Monte Carlo Approaches to Reinforcement Learning

Chris Amato
Northeastern University

with some slides from Rob Platt and UAlberta

Announcements

- DP assignment due today
- MC (programming) assignment out today and due 10/11
- TD quiz on Monday (so read TD chapter)
- First midterm 10/15

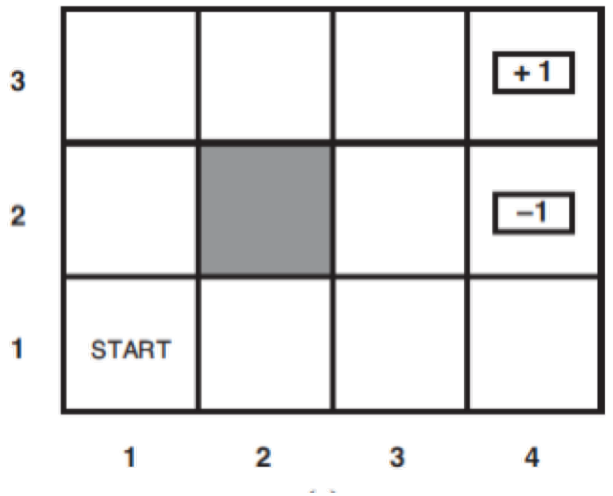
Monte Carlo Methods

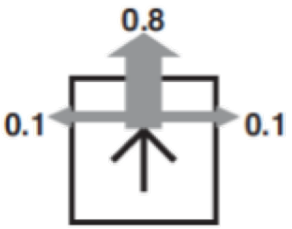
- Our first *learning* method for MDPs
- Often hard to actually know transition and reward values or the problem is too big to repeatedly iterate over states and actions
- Monte Carlo methods estimate the return using samples starting from a given state
- Monte Carlo methods learn from *complete* sample returns
  - Only defined for episodic tasks (in this book)
- Like an associative version of a bandit method

- Note: not talking about Monte Carlo tree search today (this is a planning method and we'll talk about it later)

MDP dynamic programming (planning)

We know the reward function

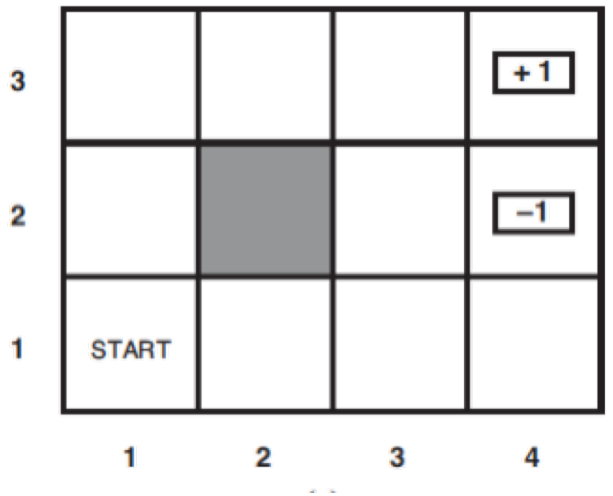


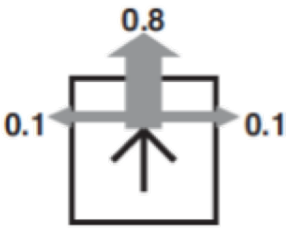


We know the probabilities of moving in each direction when an action is executed

Reinforcement Learning

We don't know the reward function





We don't know the probabilities of moving in each direction when an action is executed

RL still assumes that we have an MDP

Model-free RL example

Initial

A Learning Trial

After Learning [1K Trials]

[Kohl and Stone, ICRA 2004]

Not quite the same method we'll talk about today, but related

Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Training

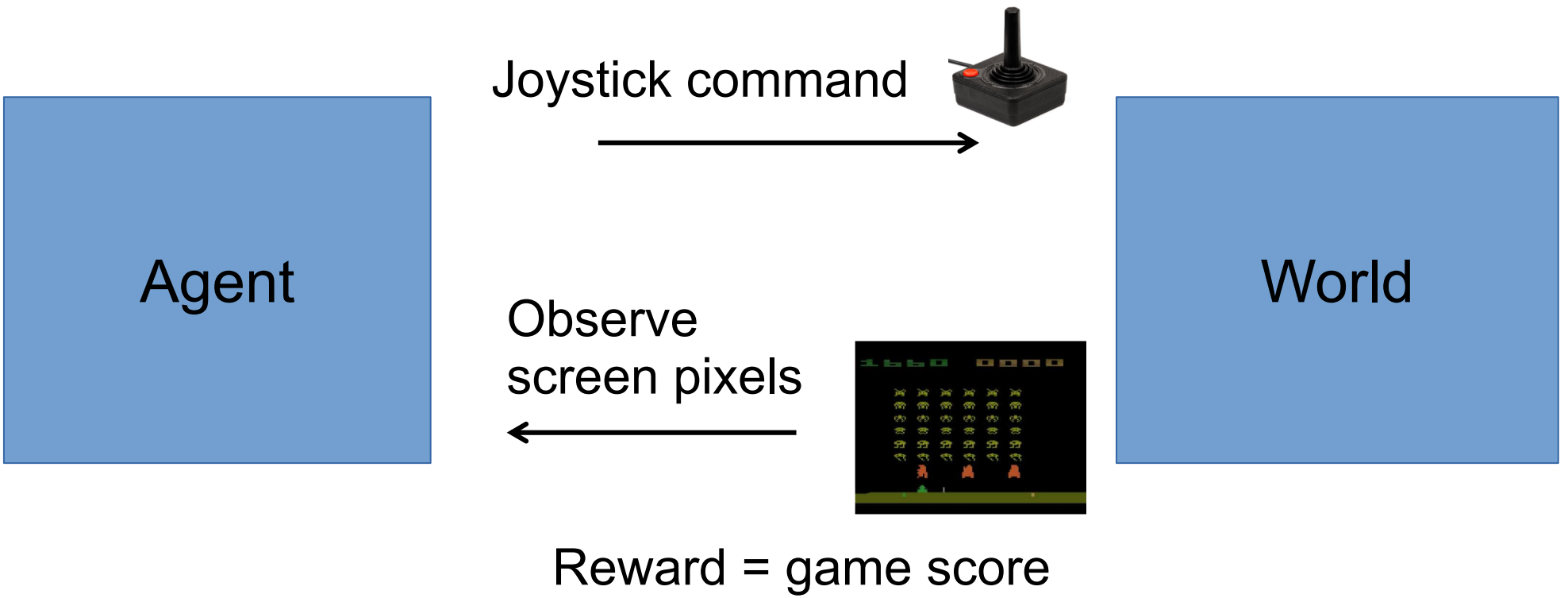
[Kohl and Stone, ICRA 2004]

Example: Learning to Walk

Finished

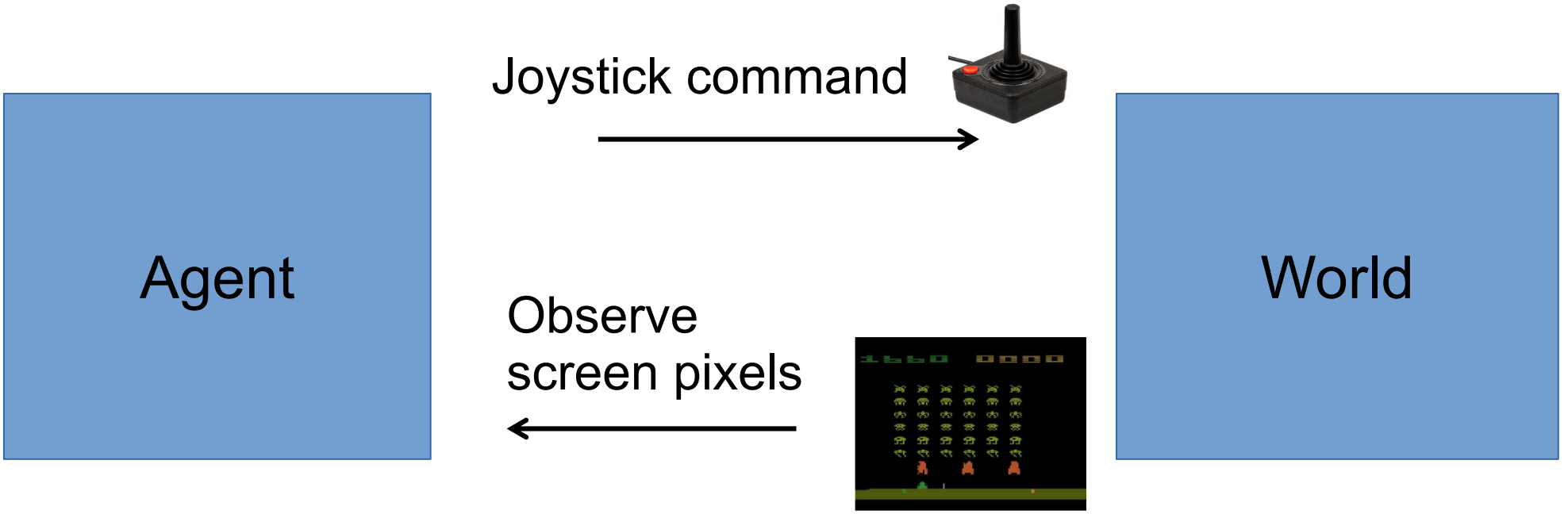
[Kohl and Stone, ICRA 2004]

Model-Free Reinforcement Learning



Goal: learn a value (or policy) function through trial-and-error experience
(not learning a model of the MDP, hence it is model-free)

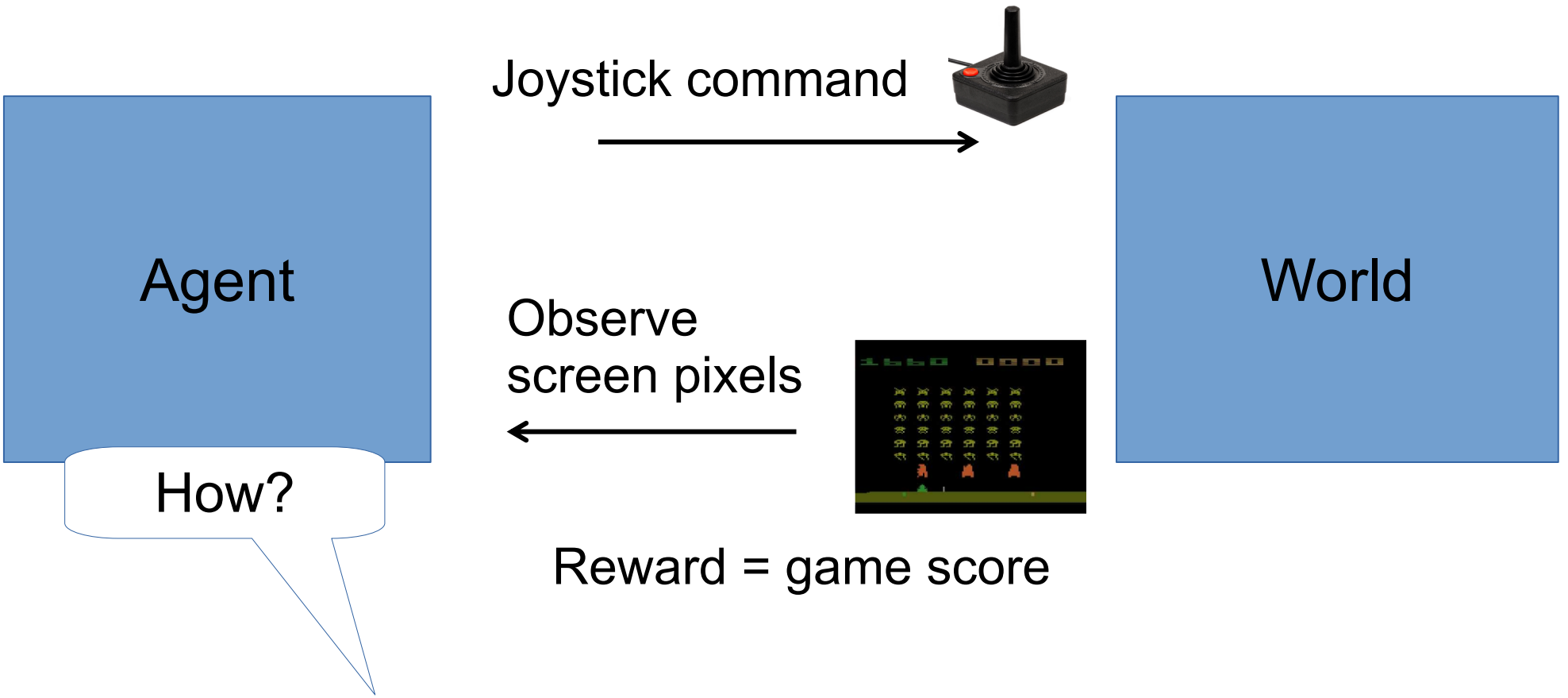
Model-Free Reinforcement Learning



Goal: learn a value function through trial-and-error experience

Recall: $V^{\pi}(s_t = s) \equiv$ Value of state s when acting according to policy π

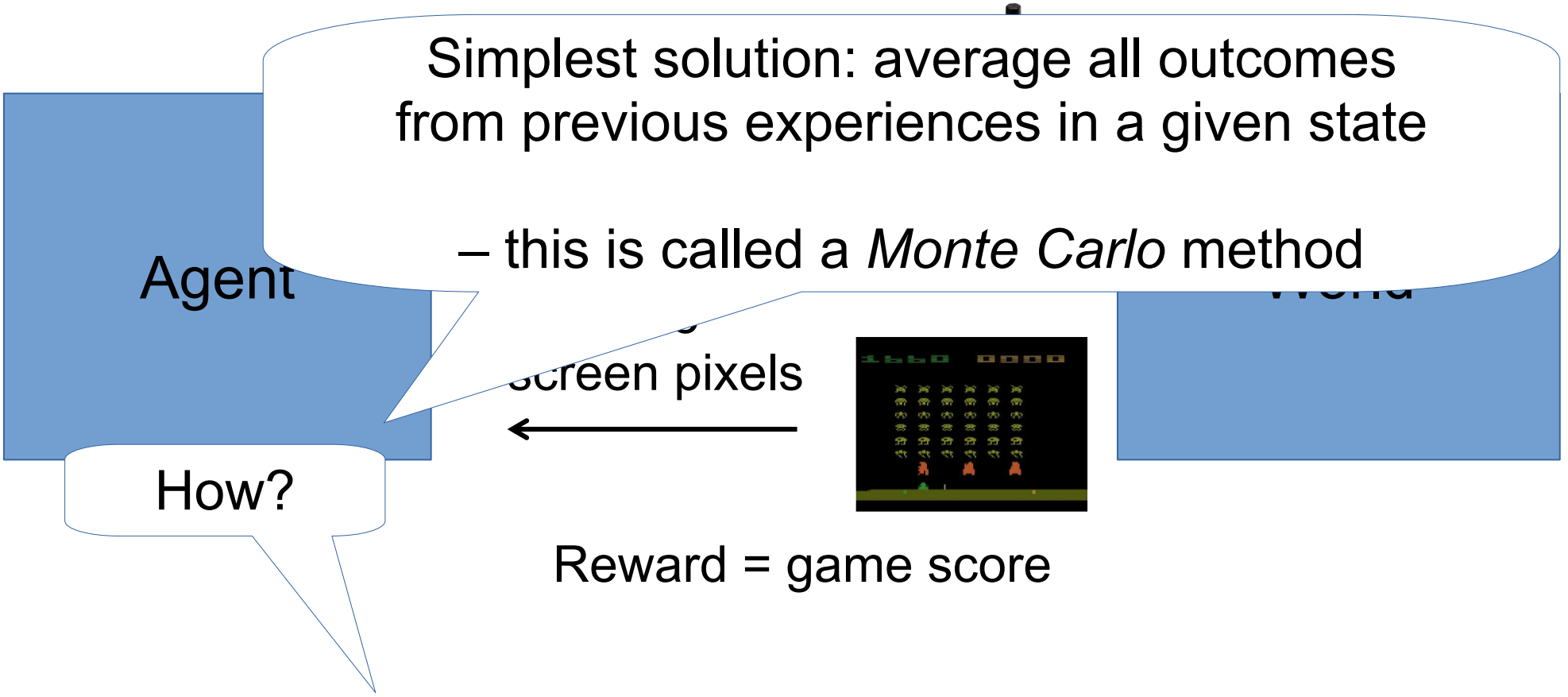
Model-Free Reinforcement Learning



Goal: learn a value function through trial-and-error experience

Recall: $V^{\pi}(s_t = s) \equiv$ Value of state s when acting according to policy π

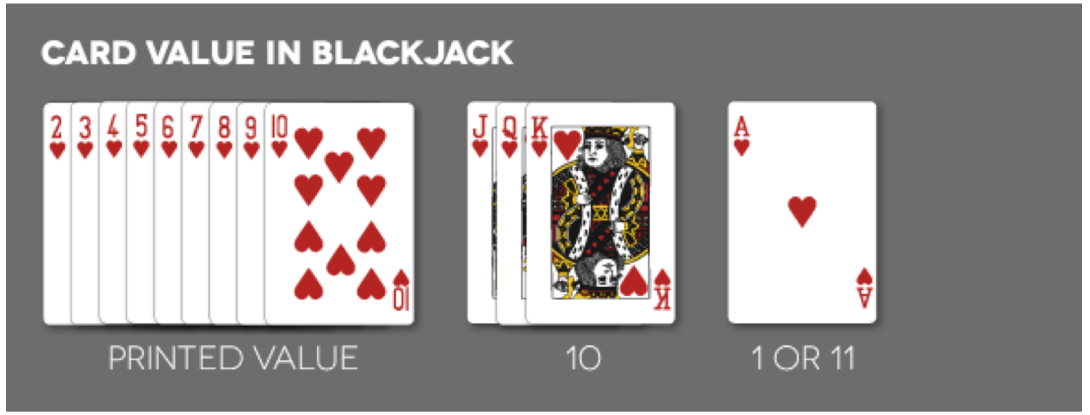
Model-Free Reinforcement Learning



Goal: learn a value function through trial-and-error experience

Recall: $V^{\pi}(s_t = s) \equiv$ Value of state s when acting according to policy π

Running Example: Blackjack



State: sum of cards in your hand + dealer's showing card + does agent have usable ace?

Actions: hit, stand, double, split

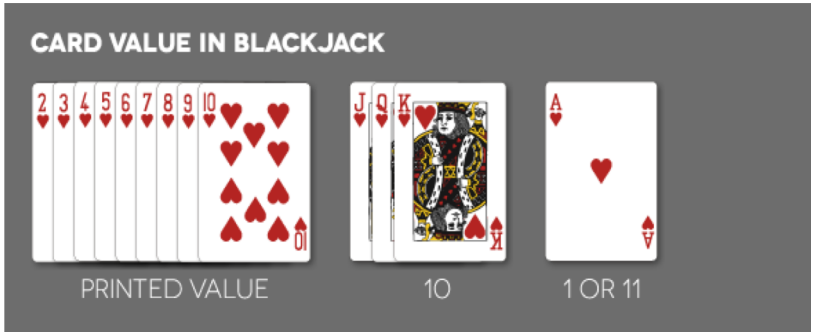
Objective: Have your card sum be greater than the dealer's without exceeding 21

Reward: +1 for winning, 0 for a draw, -1 for losing

Discounting: $\gamma = 1$

Dealer policy: draw until sum at least 17

Running Example: Blackjack



Dealer's Up Card

← Your Hand	2	3	4	5	6	7	8	9	10	A
17+	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	H
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
5 - 8	H	H	H	H	H	H	H	H	H	H
A 8 - 10	S	S	S	S	S	S	S	S	S	S
A, 7	S	D	D	D	D	S	S	H	H	H
A, 6	H	D	D	D	D	H	H	H	H	H
A, 5	H	H	D	D	D	H	H	H	H	H
A, 4	H	H	D	D	D	H	H	H	H	H
A, 3	H	H	H	D	D	H	H	H	H	H
A, 2	H	H	H	D	D	H	H	H	H	H
A,A 8,8	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
10, 10	S	S	S	S	S	S	S	S	S	S
9, 9	SP	SP	SP	SP	SP	S	SP	SP	S	S
7, 7	SP	SP	SP	SP	SP	SP	H	H	H	H
6, 6	SP	SP	SP	SP	SP	H	H	H	H	H
5, 5	D	D	D	D	D	D	D	D	H	H
4, 4	H	H	H	SP	SP	H	H	H	H	H
3, 3	SP	SP	SP	SP	SP	SP	H	H	H	H
2, 2	SP	SP	SP	SP	SP	SP	H	H	H	H
	2	3	4	5	6	7	8	9	10	A

If doubling down after splitting is not allowed, then just hit the following:
2,2 and 3,3 vs. 2 and 3 4,4 vs. 5 and 6 6,6 vs. 2

HIT | STAND | DOUBLE DOWN | SPLIT

Blackjack "Basic Strategy" is a set of rules for play so as to maximize return

- well known in the gambling community
- how might an RL agent *learn* the Basic Strategy?

Monte Carlo Policy Evaluation

Given a policy, π , estimate the value function, $V(s)$, for all states, $s \in \mathcal{S}$

Monte Carlo Policy Evaluation (first visit):

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:

Generate an episode using π

For each state s appearing in the episode:

$G \leftarrow$ the return that follows the first occurrence of s

Append G to $Returns(s)$

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

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:

State	Action	Next State	Reward

Monte Carlo Policy Evaluation: Example

Dealer card:

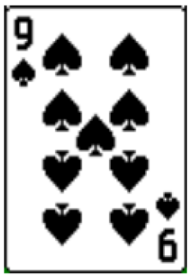
Agent's hand:

State	Action	Next State	Reward
19, 10, no			

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:



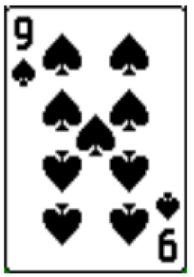
Agent sum, dealer's card, ace?

State	Action	Next State	Reward
19, 10, no			

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:



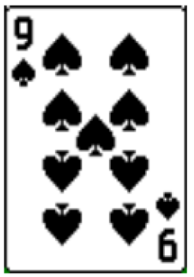
Agent sum, dealer's card, ace?

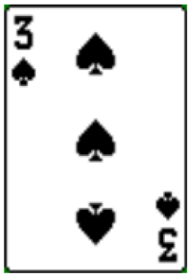
State	Action	Next State	Reward
19, 10, no	HIT		

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:





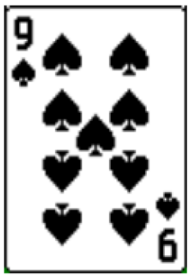
Agent sum, dealer's card, ace?

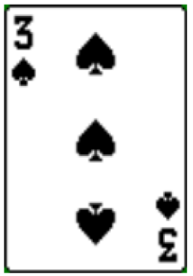
State	Action	Next State	Reward
19, 10, no	HIT	22, 10, no	-1

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:





Agent sum, dealer's card, ace?

Bust!
(reward = -1)

State	Action	Next State	Reward
19, 10, no	HIT	22, 10, no	-1

Monte Carlo Policy Evaluation: Example

State	Action	Next State	Reward
19, 10, no	HIT	22, 10, no	-1

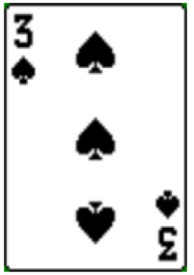
Upon episode termination, make the following value function updates:

$$V((19, 10, no)) \leftarrow -1$$
$$V((22, 10, no)) \leftarrow -1$$

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:

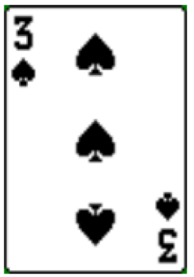


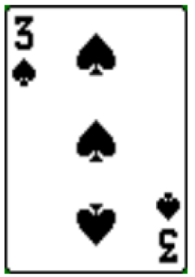
State	Action	Next State	Reward
13, 10, no			

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:



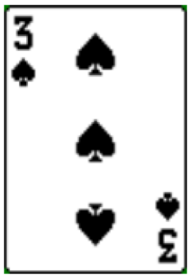


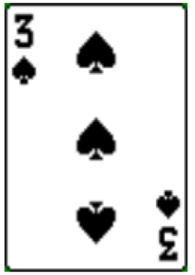
State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:



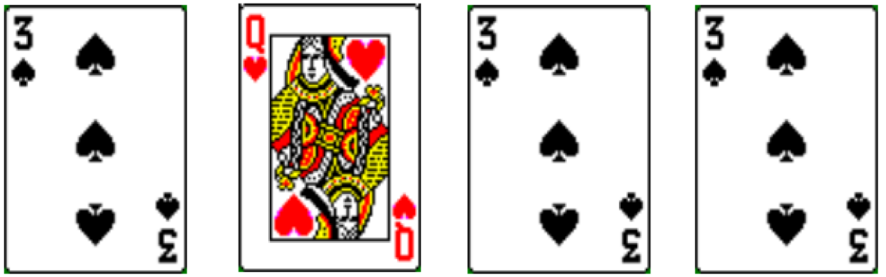


State	Action	Next State	Reward
13, 10, no 16, 10, no	HIT	16, 10, no	0

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:

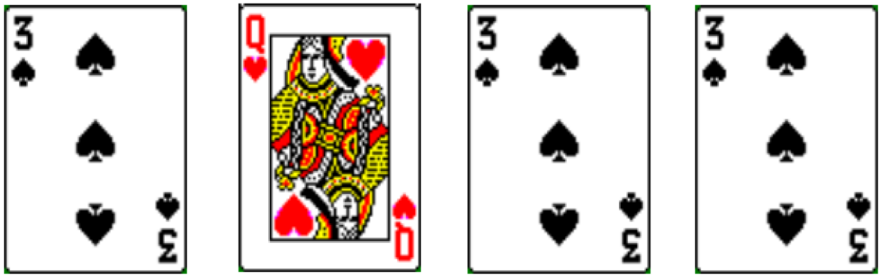


State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0
16, 10, no	HIT	19, 10, no	0

Monte Carlo Policy Evaluation: Another Example

Dealer card:

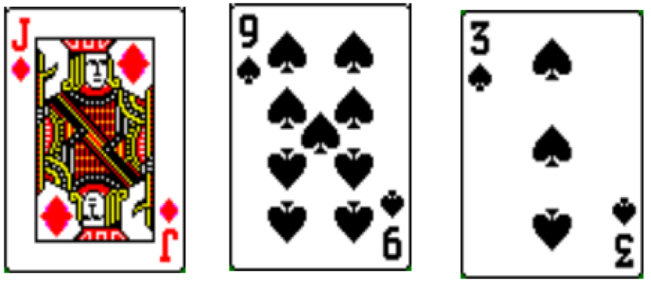
Agent's hand:



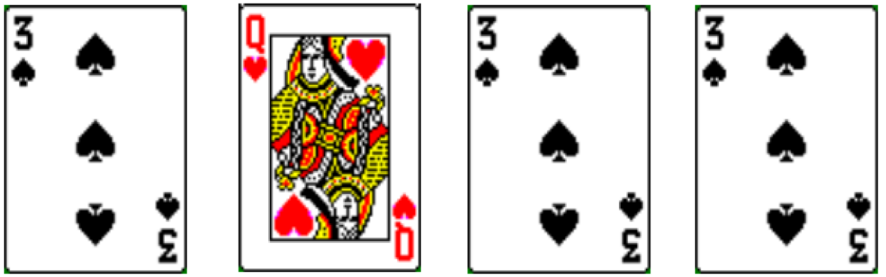
State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0
16, 10, no	HIT	19, 10, no	0
19, 10, no			

Monte Carlo Policy Evaluation: Another Example

Dealer card:



Agent's hand:



State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0
16, 10, no	HIT	19, 10, no	0
19, 10, no	STAND	19, 22, no	1

Monte Carlo Policy Evaluation: Another Example

State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0
16, 10, no	HIT	19, 10, no	0
19, 10, no	STAND	19, 22, no	1

Upon episode termination, make the following value function updates:

$$V((13, 10, no)) \leftarrow 1$$
$$V((16, 10, no)) \leftarrow 1$$
$$V((19, 10, no)) \leftarrow 1$$

Monte Carlo Policy Evaluation

Given a policy, π , estimate the value function, $V(s)$, for all states, $s \in \mathcal{S}$

Monte Carlo Policy Evaluation (first visit):

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:

Generate an episode using π

For each state s appearing in the episode:

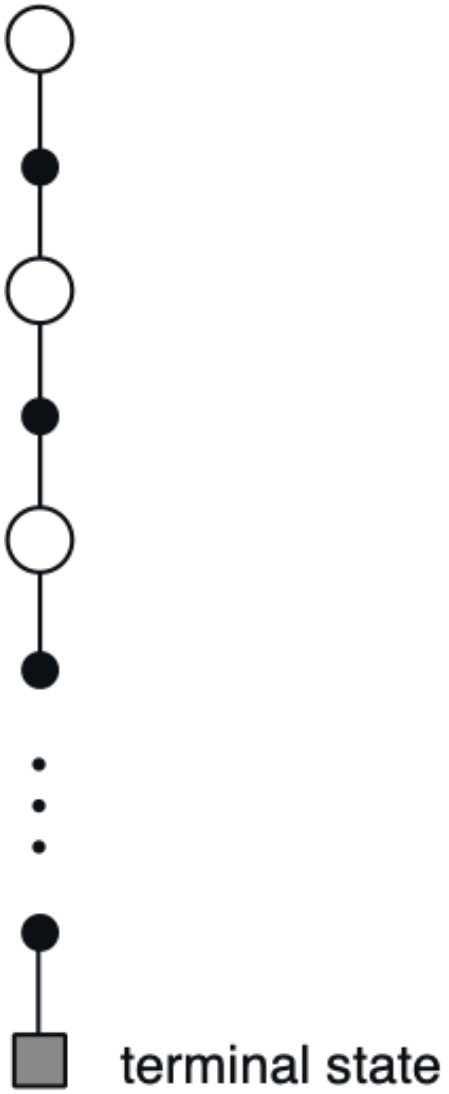
$G \leftarrow$ the return that follows the first occurrence of s

Append G to $Returns(s)$

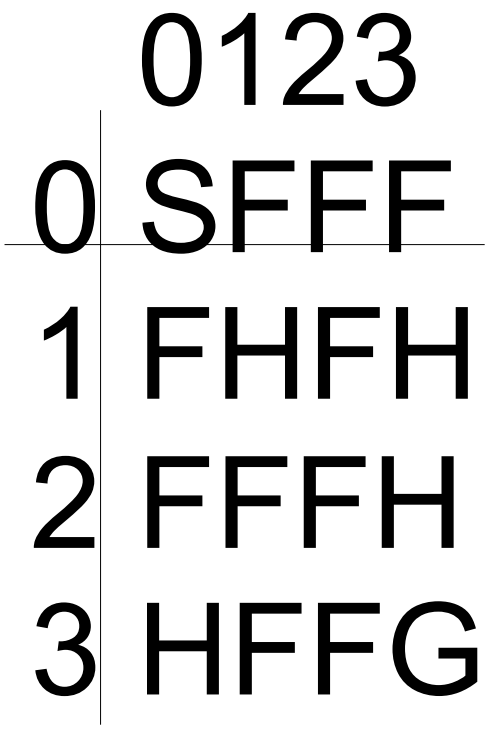
$V(s) \leftarrow \text{average}(Returns(s))$

Monte Carlo backup diagram

- Entire rest of episode included
- Only one choice considered at each state (unlike DP)
  - thus, there will be an explore/exploit dilemma
- Does not bootstrap from successor states's values (unlike DP)
- Time required to estimate one state does not depend on the total number of states



Think-pair-share: frozenlake env

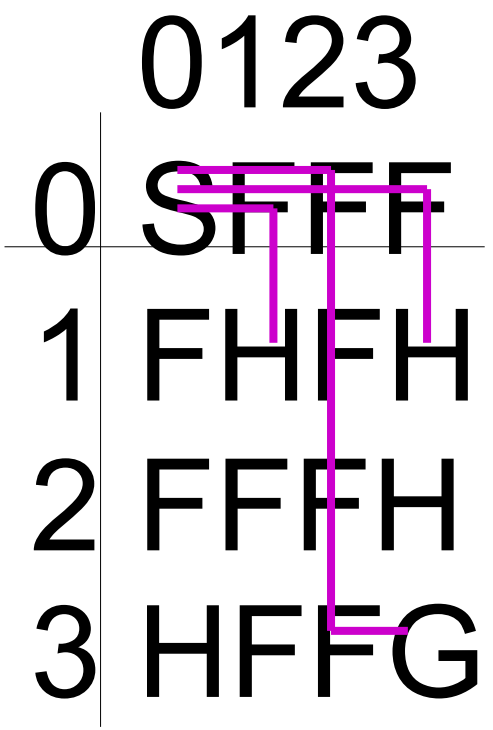


States: grid world coordinates

Actions: L, R, U, D

Reward: 0 except at G

Think-pair-share: frozenlake env



States: grid world coordinates

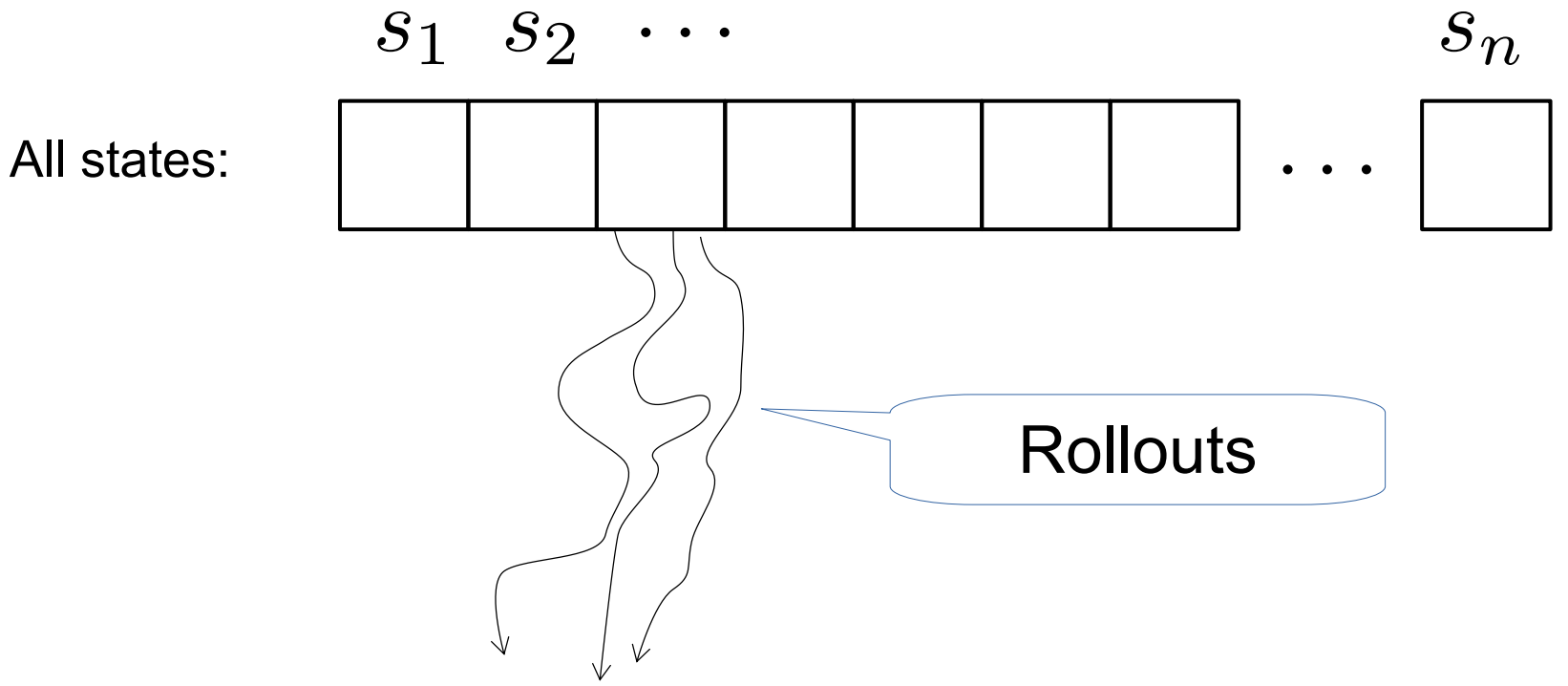
Actions: L, R, U, D

Reward: 0 except at G where r=1

Given: three episodes as shown

Calculate: values of states on top row as calculated by MC

Monte Carlo Policy Evaluation: General idea



For each state, we're going to "roll out" the policy until episode ends.

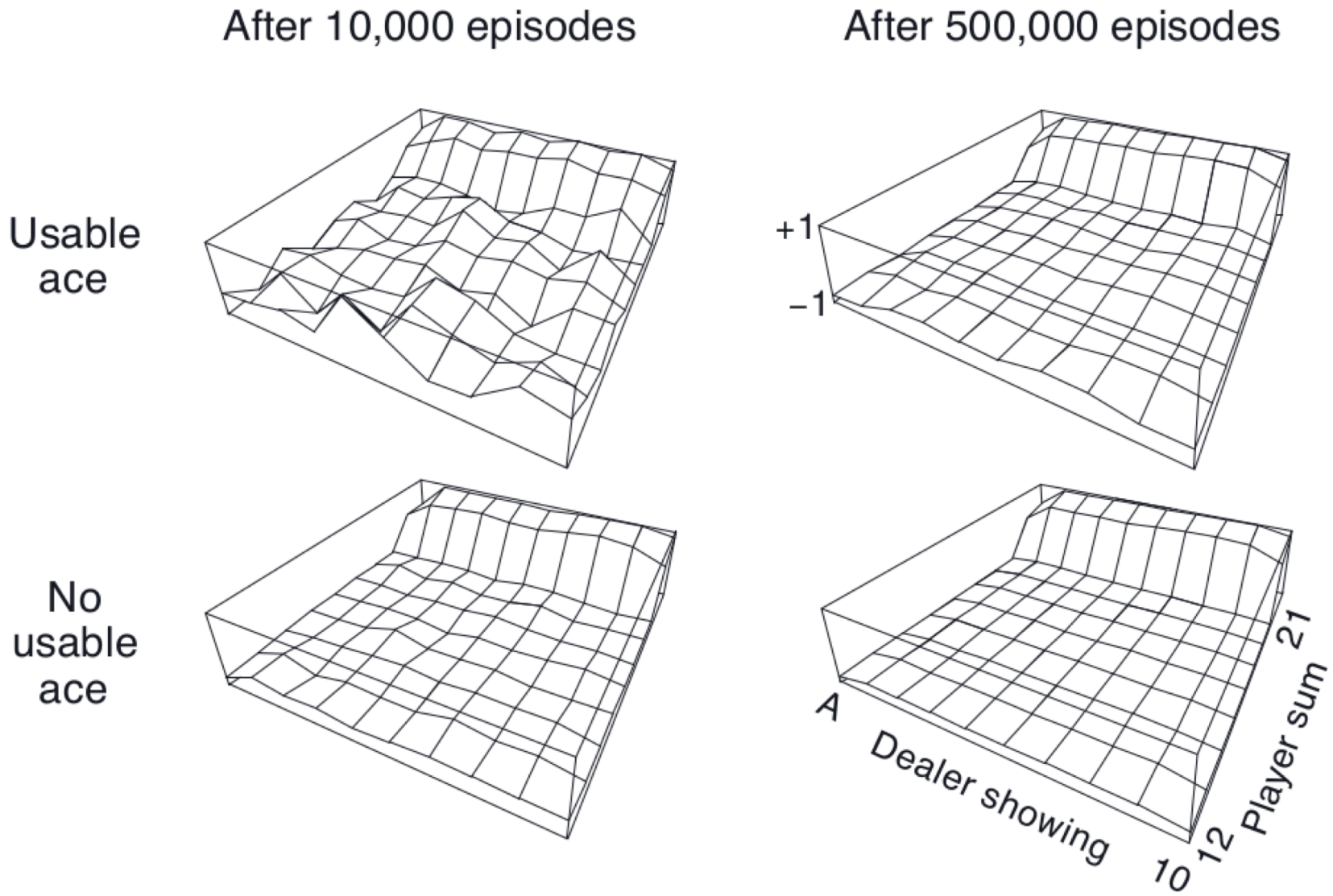
– need to do this many times in order to evaluate a state accurately

– the value of that state is the average discounted reward over all rollouts

This works well for blackjack b/c each game starts in a randomly selected state

– forces us to explore the entire state space

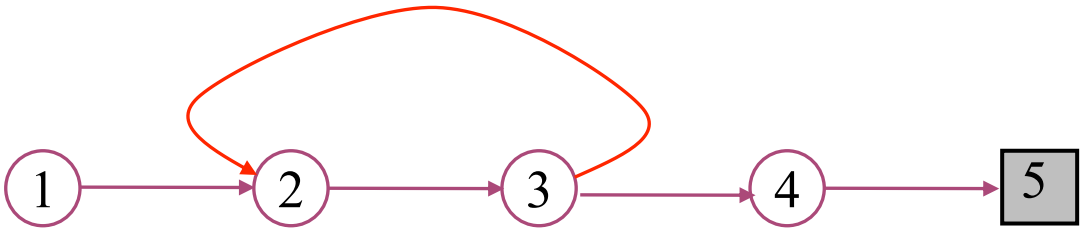
Monte Carlo Policy Evaluation: Example



Value function learned for “hit everything except for 20 and 21” policy

Monte Carlo Policy Evaluation

- *Goal:* learn $v_\pi(s)$
- *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 asympotically

Monte Carlo Action-Value Policy Evaluation

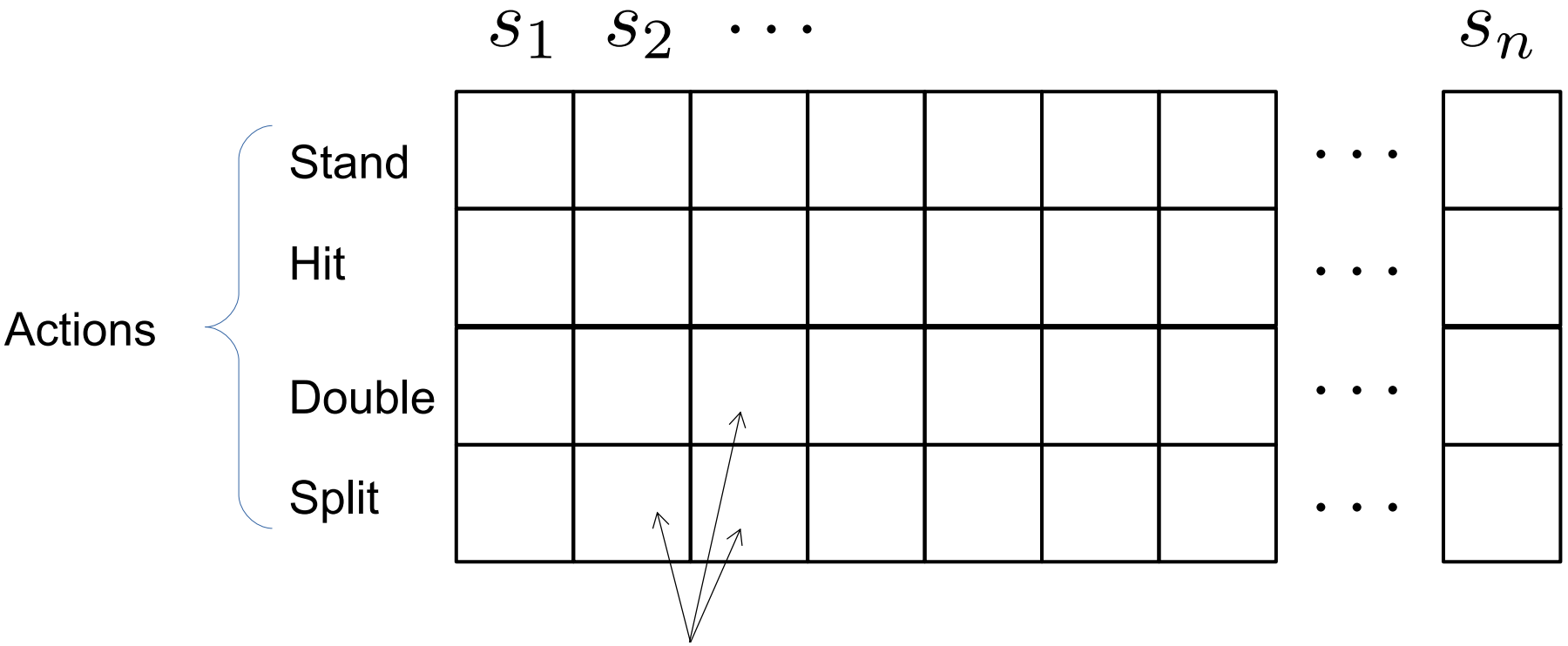
Two different types of policy evaluation:

1) state-value evaluation: estimate $V^{\pi}(s_t = s)$

State-value fn

2) action-value evaluation: estimate $Q^{\pi}(s_t = s, a_t = a)$

Action-value fn



Monte Carlo Action-Value Policy Evaluation

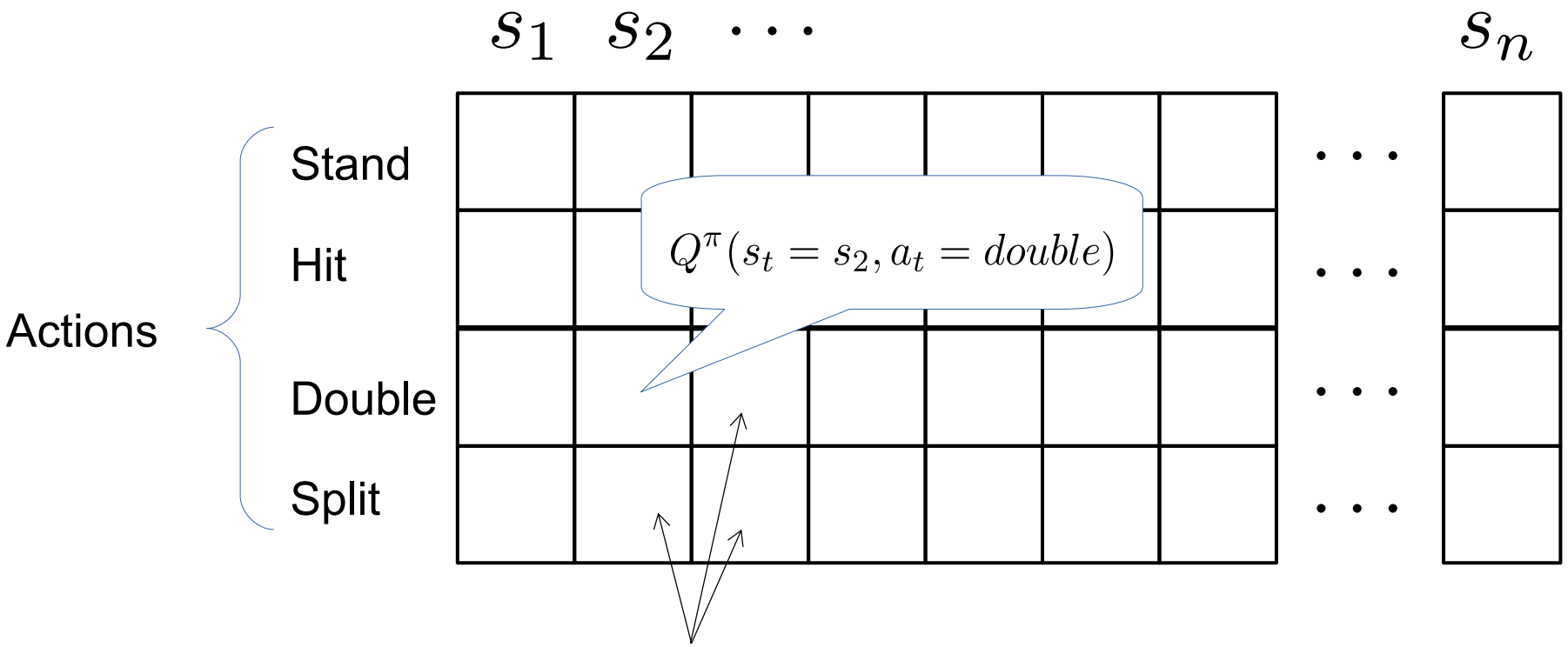
Two different types of policy evaluation:

1) state-value evaluation: estimate $V^{\pi}(s_t = s)$

State-value fn

2) action-value evaluation: estimate $Q^{\pi}(s_t = s, a_t = a)$

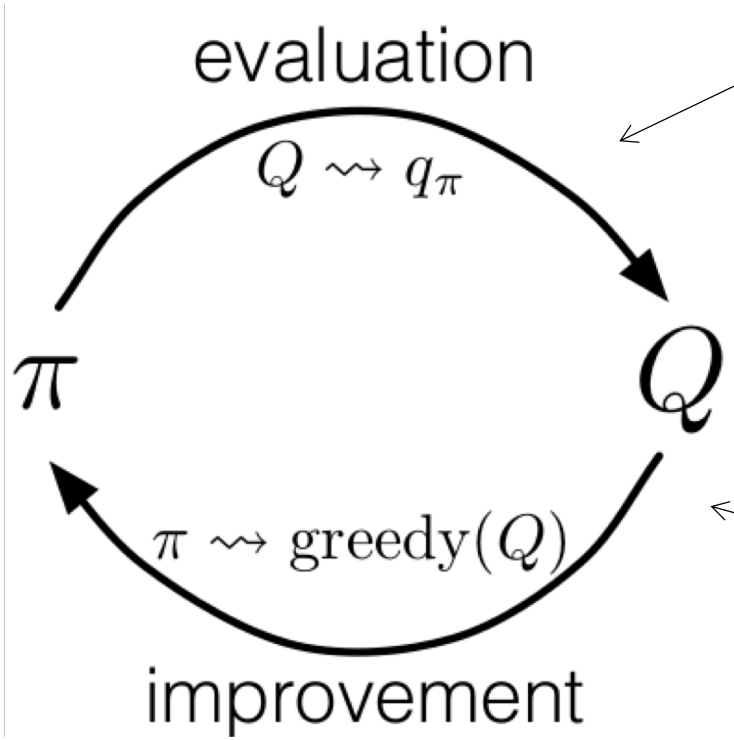
Action-value fn



Monte Carlo Control

So far, we're only talking about policy *evaluation*

… but RL requires us to find a policy, not just evaluate it… How?



Estimate $Q^\pi(s_t = s, a_t = a)$ via rollouts

$$\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q^\pi(s_t = s, a_t = a)$$

Key idea: evaluate/improve policy iteratively...

Monte Carlo Control

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

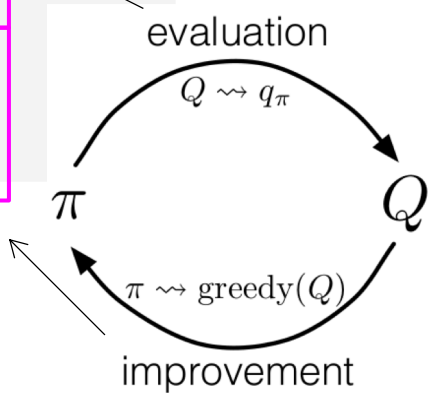
Repeat forever:
    Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
    Generate an episode starting from S_0, A_0, following π
    For each pair s, a appearing in the episode:
        $G \leftarrow$ the return that follows the first occurrence of s, a
        Append G to $Returns(s, a)$
        $Q(s, a) \leftarrow$ average$(Returns(s, a))$
    For each s in the episode:
        $\pi(s) \leftarrow \text{argmax}_a \, Q(s, a)$

Monte Carlo Control

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
Generate an episode starting from S_0, A_0, following π
For each pair s,a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s,a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

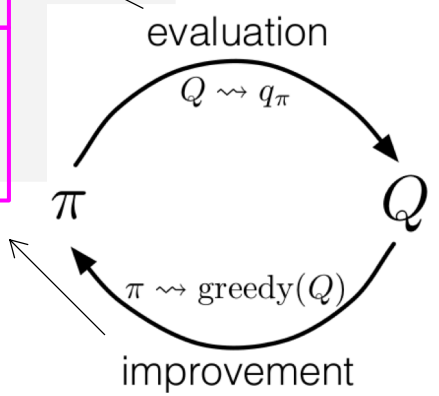


Think-pair-share

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
Generate an episode starting from S_0, A_0, following π
For each pair s,a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s,a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$



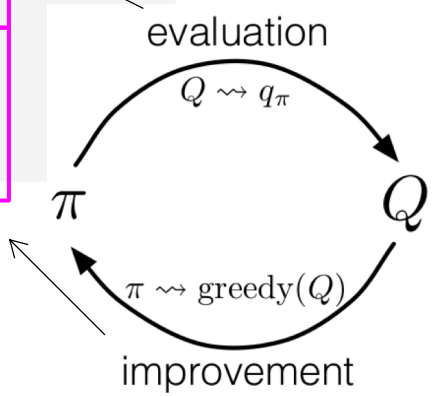
How would you make a Monte Carlo version of policy iteration?

Question

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
Generate an episode starting from S_0, A_0, following π
For each pair s,a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s,a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$



Is this more like policy iteration, value iteration, or something else?

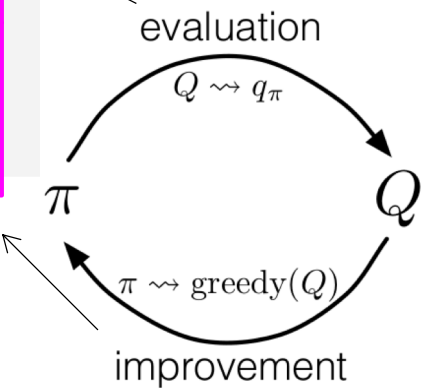
Monte Carlo Control

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probal > 0
Generate an episode starting from S_0, A_0, following π
For each pair s,a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s,a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Notice there is only one step of policy evaluation
– that's okay.
– each evaluation iter moves value fn toward its optimal value. Good enough to improve policy.



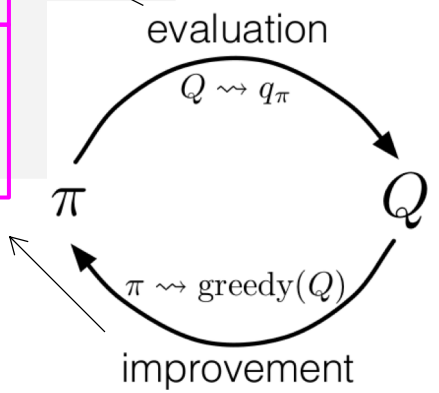
Monte Carlo Control

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probab > 0
Generate an episode starting from S_0, A_0, following π
For each pair s,a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s,a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Just like DP: evaluation will improve or policy will improve!



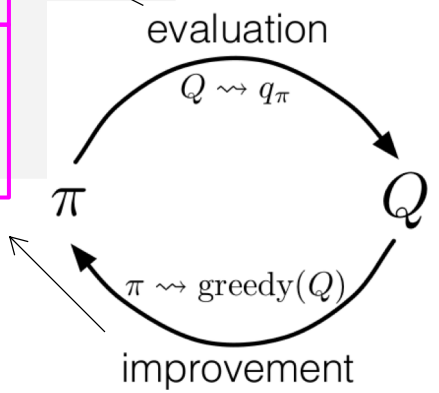
Monte Carlo Control

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probab > 0
Generate an episode starting from S_0, A_0, following π
For each pair s,a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s,a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow$ average$(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Just like DP: evaluation will improve or policy will improve!



Monte Carlo Control

Monte Carlo, Exploring Starts

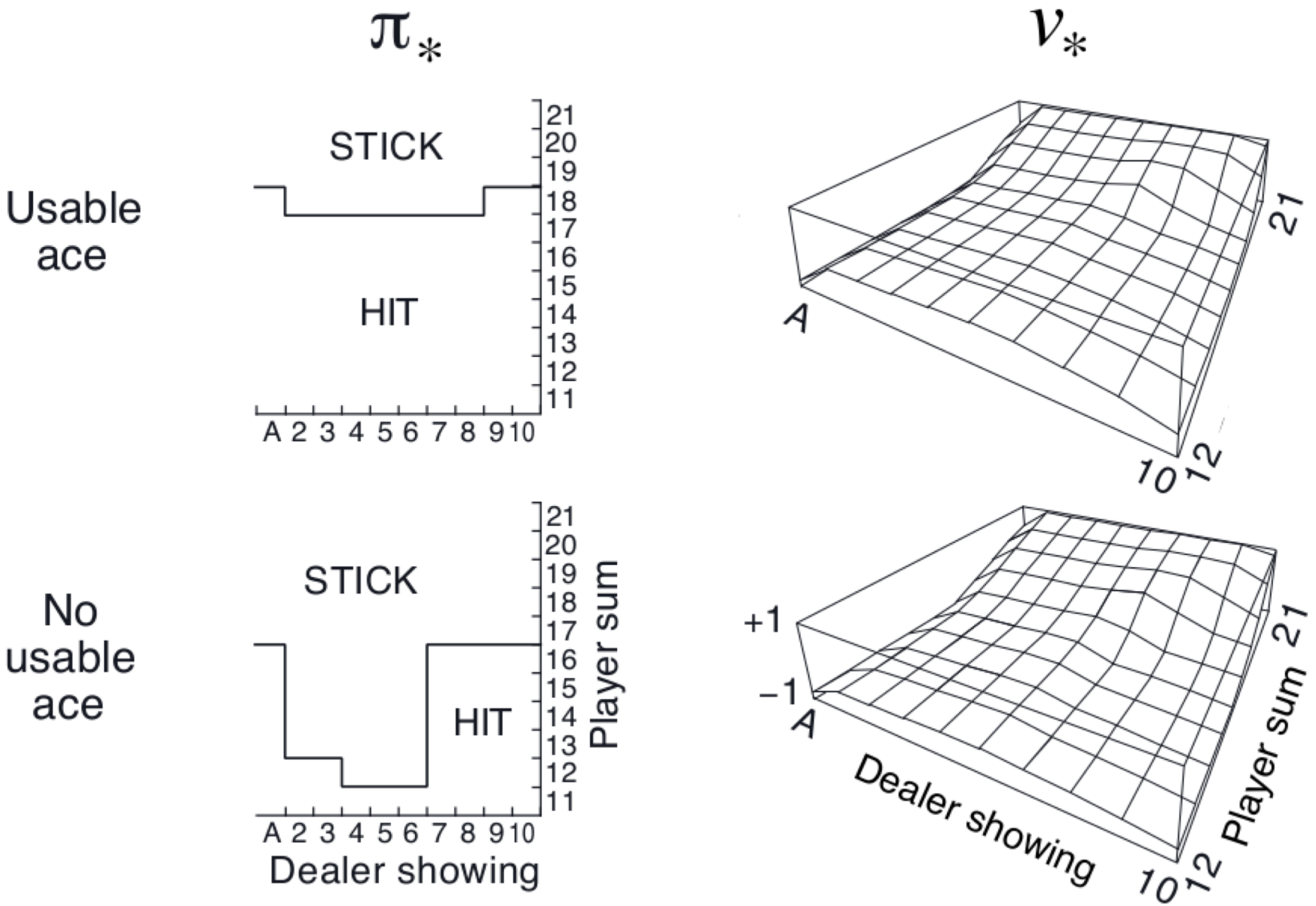
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$\pi(s) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
Generate an episode starting from S_0, A_0, following π
For each pair s, a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

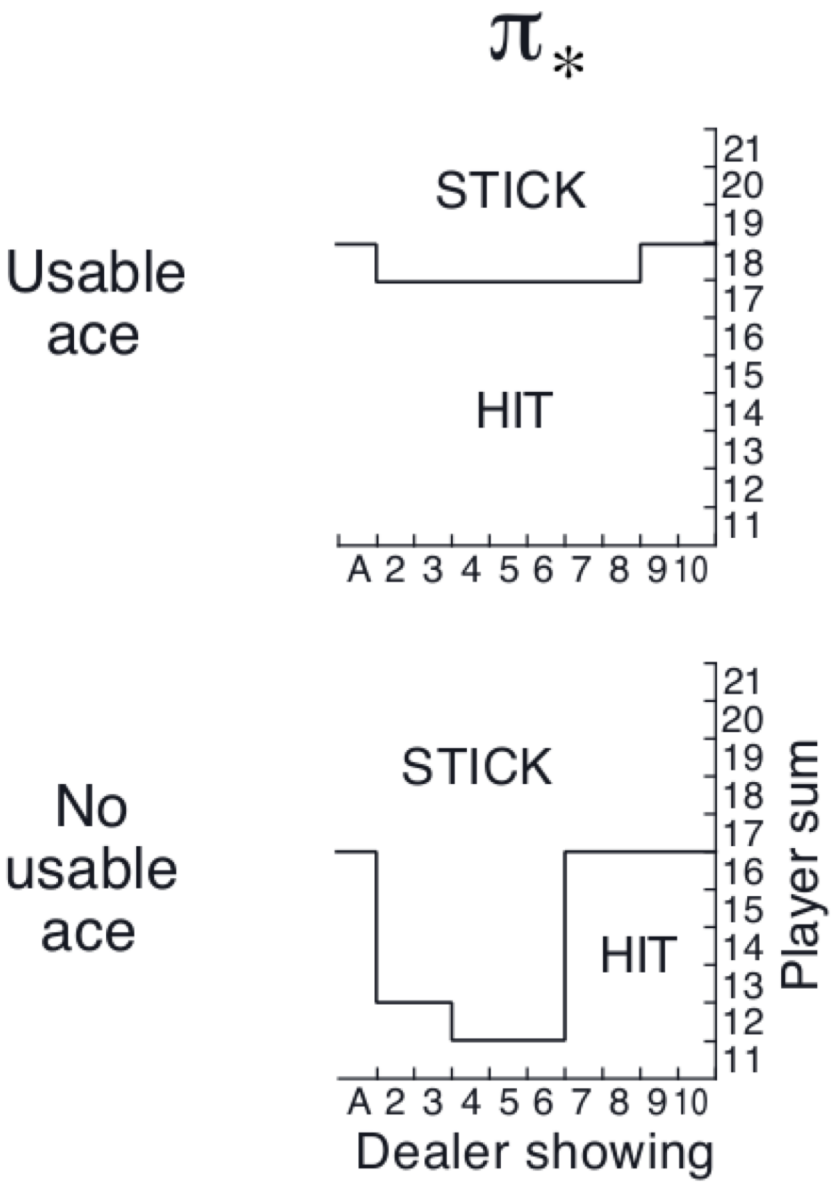
Exploring starts:
– each episode starts with a random action taken from a random state

Why do you need exploring starts?

Monte Carlo Control



Monte Carlo Control



What the MC agent learned

Dealer's Up Card

← Your Hand →	2	3	4	5	6	7	8	9	10	A
17+	S	S	S	S	S	S	S	S	S	S
16	S	S	S	S	S	H	H	H	H	H
15	S	S	S	S	S	H	H	H	H	H
14	S	S	S	S	S	H	H	H	H	H
13	S	S	S	S	S	H	H	H	H	H
12	H	H	S	S	S	H	H	H	H	H
11	D	D	D	D	D	D	D	D	D	H
10	D	D	D	D	D	D	D	D	H	H
9	H	D	D	D	D	H	H	H	H	H
5 - 8	H	H	H	H	H	H	H	H	H	H
A 8 - 10	S	S	S	S	S	S	S	S	S	S
A, 7	S	D	D	D	D	S	S	H	H	H
A, 6	H	D	D	D	D	H	H	H	H	H
A, 5	H	H	D	D	D	H	H	H	H	H
A, 4	H	H	D	D	D	H	H	H	H	H
A, 3	H	H	H	D	D	H	H	H	H	H
A, 2	H	H	H	D	D	H	H	H	H	H
A,A 8,8	SP	SP	SP	SP	SP	SP	SP	SP	SP	SP
10, 10	S	S	S	S	S	S	S	S	S	S
9, 9	SP	SP	SP	SP	SP	S	SP	SP	S	S
7, 7	SP	SP	SP	SP	SP	SP	H	H	H	H
6, 6	SP	SP	SP	SP	SP	H	H	H	H	H
5, 5	D	D	D	D	D	D	D	D	H	H
4, 4	H	H	H	SP	SP	H	H	H	H	H
3, 3	SP	SP	SP	SP	SP	SP	H	H	H	H
2, 2	SP	SP	SP	SP	SP	SP	H	H	H	H
	2	3	4	5	6	7	8	9	10	A

If doubling down after splitting is not allowed, then just hit the following:
2,2 and 3,3 vs. 2 and 3 4,4 vs. 5 and 6 6,6 vs. 2

HIT | STAND | DOUBLE DOWN | SPLIT

The official "basic strategy"

MC Control: Convergence

- Greedified policy meets the conditions for policy improvement:

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &= v_{\pi_k}(s). \end{aligned}$$

- And thus must be $\geq \pi_k$ by the policy improvement theorem
- This assumes exploring starts and infinite number of episodes for MC policy evaluation
- To solve the latter:
  - update only to a given level of performance
  - alternate between evaluation and improvement per episode

MC Control: Convergence

- Greedified policy meets the conditions for policy improvement:

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s, a)) \\ &= \max_a q_{\pi_k}(s, a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &= v_{\pi_k}(s). \end{aligned}$$

- And thus must be ≥ π_k by the policy improvement theorem
- This assumes exploring starts and infinite [illegible] episodes for MC
- To [illegible]
  - [illegible]
  - [illegible]

Let π and π' be any pair of deterministic policies such that: $Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s), \forall s \in \mathcal{S}$

Then, policy π' must be as good or better than π .

E-greedy exploration

So far, we have avoided talked about exploration during RL

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

Repeat forever:
    Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
    Generate an episode starting from S_0, A_0, following π
    For each pair s, a appearing in the episode:
        $G \leftarrow$ the return that follows the first occurrence of s, a
        Append G to $Returns(s, a)$
        $Q(s, a) \leftarrow \text{average}(Returns(s, a))$
    For each s in the episode:
        $\pi(s) \leftarrow \text{argmax}_a \, Q(s, a)$

E-greedy exploration

So far, we have avoided talked about exploration during RL

Monte Carlo, Exploring Starts:

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

Repeat forever:
Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all pairs have probability > 0
Generate an episode starting from S_0, A_0, following π
For each pair s, a appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s, a
Append G to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
For each s in the episode:
$\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Without exploring starts, we are not guaranteed to explore the state/action space
– why is this a problem?
– what happens if we never experience certain transitions?

E-greedy exploration

So far, we have avoided talked about exploration during RL

Monte Carlo, Exploring Starts:

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

Repeat forever:
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all ave probability > 0
  Generate an episode starting fr
  For each pair s,a appearing in
    $G \leftarrow$ the return that follows
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow$ average$(Returns(s,a))$
  For each s in the episode:
    $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Without exploring starts, we are not guaranteed to explore the state/action space
– why is this a problem?
– what happens if we never experience certain transitions?

Can we accomplish this without exploring starts?

E-greedy exploration

So far, we have avoided talked about exploration during RL

Monte Carlo, Exploring Starts:

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

Repeat forever:
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ s.t. all ave probability > 0
  Generate an episode starting fr
  For each pair s,a appearing in
    $G \leftarrow$ the return that follows
    Append G to $Returns(s,a)$
    $Q(s,a) \leftarrow$ average($Return$
  For each s in the episode:
    $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$

Without exploring starts, we are not guaranteed to explore the state/action space
– why is this a problem?
– what happens if we never experience certain transitions?

Can we accomplish this without exploring starts?

Yes: create a stochastic (e-greedy) policy

E-greedy exploration

Greedy policy at state s :

$$\pi(a|s) = a^* \quad \text{with probability 1, where} \quad a^* = \arg\max_{a \in \mathcal{A}} Q(s,a)$$

(i.e. $\pi(s) = a^*$)

E-Greedy policy at state s :

$$\pi(a|s) = \begin{cases} a^* & \text{with probability } 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} \\ a \sim \mathcal{A} \setminus a^* & \text{with probability } \frac{\epsilon}{|\mathcal{A}|} \end{cases}$$

E-greedy exploration

Greedy policy at state s :

$$\pi(a|s) = a^* \quad \text{with probability 1, where} \quad a^* = \arg\max_{a \in \mathcal{A}} Q(s,a)$$

(i.e. $\pi(s) = a^*$)

E-Greedy policy at state s :

$$\pi(a|s) = \begin{cases} a^* & \text{with probability } 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} \\ a \sim \mathcal{A} \setminus a^* & \text{with probability } \frac{\epsilon}{|\mathcal{A}|} \end{cases}$$

Action a drawn uniformly from set $\mathcal{A}$

E-greedy exploration

Greedy policy at state s :

$$\pi(a|s) = a^*$$

Guarantees every state/action will be visited infinitely often

E-Greedy policy at state s :

$$\pi(a|s) = \begin{cases} a^* & \text{with probability } 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}|} \\ a \sim \mathcal{A} \setminus a^* & \text{with probability } \frac{\epsilon}{|\mathcal{A}|} \end{cases}$$

– Notice that this is a stochastic policy (not deterministic)
– This is an example of an *e-soft* policy
– E-soft: any policy where all actions have non-zero probability

E-greedy exploration

Monte Carlo, e-greedy exploration:

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

Repeat forever:
(a) Generate an episode using π
(b) For each pair s, a appearing in the episode:
$G \leftarrow$ the return that follows 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:
$A^* \leftarrow \arg\max_a Q(s,a)$ (with ties broken arbitrarily)
For all $a \in \mathcal{A}(s)$:

$$\pi(a|s) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(s)| & \text{if } a \neq A^* \end{cases}$$

E-greedy exploration

On-policy Monte Carlo Control

- *On-policy:* learn about policy currently executing

- How do we get rid of exploring starts?
  - The policy must be eternally *soft*:
    - $\pi(a|s) > 0$ for all s and a

- Similar to GPI: move policy *towards* greedy policy (e.g., ε-greedy)

- Converges to best ε-soft policy

Off-policy methods

- Can I learn a (hard/deterministic) greedy policy while executing a soft policy? (learn an optimal policy while exploring!)
- Learn the value of the *target policy* π from experience due to *behavior policy b*
- For example, π is the greedy policy (and ultimately the optimal policy) while μ is exploratory (e.g., ε-soft)
- In general, we only require *coverage*, i.e., that *b* generates behavior that covers, or includes, π

$\pi(a|s) > 0$ for every *s,a* at which $b(a|s) > 0$

- Idea: *importance sampling*
  - Weight each return by the *ratio of the probabilities* of the trajectory under the two policies
- We are going to start with fixed (but different) π and *b*

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under π:

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

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under π:

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

- In importance sampling, each return is weighted by the relative probability of the trajectory under the two policies

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

- This is called the *importance sampling ratio*

Importance Sampling Ratio

- Probability of the rest of the trajectory, after S_t, under π:

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

- In importance sampling, each return is weighted by the relative probability of the trajectory under the two policies

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

- This is called the *importance sampling ratio*
- All importance sampling ratios have expected value 1

$$\mathbb{E}\left[\frac{\pi(A_k|S_k)}{b(A_k|S_k)}\right] \doteq \sum_a b(a|S_k)\frac{\pi(a|S_k)}{b(a|S_k)} = \sum_a \pi(a|S_k) = 1$$

Importance sampling

- What can we do with the importance sampling ratio?

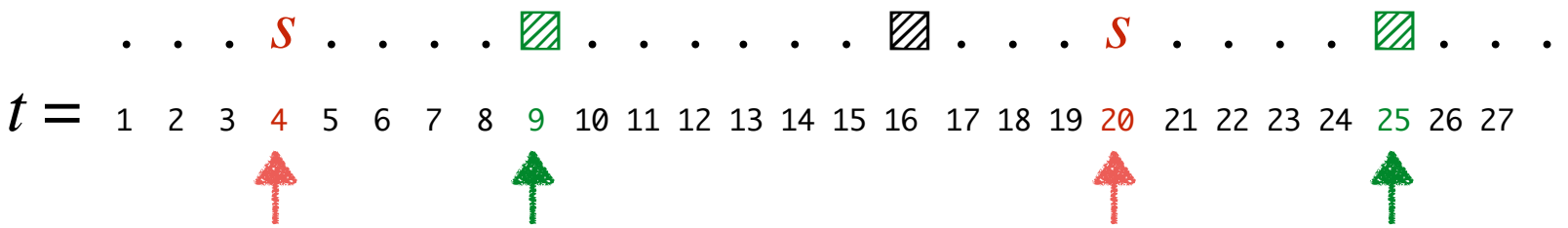
Importance sampling

- What can we do with the importance sampling ratio?

- Reweight the returns from *b* to have the correct expectation according to π!

Importance Sampling

- New notation: time steps increase across episode boundaries:



$\mathcal{T}(s) = \{4, 20\}$ set of start times

$T(4) = 9 \quad T(20) = 25$ next termination times

- *Ordinary importance sampling* forms estimate

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

Return starting at *t*

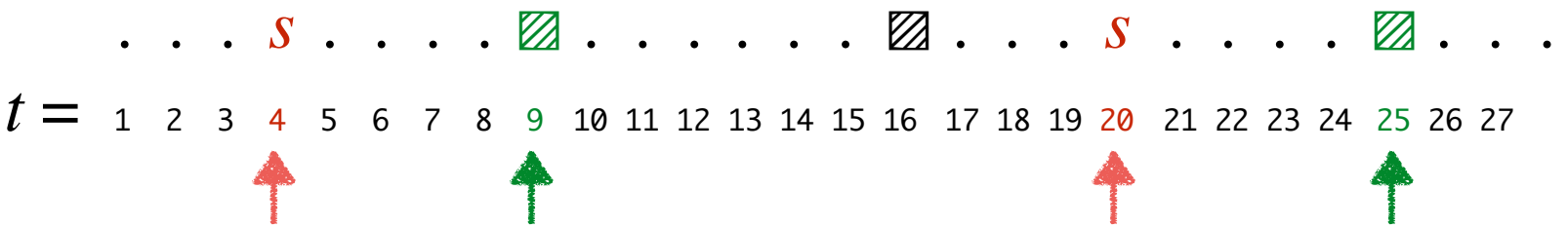
Reweight return over whole trajectory

Number of times *s* is visited

Scaling the returns and averaging them

Importance Sampling

- New notation: time steps increase across episode boundaries:



$\mathcal{T}(s) = \{4, 20\}$ set of start times

$T(4) = 9 \quad T(20) = 25$ next termination times

- *Ordinary importance sampling* forms estimate

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

Return starting at t

Number of times *s* is visited

Reweight return

- Whereas *weighted importance sampling* forms estimate

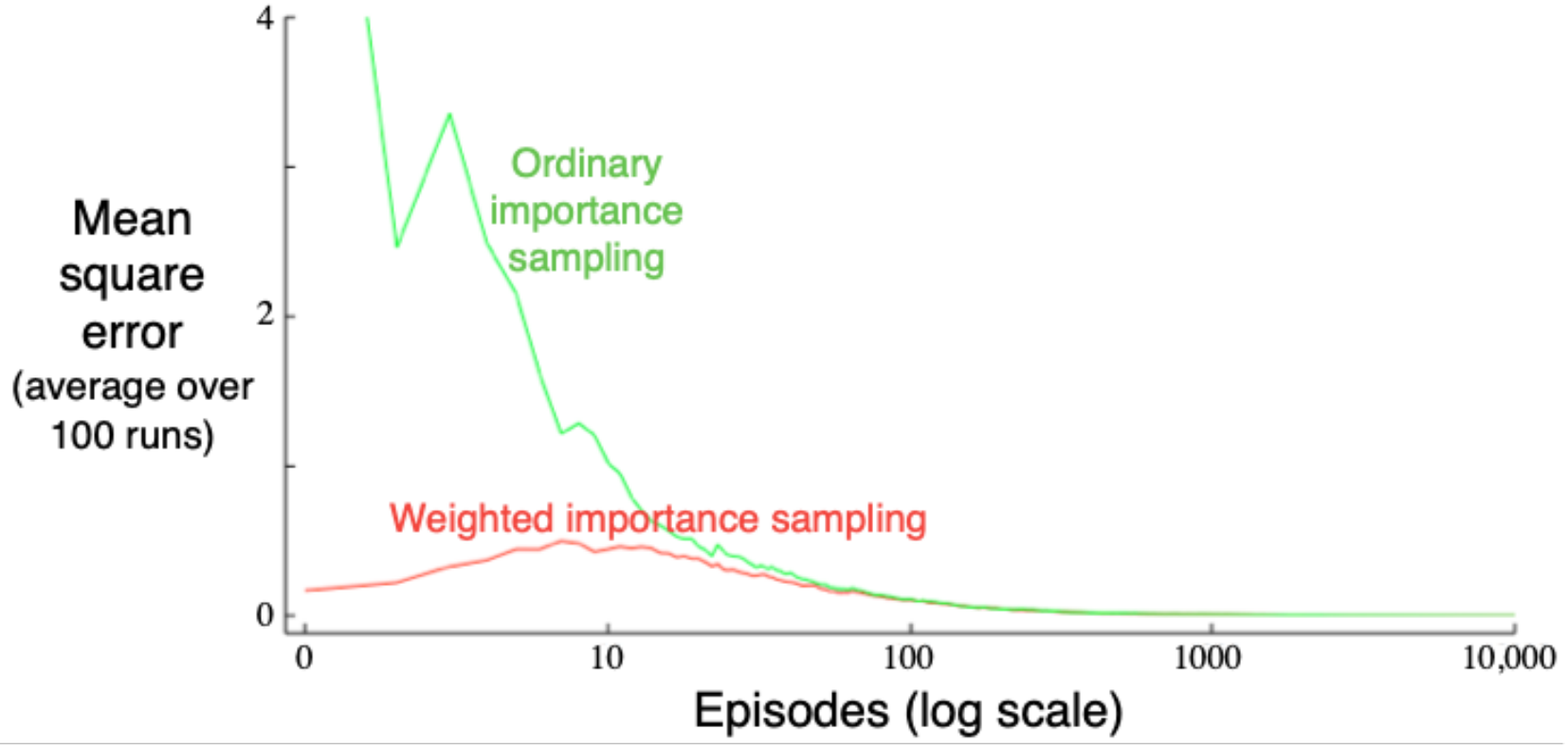
$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

As a result, this has bias, but lower variance

The importance ratio again

Example: The value of a *single* Blackjack state

- State is player-sum 13, dealer-showing 2, useable ace
- Target policy is stick only on 20 or 21
- Behavior policy is equiprobable
- True value ≈ −0.27726



MC Summary

- MC methods estimate value function by doing rollouts
- Can estimate either the state value function, $Q(s, a)$, or the action value function, $V(s)$
- MC Control alternates between policy evaluation and policy improvement
- E-greedy exploration explores all possible actions while preferring greedy actions
- MC has several advantages over DP:
  - Can learn directly from interaction with environment
  - No need for full models
- On-policy methods often learn faster than off-policy methods, but require learning about an exploration policy