Introduction to Reinforcement Learning

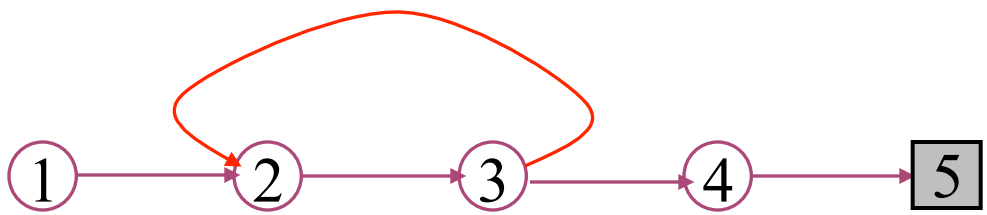
Part 4: Monte Carlo Learning to solve Blackjack

Monte Carlo Methods for Solving MDPs

- Monte Carlo methods are *learning* methods
  - Experience → values, policy
- Monte Carlo methods can be used in two ways:
  - *On-line:* No model necessary and still attains optimality
  - *Simulated:* Planning without a full probability model
- Monte Carlo methods learn from *complete* sample returns
  - Only defined for episodic

Monte Carlo Policy Evaluation

- *Goal:* learn v_π
- *Given:* some number of episodes under π which contain s
- *Idea:* Average returns observed after visits to s



- *Every-Visit MC:* average returns for *every* time s is visited in an episode
- *First-visit MC:* average returns only for *first* time s is visited in an episode
- Both converge asymptotically to v_π

First-visit Monte Carlo policy evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

$G \leftarrow$ return following the first occurrence of s

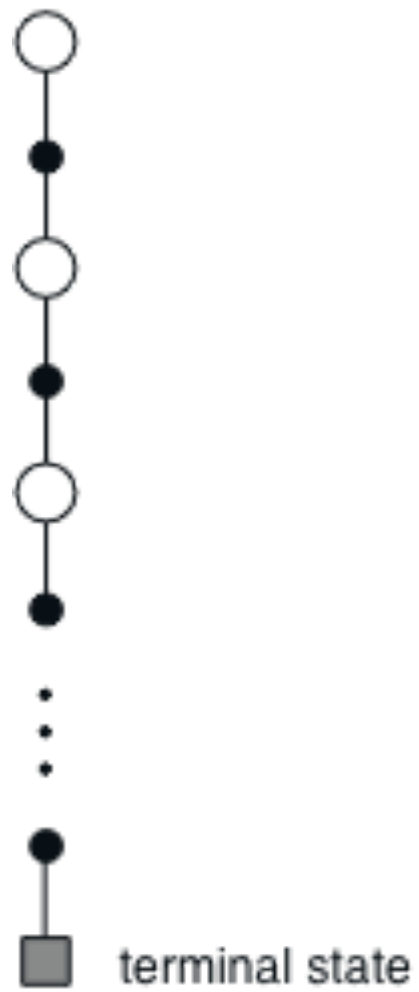
Append G to $Returns(s)$

$V(s) \leftarrow \text{average}(Returns(s))$

Backup diagram for Monte Carlo

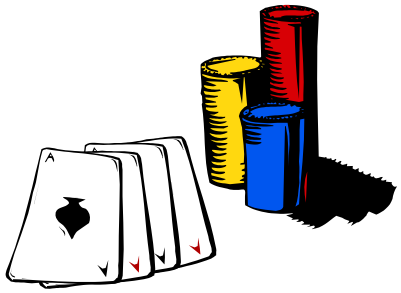
- Entire episode included
- Only one choice at each state

- Time required to estimate one state does not depend on the total number of states



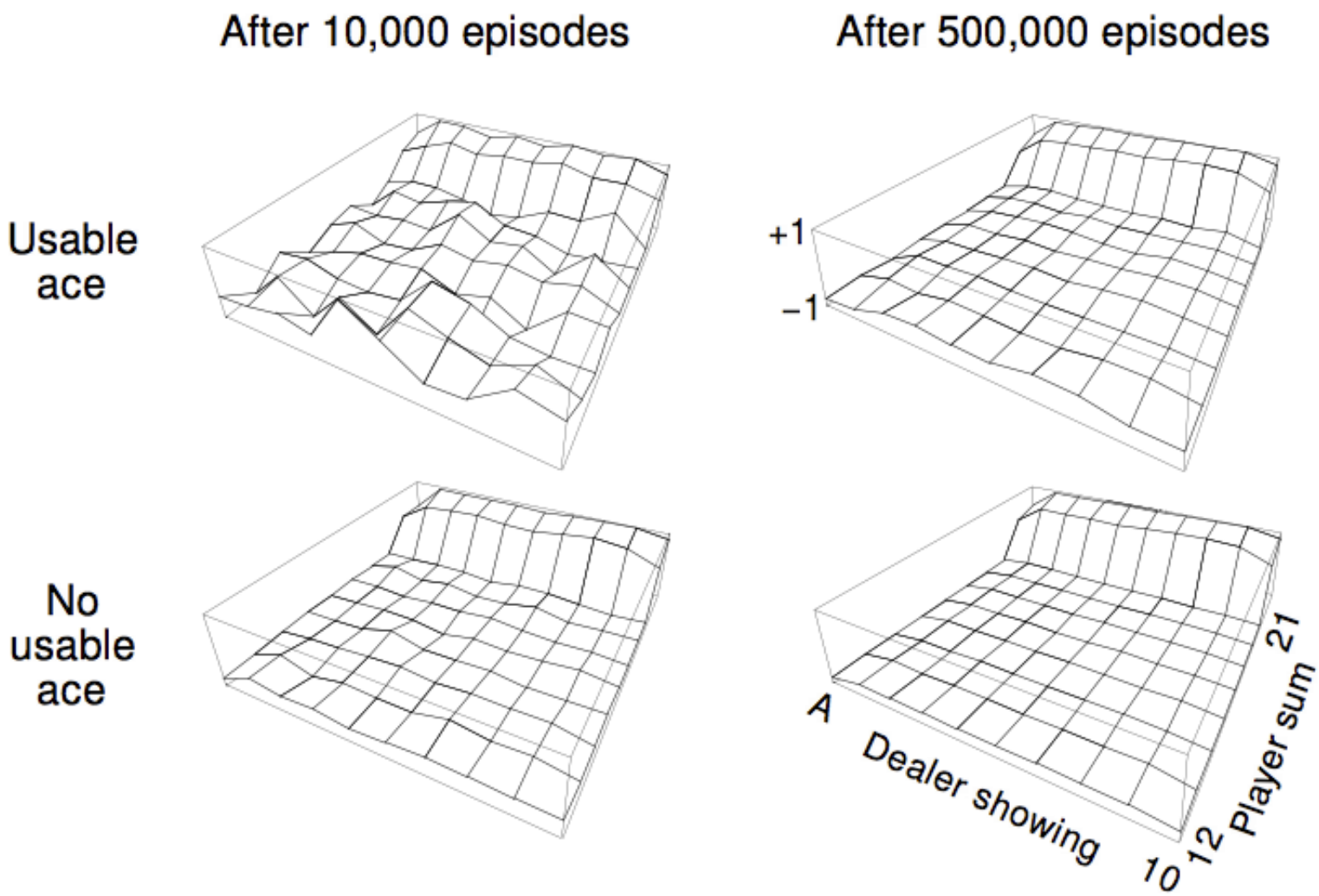
Blackjack example

- *Object:* Have your card sum be greater than the dealers without exceeding 21.
- *States* (200 of them):
  - current sum (12-21)
  - dealer's showing card (ace-10)
  - do I have a useable ace?



- *Reward:* +1 for winning, 0 for a draw, -1 for losing
- *Actions:* stick (stop receiving cards), hit (receive another card)
- *Policy:* Stick if my sum is 20 or 21, else hit

Blackjack value functions



Monte Carlo Exploring Starts

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list

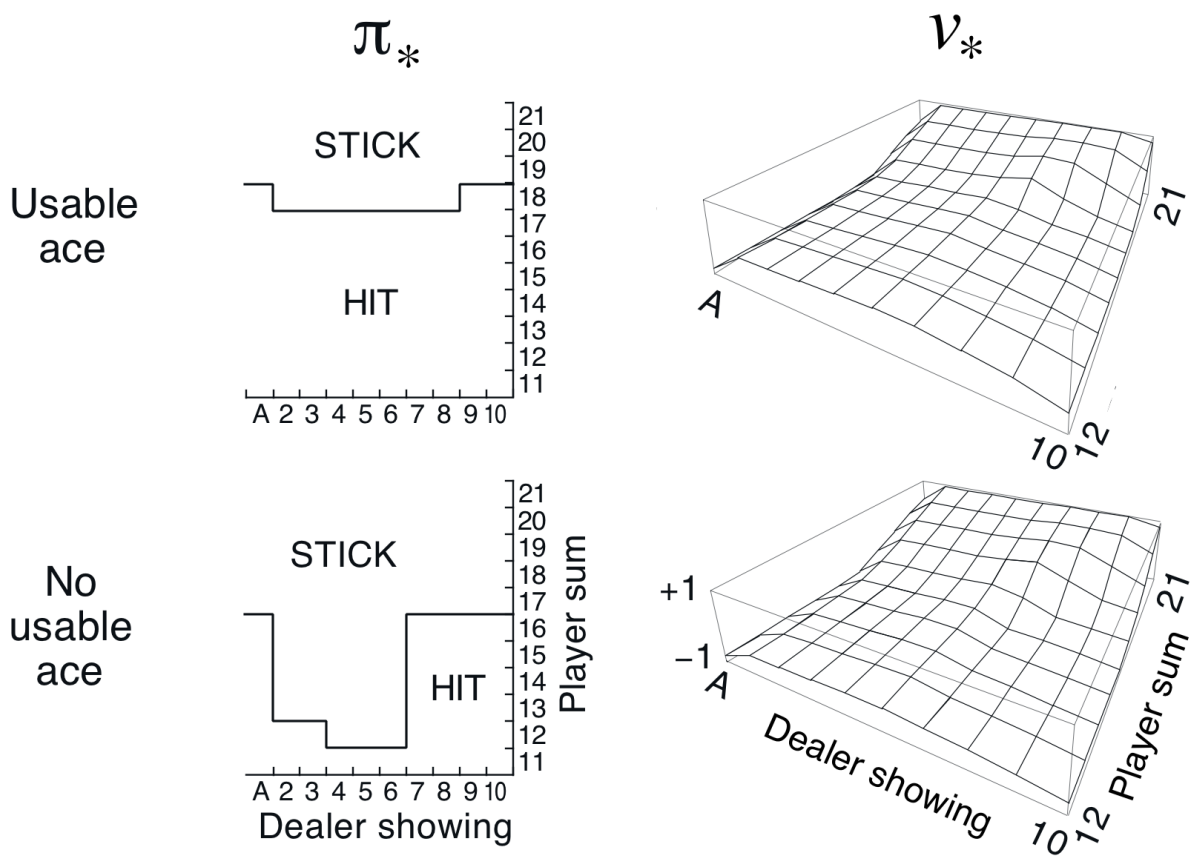
Fixed point is optimal policy π^*

Now proven (almost)

Repeat forever:
(a) Generate an episode using exploring starts and π
(b) For each pair s, a appearing in the episode:
$G \leftarrow$ return following the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
(c) For each s in the episode:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Blackjack example continued

- Exploring starts
- Initial policy as described before



Monte Carlo Exploring Starts

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list

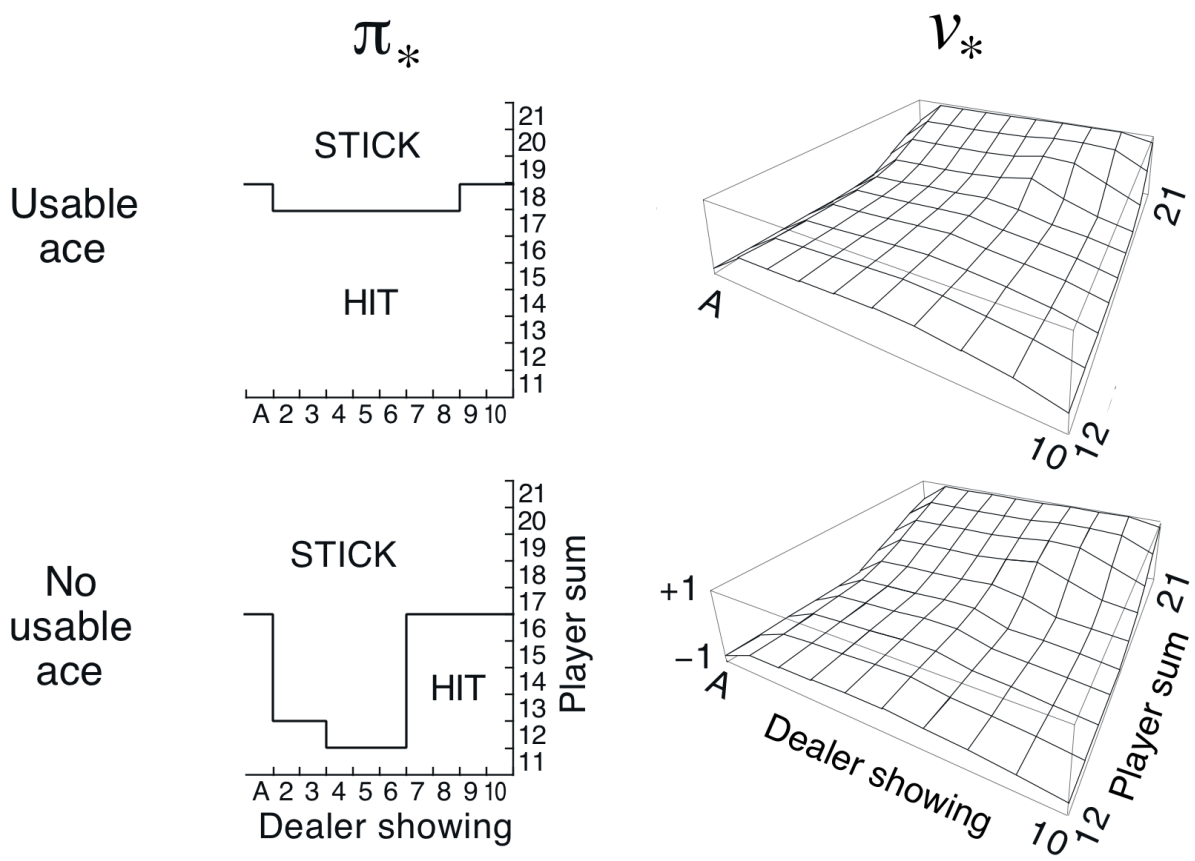
Fixed point is optimal policy π^*

Now proven (almost)

Repeat forever:
(a) Generate an episode using exploring starts and π
(b) For each pair s, a appearing in the episode:
$G \leftarrow$ return following the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
(c) For each s in the episode:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Blackjack example continued

- Exploring starts
- Initial policy as described before



On-policy Monte Carlo Control

- *On-policy:* learn about policy currently executing
- How do we get rid of exploring starts?
  - Need *soft* policies: $\pi(a|s) > 0$ for all s and a
  - e.g. ε-soft policy:

$$\frac{\varepsilon}{|A(s)|} \qquad\qquad 1-\varepsilon+\frac{\varepsilon}{|A(s)|}$$

non-max greedy

- Similar to GPI: move policy *towards* greedy policy (i.e. ε-soft)
- Converges to best ε-soft policy

Summary for Monte Carlo Methods

- Monte Carlo methods learn directly from experience
  - *On-line:* No model necessary and still attains optimality
  - *Simulated:* No need for a *full* model
- Monte Carlo methods learn from *complete* sample returns
  - Only defined for episodic tasks
- The *Monte Carlo exploring starts* algorithm avoids the vexing issue of maintaining exploration