrow{\text{E}} q_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_{\pi_*}$$

On-policy first-visit MC control, estimates $\pi \approx \pi_*$

```
Input: (small) $\epsilon > 0$
Initialize:
    $\pi \leftarrow$ an arbitrary $\epsilon$-soft policy
    $Q(s,a) \in \mathbb{R}$ (arbitrarily, for example $= 0$)
    $Returns(s,a) \leftarrow$ empty list
Loop forever: (for each episode)
    Generate an episode following $\pi$: $S_0, A_0, R_0, S_1, ..., R_T$
    $G \leftarrow 0$
    Loop for each step of the episode, $t = T-1, T-2, ..., 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless $(S_t, A_t)$ appears in $(S_{t-1}, A_{t-1}), ..., (S_0, A_0)$:
            Append $G$ to $Returns(S_t, A_t)$
            $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
            $A^* \leftarrow \text{argmax}_a Q(S_t, a)$, ties broken arbitrarily
            For all $a \in \mathcal{A}(S_t)$:
                $\pi(a|S_t) \leftarrow \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \epsilon/|\mathcal{A}(S_t)| & \text{otherwise} \end{cases}$
```

On policy vs. off policy algorithm

- On-policy algorithms learn a policy while following this policy in the algorithm
- Off-policy algorithms learn a policy different from the one used to generate the data

- On-policy algorithms require a *soft* policy, which means that

$$\pi(a|s) > 0, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}$$

Importance Sampling

- epsilon-soft policies are a compromise:
- we use an almost optimal policy to learn action values while still exploring

- We would like to use 2 policies:
  - A target policy π that we want to learn
  - A behavior policy b that we use for exploring

- constraint: every action under π must also be taken under b with non-zero probability

Importance Sampling

The probability of a state trajectory under a policy is:

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, ..., S_T | S_t, A_{t:T-1} \sim \pi\} \\
&= \pi(A_t|S_t) p(S_{t+1}|S_t, A_t) \pi(A_{t+1}|S_{t+1} \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&= \prod_{k=1}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)
\end{aligned}$$

So the relative probability of two policies is

$$\begin{aligned}
\rho_{t:T} &\doteq \frac{\prod_{k=1}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=1}^{T-1} b(A_k|S_k) p(S_{k+1}|S_k, A_k)} \\
&= \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}
\end{aligned}$$

Corection of averaged returns

Ordinary importance sampling:

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$

Weighted importance sampling
(derivation in the book)

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1}}$$

where $\mathcal{T}(s)$ is the set of all timesteps
that pass-through s

Off-policy MC prediction, estimates $Q \approx q_\pi$

```
Input: a policy $\pi$
Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily, for example $= 0$)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any soft policy compatible with $\pi$
    Generate an episode following b: $S_0, A_0, R_0, S_1, ..., R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of the episode, $t = T-1, T-2, ..., 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
```

Off-policy MC control, estimates $\pi \approx \pi_*$

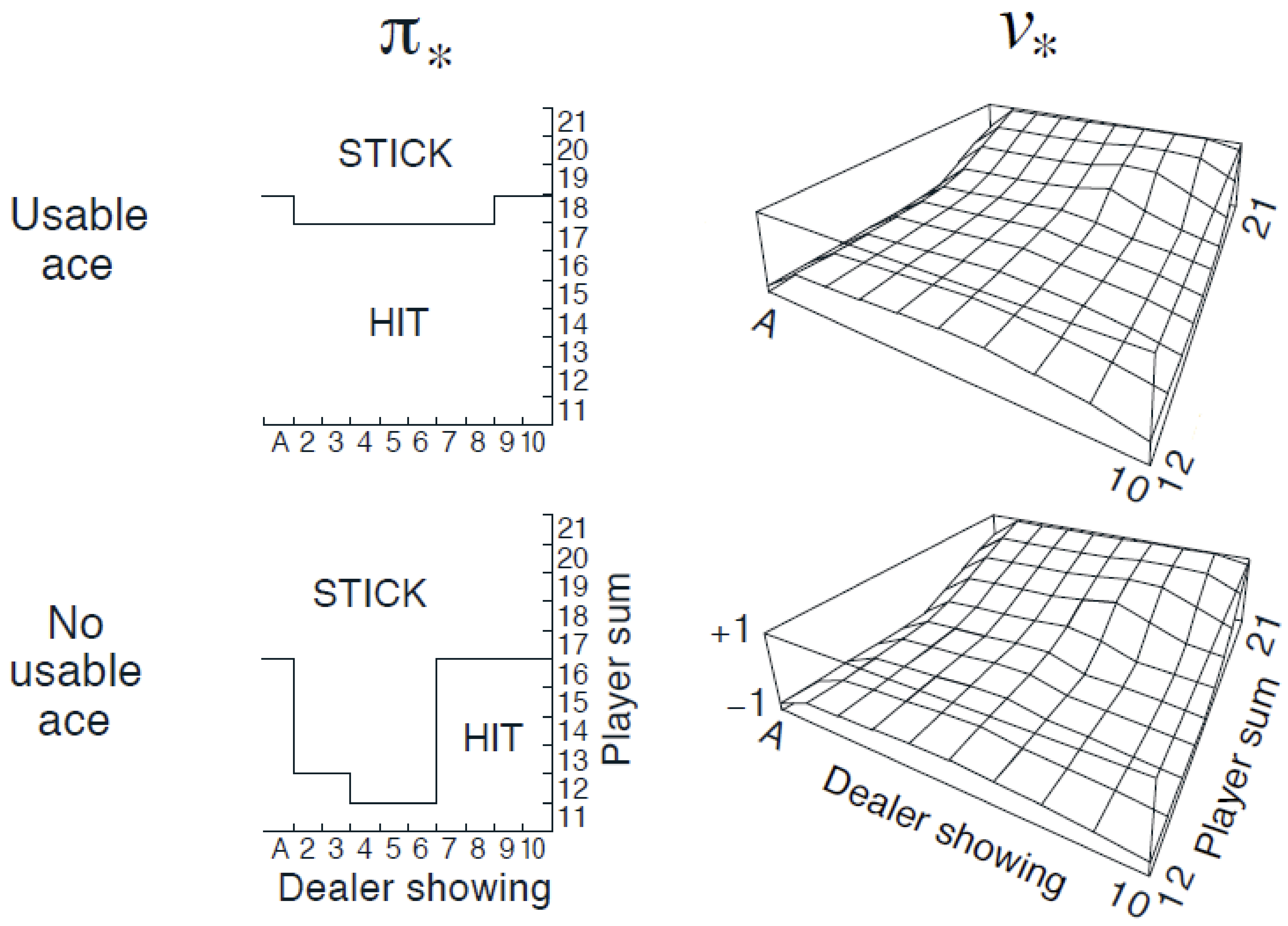
```
Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily, for example $= 0$)
    $C(s,a) \leftarrow 0$
    $\pi(s) \leftarrow \text{argmax}_a Q(s,a)$    (with ties broken consistently)

Loop forever (for each episode):
    $b \leftarrow$ any soft policy
    Generate an episode following b: $S_0, A_0, R_0, S_1, ..., R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of the episode, $t = T-1, T-2, ..., 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)}[G - Q(S_t, A_t]$
            $\pi(a|S_t) \leftarrow \text{argmax}_a Q(S_t, A_t)$    (with ties broken consistently)
            If $A_t \neq \pi(S_t)$ then exit inner Loop (next episode)
            $W \leftarrow W \frac{1}{b(A_t|S_t)}$
```

Black Jack Example

Everything is know about the black jack environment...

Could the problem be solved using Dynamic Programming?



Optimal policy and value function of blackjack

Summary (I)

- In MC, experience comes from *sample episodes.*

- Advantages over DP:
  - Interaction is with the environment, so no model of the environment's dynamic is needed
  - If models are available, they can be used with simulation in the form of sample models
  - MC can focus on a small subset of states

- Control methods follow the scheme of *generalized policy iteration* (GPI)

Summary (II)

- MC control methods need to maintain sufficient exploration
- No bootstrapping (learning from other value estimates) is possible
- In on-policy methods, the agent tries to optimize the policy it follows
- In off-policy methods, a deterministic optimal policy is learned, while following a possibly unrelated policy

HSLU Lucerne University of Applied Sciences and Arts

{{DN:Hierarchy|Organisation Bezeichnung Spez.EN|ID:32|Hierarchy:1}}
Research
Prof. Dr. Thomas Koller
Lecturer

Phone direct +41 41 757 68 32
thomas.koller@hslu.ch

FH Zentralschweiz