Monte Carlo Approaches to Reinforcement Learning

Chris Amato
Northeastern University

with some slides from Rob Platt, Lawson Wong and UAlberta

Announcements

- Exercise 3 (DP) due tomorrow (Oct 7)
- Exercise 4 (Monte Carlo) out next week

Summary: Dynamic Programming

- Value iteration

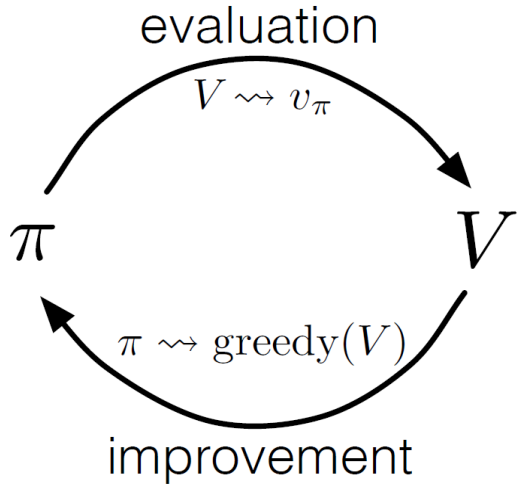
$$V(s) \leftarrow \max_a \sum_{s',r} p(s', r \,|\, s, a)\left[r + \gamma V(s')\right]$$

- Policy evaluation

$$V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s))\left[r + \gamma V(s')\right]$$

- Policy iteration

"Almost all reinforcement learning methods are well described as generalized policy iteration."



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

where $\xrightarrow{\text{E}}$ denotes a policy *evaluation* and $\xrightarrow{\text{I}}$ denotes a policy *improvement*

From planning to learning

- Value iteration

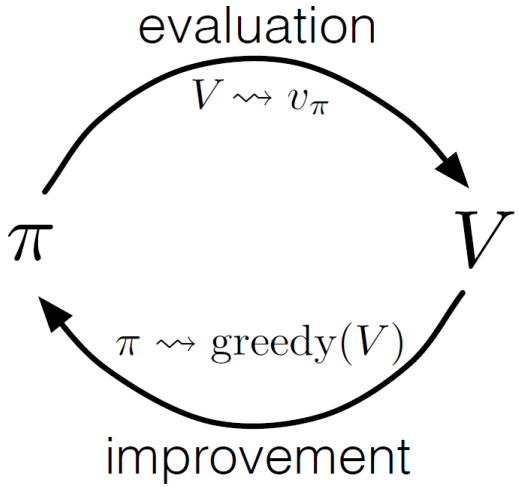
$$V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$$

- Policy evaluation

$$V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\left[r + \gamma V(s')\right]$$

- Policy iteration

"Almost all reinforcement learning methods are well described as generalized policy iteration."



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

where $\xrightarrow{\text{E}}$ denotes a policy *evaluation* and $\xrightarrow{\text{I}}$ denotes a policy *improvement*

From planning to learning

- Value iteration

$$V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$$

- Polic

$V(s)$ $(s')]$

- Polic

"Almos
well de

π_0

valuation
$V \rightsquigarrow v_\pi$
V
$\rightsquigarrow \text{greedy}(V)$
provement

where $\xrightarrow{\text{E}}$ denotes a policy *evaluation* and $\xrightarrow{\text{I}}$ denotes a policy *improvement*

So far, our algorithms require the model p(s', r | s, a)

Can we learn from experience instead, via simulator or the real world?

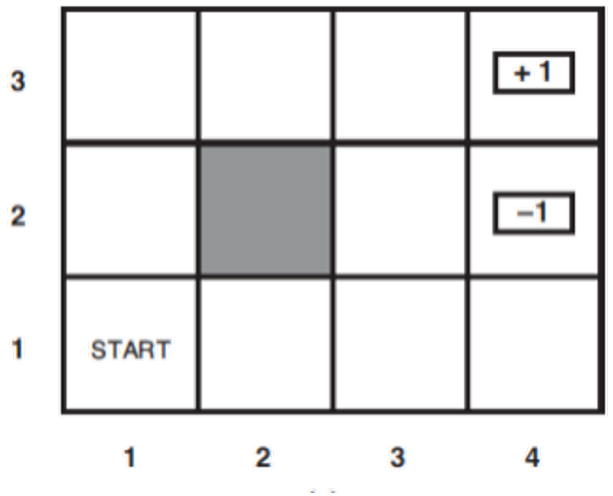
Monte Carlo Methods

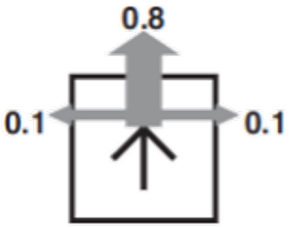
- Our first *learning* method for MDPs
- Often hard to actually know transition and reward values or the problem is too big to repeatedly iterate over states and actions
- Monte Carlo methods estimate the return using samples starting from a given state
- Monte Carlo methods learn from *complete* sample 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

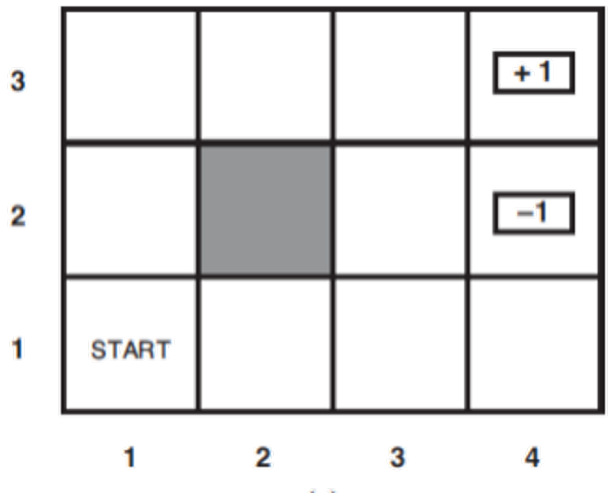


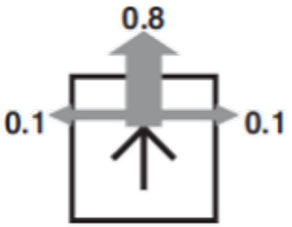


We know the probabilities of moving in each direction when an action is executed

Reinforcement Learning

We don't know the reward function

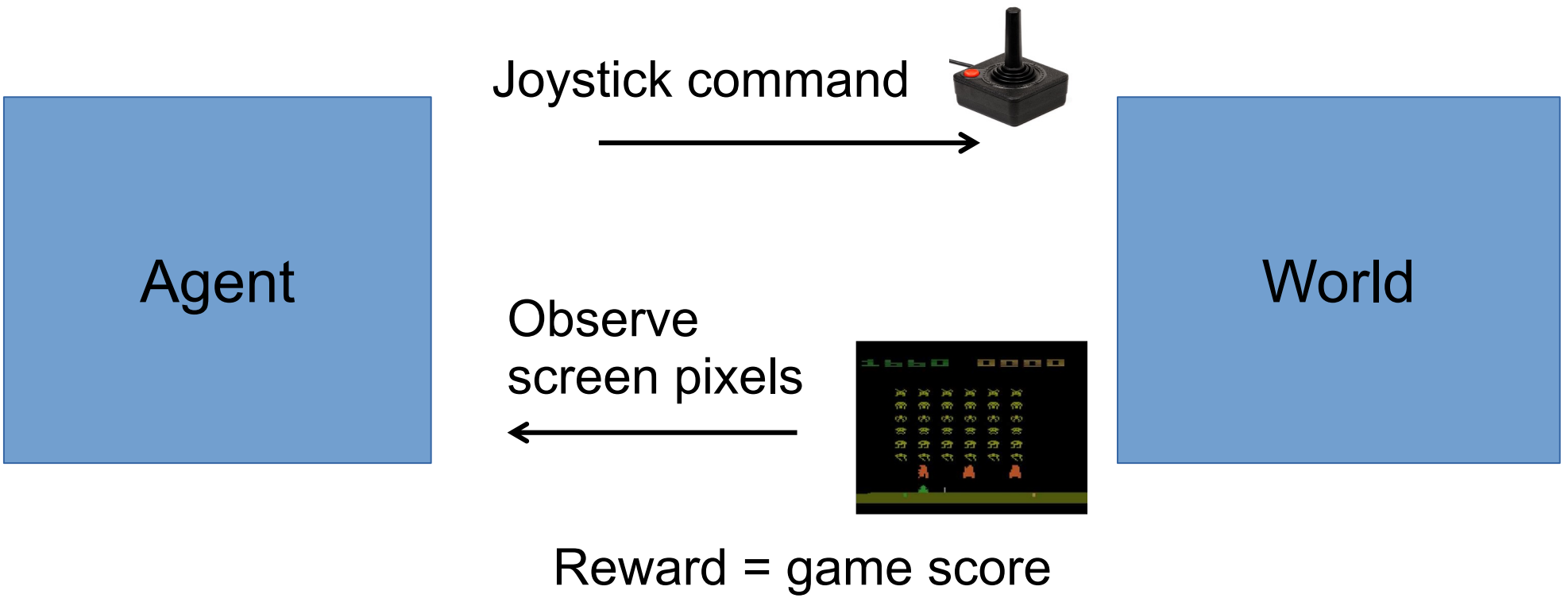




We don't know the probabilities of moving in each direction when an action is executed

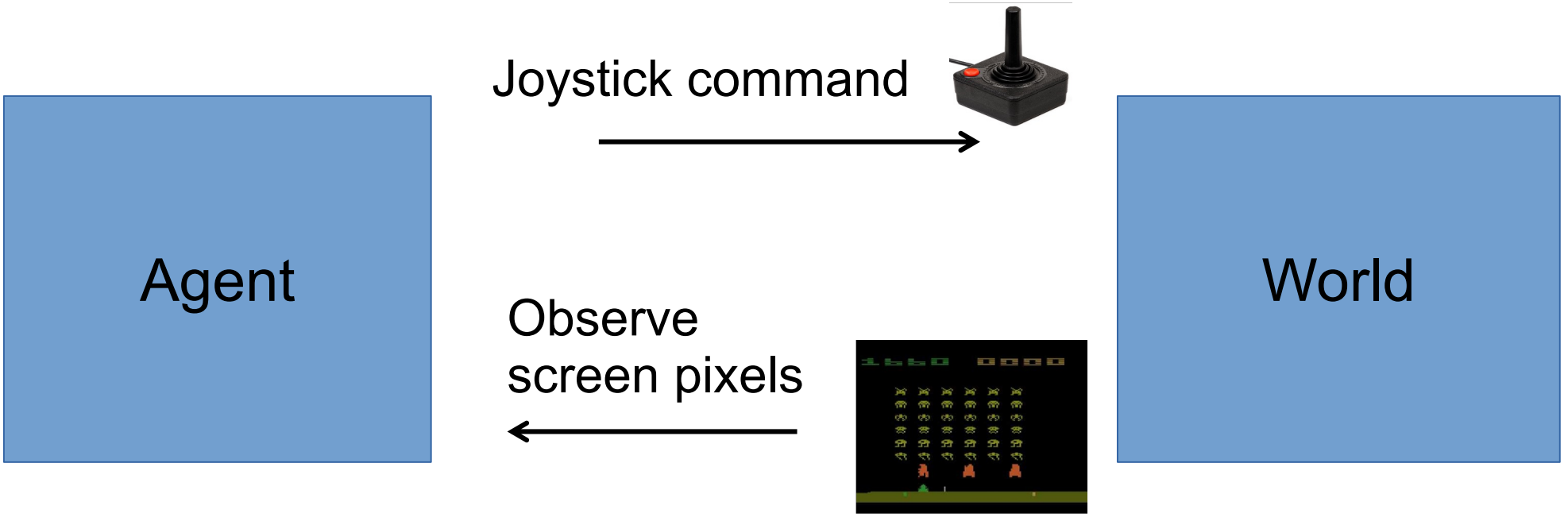
RL still assumes that we have an MDP

Model-Free 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)

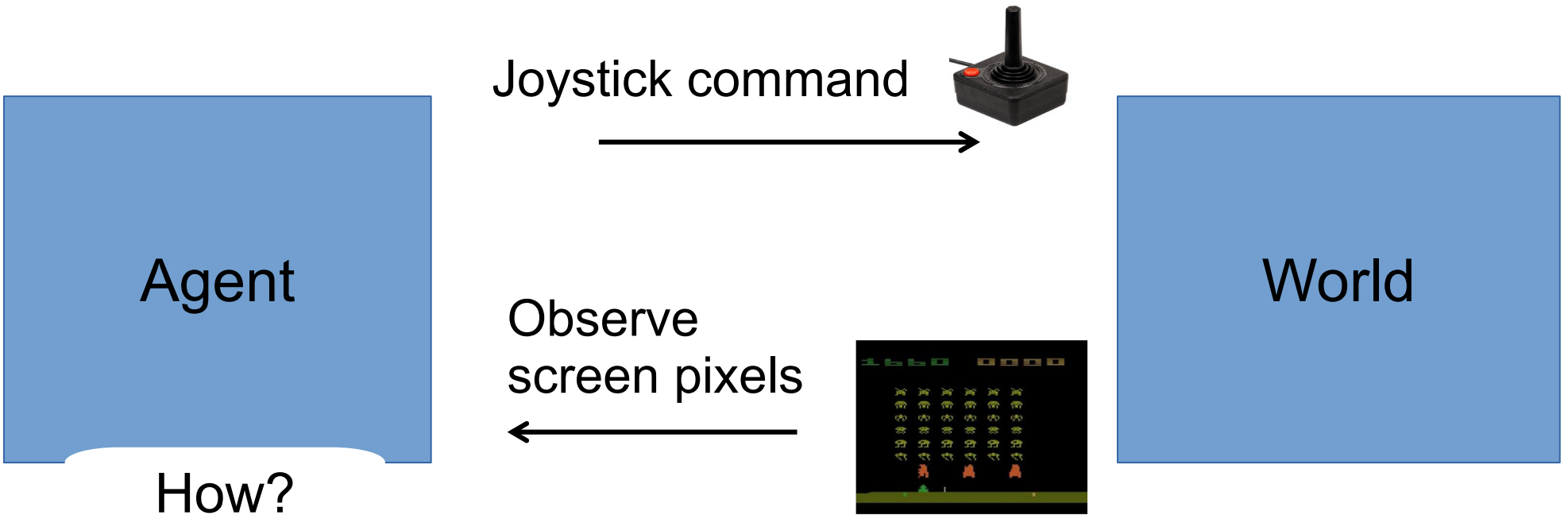
Model-Free Reinforcement Learning



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



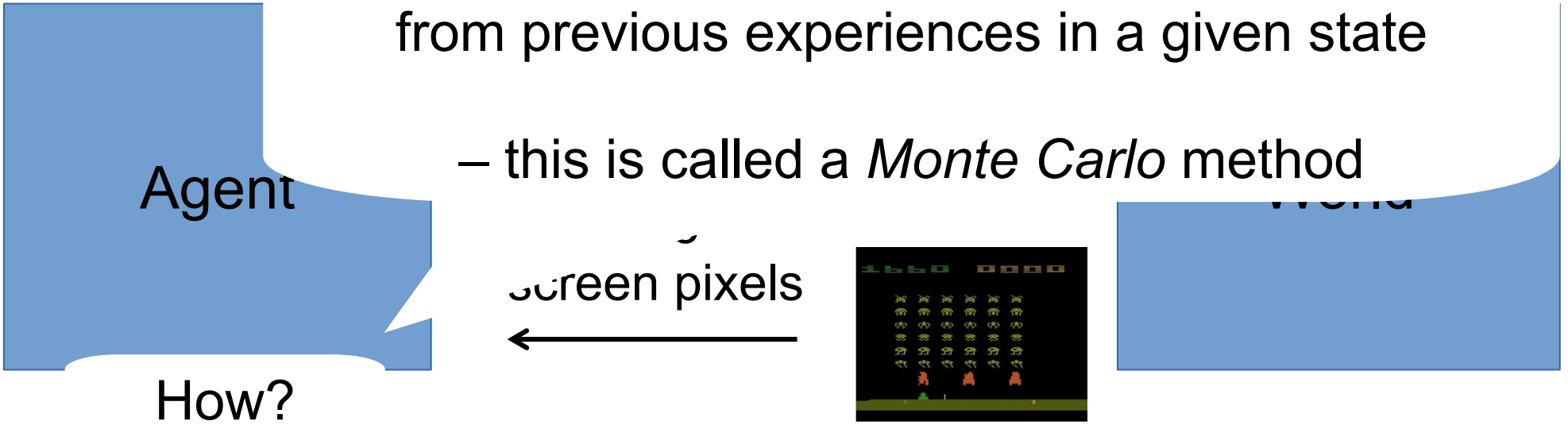
Goal: learn a value function through trial-and-error experience

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

Model-Free Reinforcement Learning

Simplest solution: average all outcomes from previous experiences in a given state

– this is called a *Monte Carlo* method

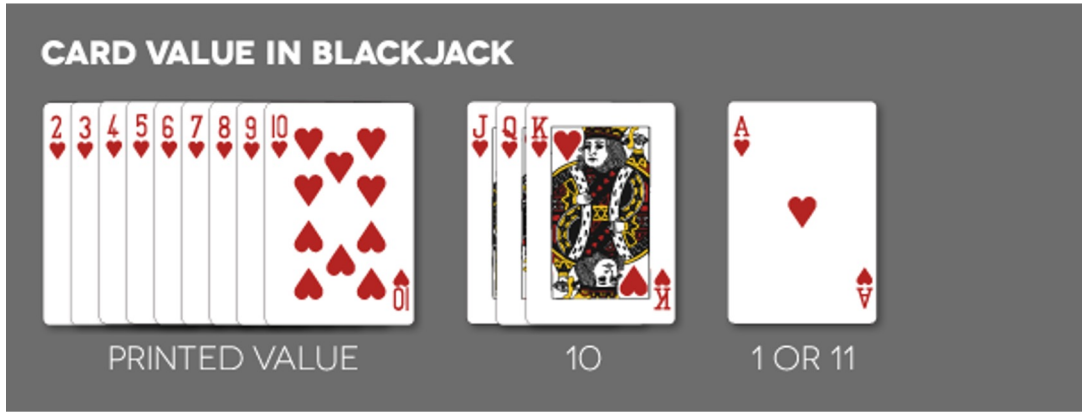


Reward = game score

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

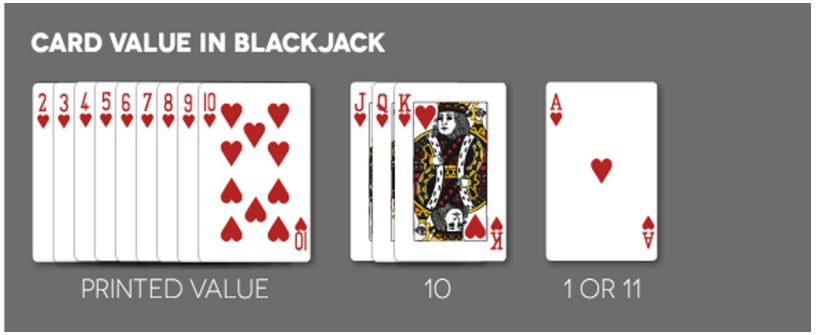
Objective: Have your card sum be greater than the dealer's without exceeding 21

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

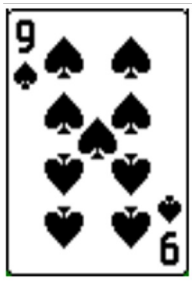
Append G to $Returns(s)$

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

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:

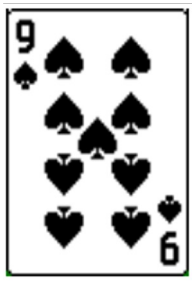


State	Action	Next State	Reward

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:

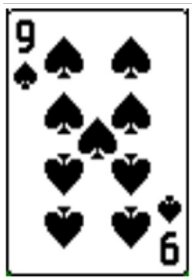


State	Action	Next State	Reward
19, 10, no			

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:



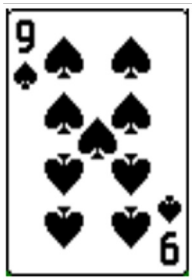
Agent sum, dealer's card, ace?

State	Action	Next State	Reward
19, 10, no			

Monte Carlo Policy Evaluation: Example

Dealer card:

Agent's hand:



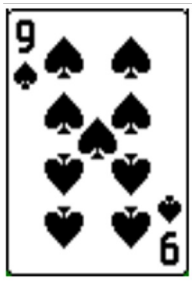
Agent sum, dealer's card, ace?

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

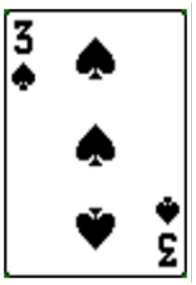
Upon episode termination, make the following value function updates:

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

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:



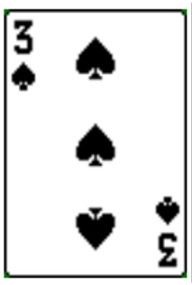
State	Action	Next State	Reward

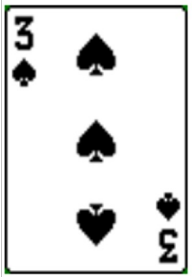
13, 10, no

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:



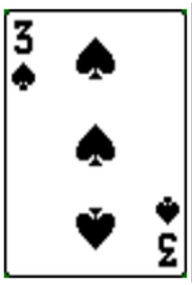


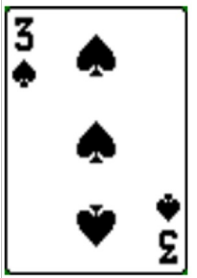
State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:



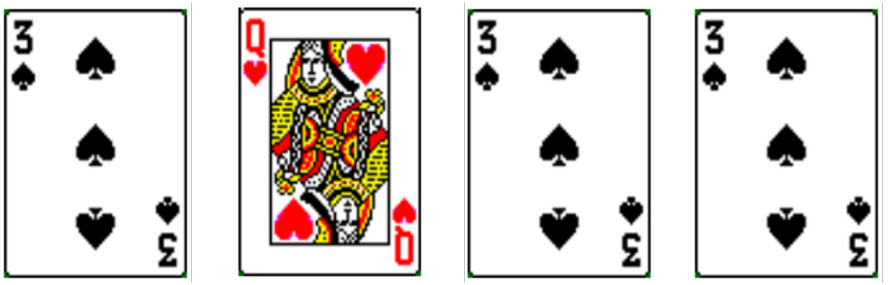


State	Action	Next State	Reward
13, 10, no 16, 10, no	HIT	16, 10, no	0

Monte Carlo Policy Evaluation: Another Example

Dealer card:

Agent's hand:

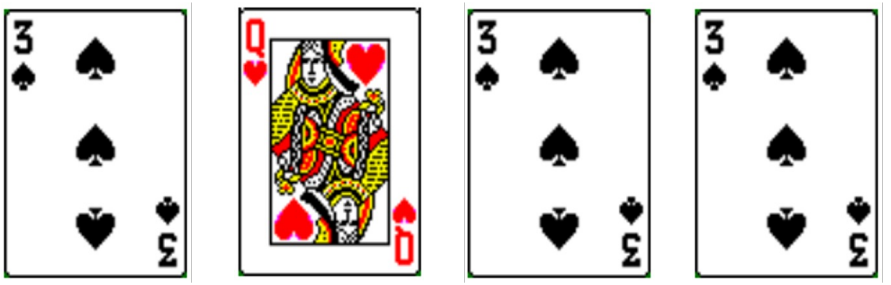


State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0
16, 10, no	HIT	19, 10, no	0

Monte Carlo Policy Evaluation: Another Example

Dealer card:

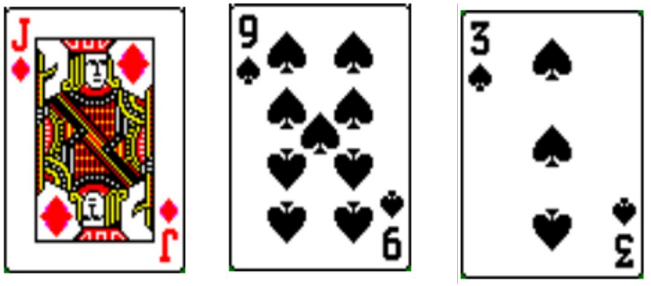
Agent's hand:



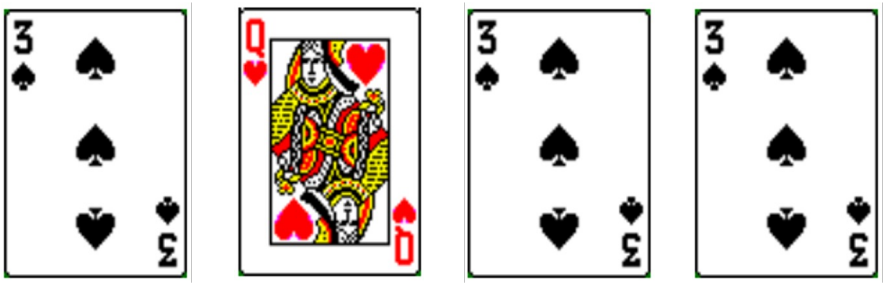
State	Action	Next State	Reward
13, 10, no	HIT	16, 10, no	0
16, 10, no	HIT	19, 10, no	0
19, 10, no			

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:

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

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

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

Remember: Multi-armed bandit

Learning: Estimate the Q-values (expected rewards)

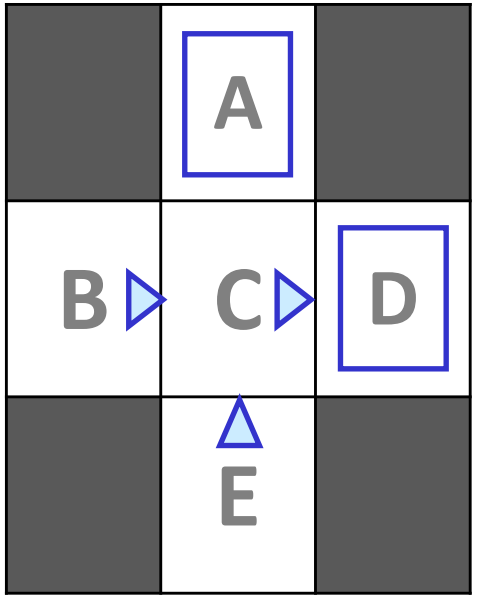
$$q_*(a) \doteq \mathbb{E}[R_t \mid A_t = a]$$

Given some experience pulling an arm, how to estimate?

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

Example: Monte Carlo evaluation

States: A,B,C,D,E
Actions: left, right, up, down, exit



Blue arrows denote policy that agent is following during learning (exploration policy)

Observations
(from 4 episodes):
(s, a, s', r) tuples

1. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

Episode 1 consists of three steps

2. E, up, C, -1
   C, right, A, -1
   A, exit, term, -10

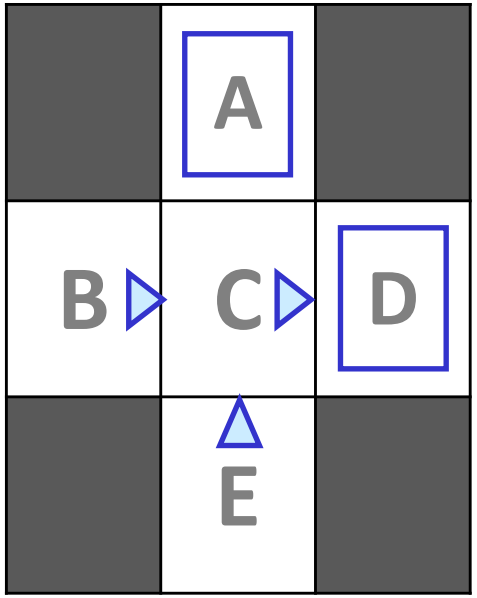
Transition noise

3. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

4. E, up, C, -1
   C, right, D, -1
   D, exit, term, 10

Example: Monte Carlo evaluation

States: A,B,C,D,E
Actions: left, right, up, down, exit



Blue arrows denote policy that agent is following during learning (exploration policy)

Observations (from 4 episodes): (s, a, s', r) tuples

1. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

2. E, up, C, -1
   C, right, A, -1
   A, exit, term, -10

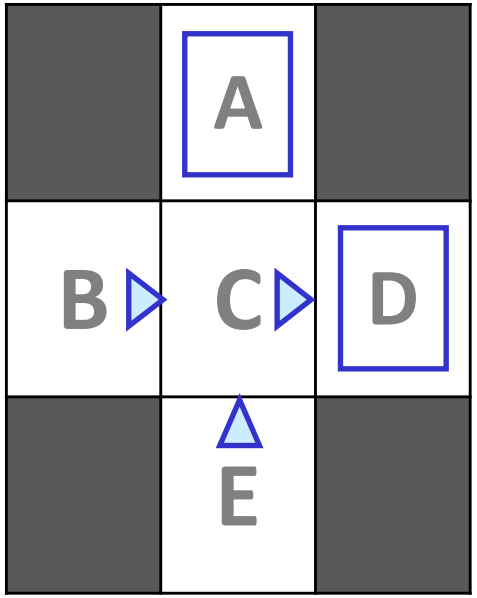
3. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

4. E, up, C, -1
   C, right, D, -1
   D, exit, term, 10

Estimates:
Estimate V(s) as the sample average of returns when starting from s

Example: Monte Carlo evaluation

States: A,B,C,D,E
Actions: left, right, up, down, exit



Blue arrows denote policy that agent is following during learning (exploration policy)

Observations (from 4 episodes): (s, a, s', r) tuples

1. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

2. E, up, C, -1
   C, right, A, -1
   A, exit, term, -10

3. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

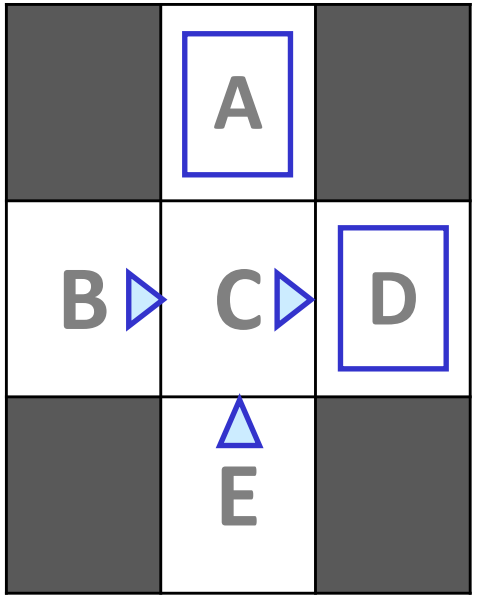
4. E, up, C, -1
   C, right, D, -1
   D, exit, term, 10

Estimates:
Estimate V(s) as the sample average of returns when starting from s

V(A) = [(-10)] / 1
= -10
V(B) = [(-1-1+10) + (-1-1+10)] / 2
= 8
V(C) = [(-1+10) + (-1-10)
+ (-1+10) + (-1+10)] / 4
= 4
V(D) = [10 + 10 + 10] / 3
= 10
V(E) = [(-1-1-10) + (-1-1+10)] / 2
= -2

Example: Monte Carlo evaluation

States: A,B,C,D,E
Actions: left, right, up, down, exit



Blue arrows denote policy that agent is following during learning (exploration policy)

Observations (from 4 episodes): (s, a, s', r) tuples

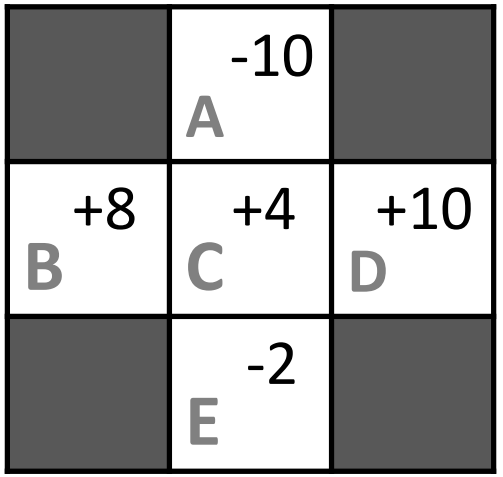
1. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

2. E, up, C, -1
   C, right, A, -1
   A, exit, term, -10

3. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

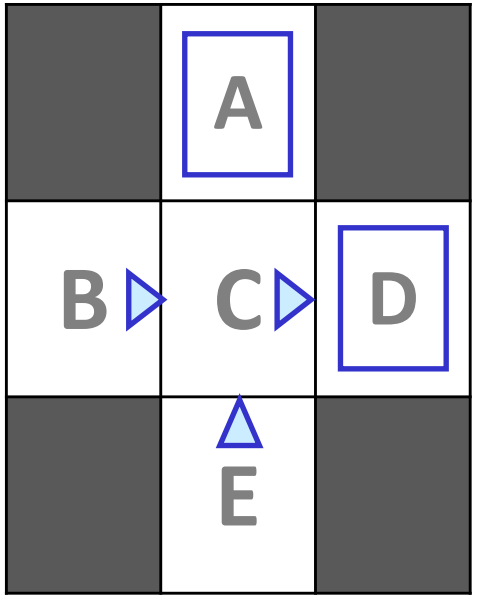
4. E, up, C, -1
   C, right, D, -1
   D, exit, term, 10

Estimates:
Estimate V(s) as the sample average of returns when starting from s



Example: Monte Carlo evaluation

States: A,B,C,D,E
Actions: left, right, up, down, exit



Blue arrows denote policy that agent is following during learning (exploration policy)

Observations (from 4 episodes): (s, a, s', r) tuples

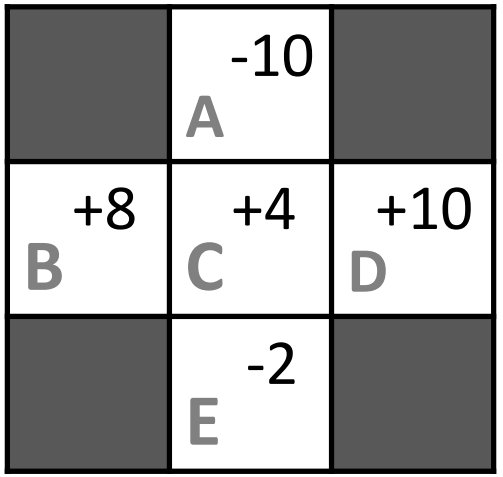
1. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

2. E, up, C, -1
   C, right, A, -1
   A, exit, term, -10

3. B, right, C, -1
   C, right, D, -1
   D, exit, term, 10

4. E, up, C, -1
   C, right, D, -1
   D, exit, term, 10

Estimates:
Estimate V(s) as the sample average of returns when starting from s



… but B and E are basically identical (both can only lead to C)

Monte Carlo Policy Evaluation ("prediction")

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

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

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

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

Monte Carlo Policy Evaluation ("prediction")

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

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

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

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

First-visit MC: Only consider return following first visit to s

Monte Carlo Policy Evaluation ("prediction")

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

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

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

~~Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:~~

Append G to $Returns(S_t)$

$V(S_t) \leftarrow$ average$(Returns(S_t))$

First-visit MC: Only consider return following first visit to s

Every-visit MC: Consider returns following every visit to s

Monte Carlo Policy Evaluation ("prediction")

First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

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

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

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

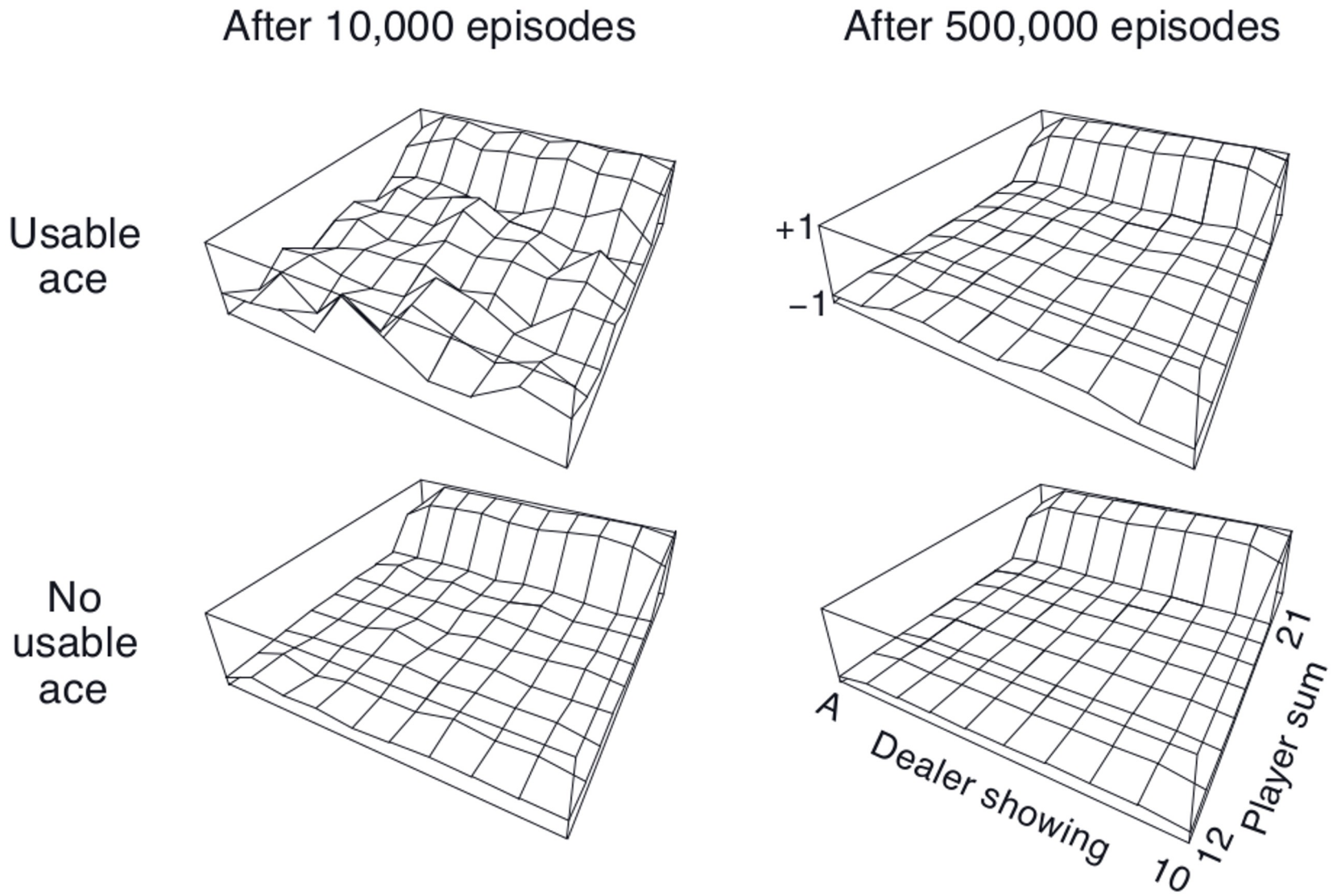
First-visit MC: Only consider return following first visit to s

- easier to analyze theoretically

Every-visit MC: Consider returns following every visit to s

- no special cases, easier to extend

Monte Carlo Policy Evaluation: Example



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

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

   Compute returns from the end:

				0
			z: 1 +	$\gamma * 0$
		x: 5 +	$\gamma * 1 +$	$\gamma^2 * 0$
	y: 2 +	$\gamma * 5 +$	$\gamma^2 * 1 +$	$\gamma^3 * 0$
x: -1 +	$\gamma * 2 +$	$\gamma^2 * 5 +$	$\gamma^3 * 1 +$	$\gamma^4 * 0$

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

   Compute returns from the end:

				0
			z: 1 +	$\gamma * 0$
		x: 5 +	$\gamma * 1 +$	$\gamma^2 * 0$
	y: 2 +	$\gamma * 5 +$	$\gamma^2 * 1 +$	$\gamma^3 * 0$
x: -1 +	$\gamma * 2 +$	$\gamma^2 * 5 +$	$\gamma^3 * 1 +$	$\gamma^4 * 0$

   Add returns to each state, recompute averages
   (or use incremental average)

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

   Compute returns from the end:

				0
(If first visit,			z: 1 +	$\gamma * 0$
do not add this)		x: 5 +	$\gamma * 1 +$	$\gamma^2 * 0$
	y: 2 +	$\gamma * 5 +$	$\gamma^2 * 1 +$	$\gamma^3 * 0$
x: -1 +	$\gamma * 2 +$	$\gamma^2 * 5 +$	$\gamma^3 * 1 +$	$\gamma^4 * 0$

   Add returns to each state, recompute averages
   (or use incremental average)

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

   Compute returns from the end:

				0
(If every visit,			z: 1 +	$\gamma * 0$
add everything)		x: 5 +	$\gamma * 1$ +	$\gamma^2 * 0$
	y: 2 +	$\gamma * 5$ +	$\gamma^2 * 1$ +	$\gamma^3 * 0$
x: -1 +	$\gamma * 2$ +	$\gamma^2 * 5$ +	$\gamma^3 * 1$ +	$\gamma^4 * 0$

   Add returns to each state, recompute averages
   (or use incremental average)

Example: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

   Assume we are doing every-visit.
   V(x) = (5.579 + 5.9) / 2 = 5.7395
   V(y) = 7.31
   V(z) = 1

Think-pair-share: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

   Assume we are doing every-visit.
   V(x) = (5.579 + 5.9) / 2 = 5.7395
   V(y) = 7.31
   V(z) = 1

2. Generate episode 2 following π: (s, a → r', s', a' → …)
   z, b → +2, y, c → +5, x, b → +1, [term]
   Compute returns, append returns, recompute averages

3. …

Think-pair-share: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

Input: Policy π to be evaluated

1. Generate episode 2 following π: (s, a → r', s', a' → …)
   z, b → +2, y, c → +5, x, b → +1, [term]

   Compute returns from the end:

				0
(If every visit,			x: 1 +	$\gamma * 0$
add everything)		y: 5 +	$\gamma * 1$ +	$\gamma^2 * 0$
	z: 2 +	$\gamma * 5$ +	$\gamma^2 * 1$ +	$\gamma^3 * 0$

   Add returns to each state, recompute averages
   (or use incremental average)

Think-pair-share: 3 states, 2 actions

States: x, y, z Actions: b, c $\gamma = 0.9$

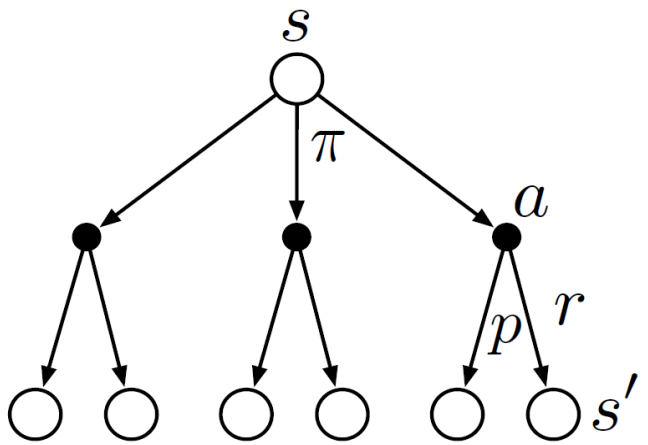
Input: Policy π to be evaluated

1. Generate episode 1 following π: (s, a → r', s', a' → …)
   x, b → -1, y, b → +2, x, c → +5, z, b → +1, [term]

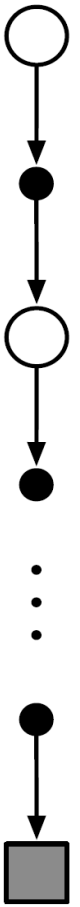
   Assume we are doing every-visit.
   V(x) = (5.579 + 5.9+1) / 3 = 4.15966
   V(y) = (7.31+5.9)/2=6.605
   V(z) = (1+7.31)/2=4.155

Monte Carlo backup diagram

$$V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \big[r + \gamma V(s') \big]$$



Backup diagram for v_π

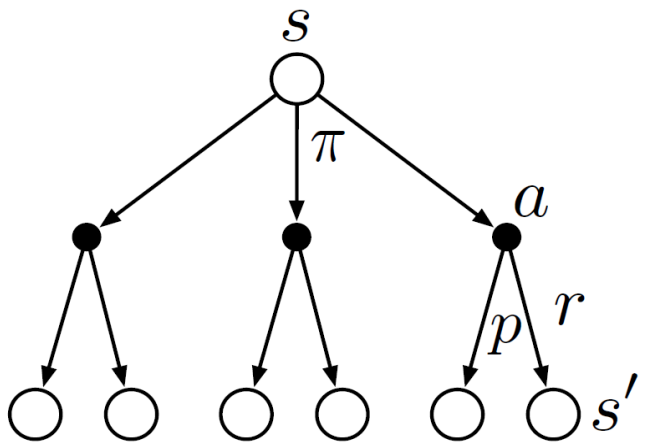


Dynamic programming vs. Monte-Carlo

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Monte Carlo backup diagram

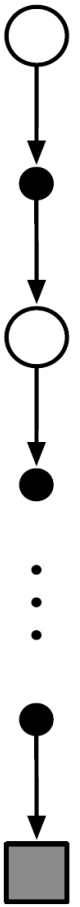
$$V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \big[r + \gamma V(s') \big]$$



Backup diagram for v_π

DP: One-step transitions

MC: Roll out to end of episode

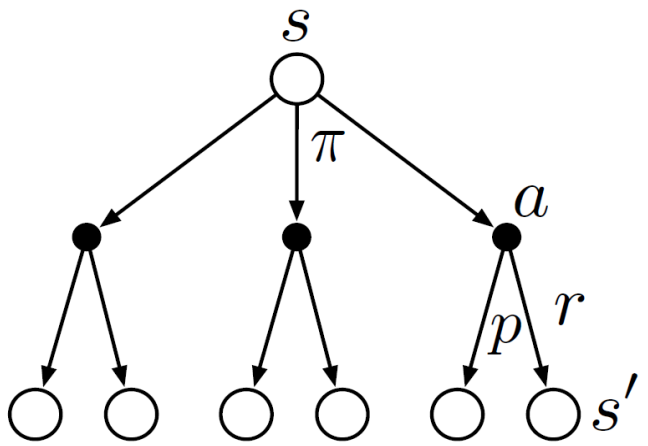


Dynamic programming vs. Monte-Carlo

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Monte Carlo backup diagram

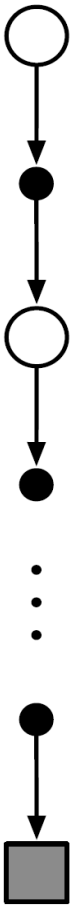
$$V(s) \leftarrow \sum_{s',r} p(s', r \,|\, s, \pi(s)) \left[r + \gamma V(s') \right]$$



Backup diagram for v_π

DP: Bootstrap
(use V(s'))

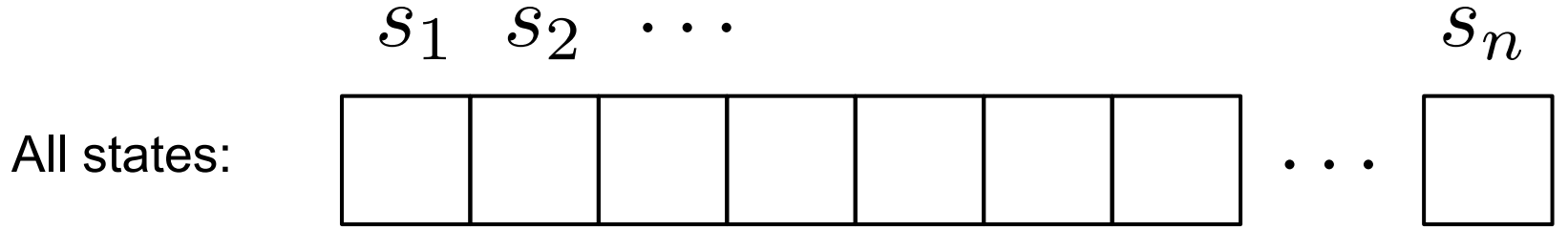
MC: No
bootstrap



Dynamic programming vs. Monte-Carlo

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Monte Carlo Policy Evaluation: General idea



Rollouts

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

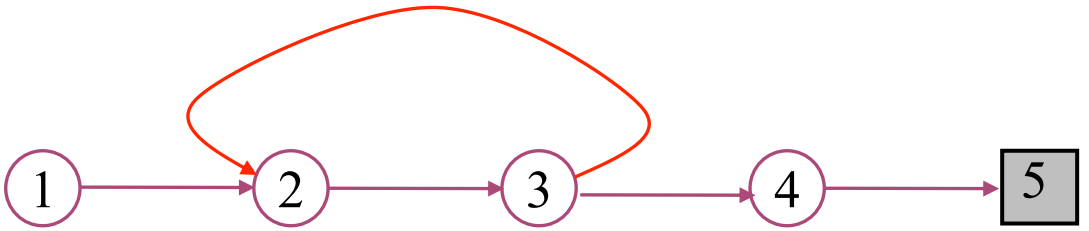
– the value of that state is the average discounted reward over all rollouts

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

- *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

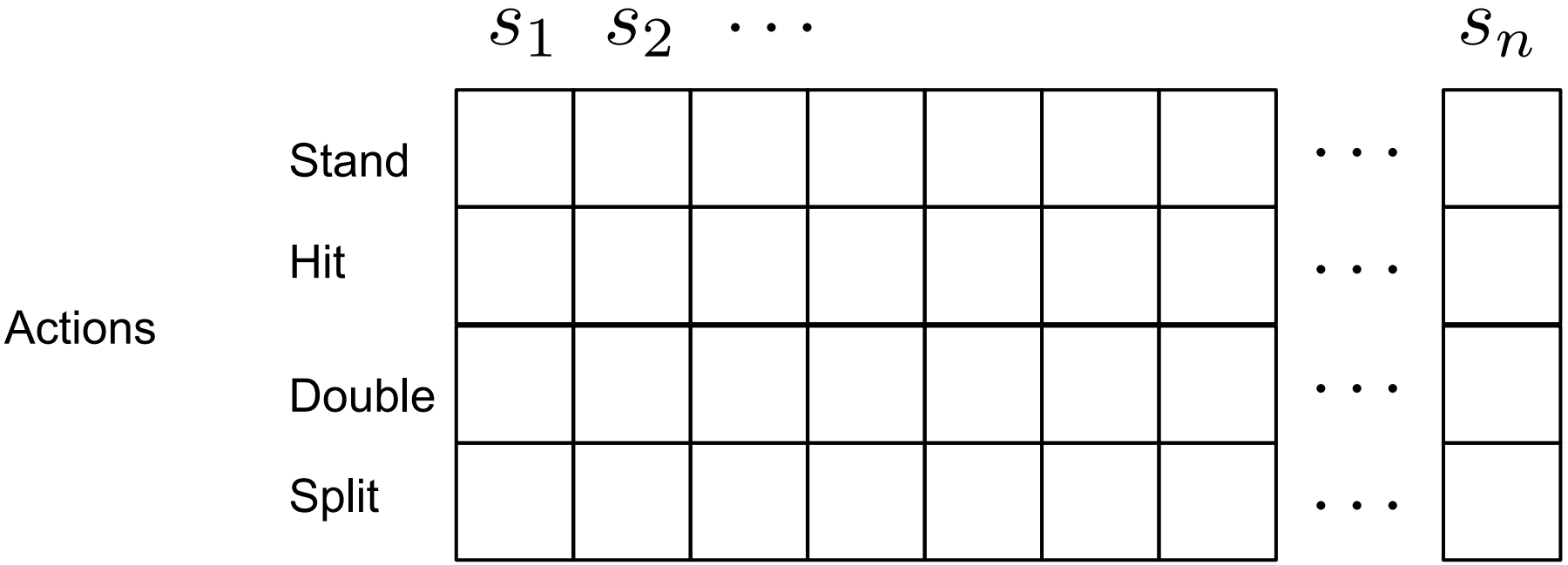
Two different types of policy evaluation:

State-value fn

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

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

Action-value fn



Do rollouts from each state/action pair

Monte Carlo Action-Value Policy Evaluation

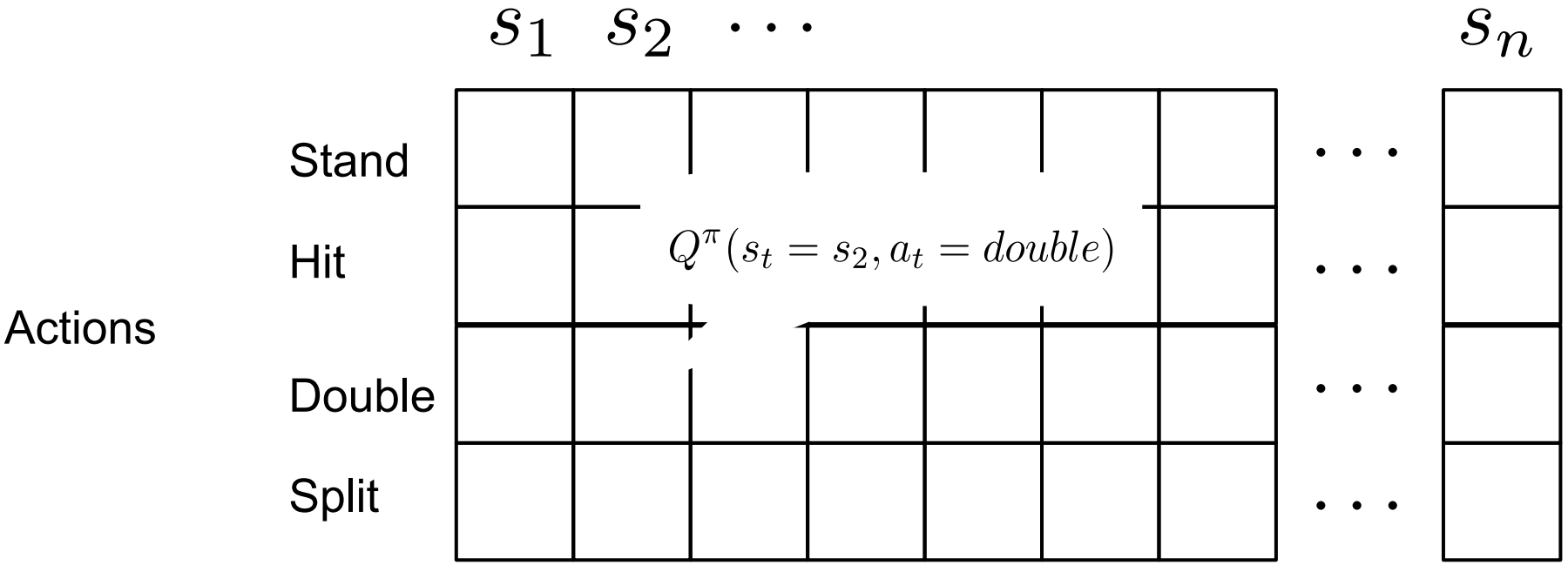
Two different types of policy evaluation:

State-value fn

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

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

Action-value fn

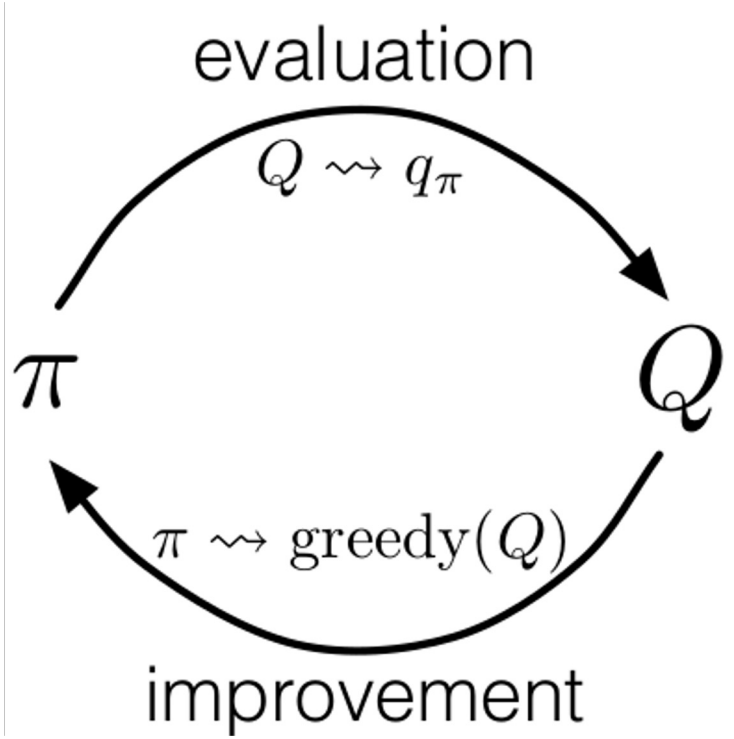


Do rollouts from each state/action pair

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

(these will converge to true q in the limit)

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

Key idea: evaluate/improve policy iteratively...

Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

First-visit version

Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

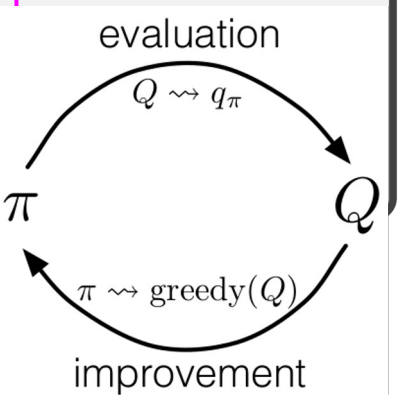
$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$



Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

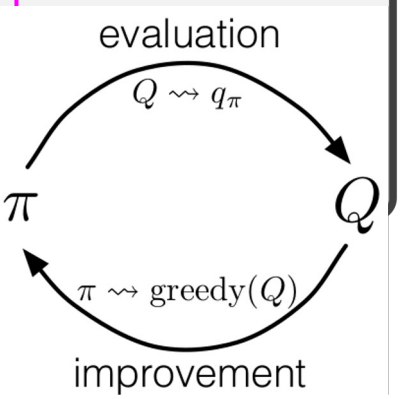
$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$



Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

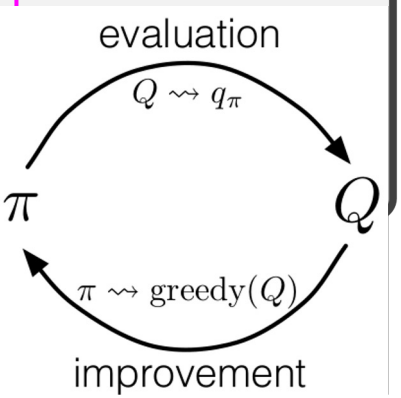
$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$



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

Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

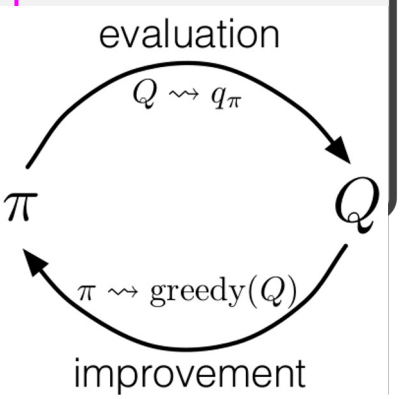
$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$



How would you make a policy iteration version of this?

Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbi

$Returns(s,a) \leftarrow$

Loop forever (for ea

Choose $S_0 \in \mathcal{S}$,

probability > 0

Generate an epis

S_{T-1}, A_{T-1}, R_T

$G \leftarrow 0$

Loop for each step

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in S_0, A $_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

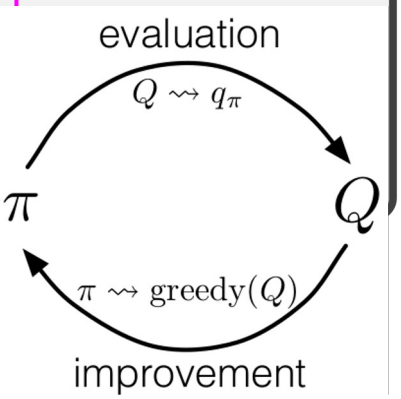
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

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 ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Loop forever (for each epi

Choose $S_0 \in \mathcal{S}$, $A_0 \in$ ability > 0

Generate an episode A_{T-1}, R_T

$G \leftarrow 0$

Loop for each step of episode,

$G \leftarrow \gamma G + R_{t+1}$

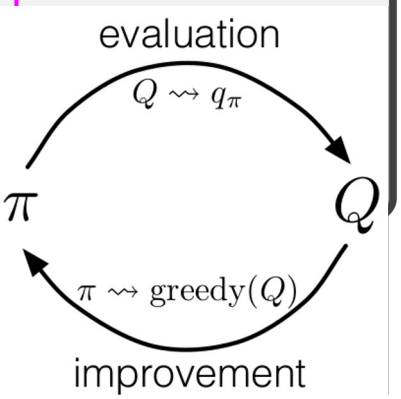
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

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



Monte Carlo Control

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in$

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in$

$Returns(s,a) \leftarrow$ empty list, for all

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

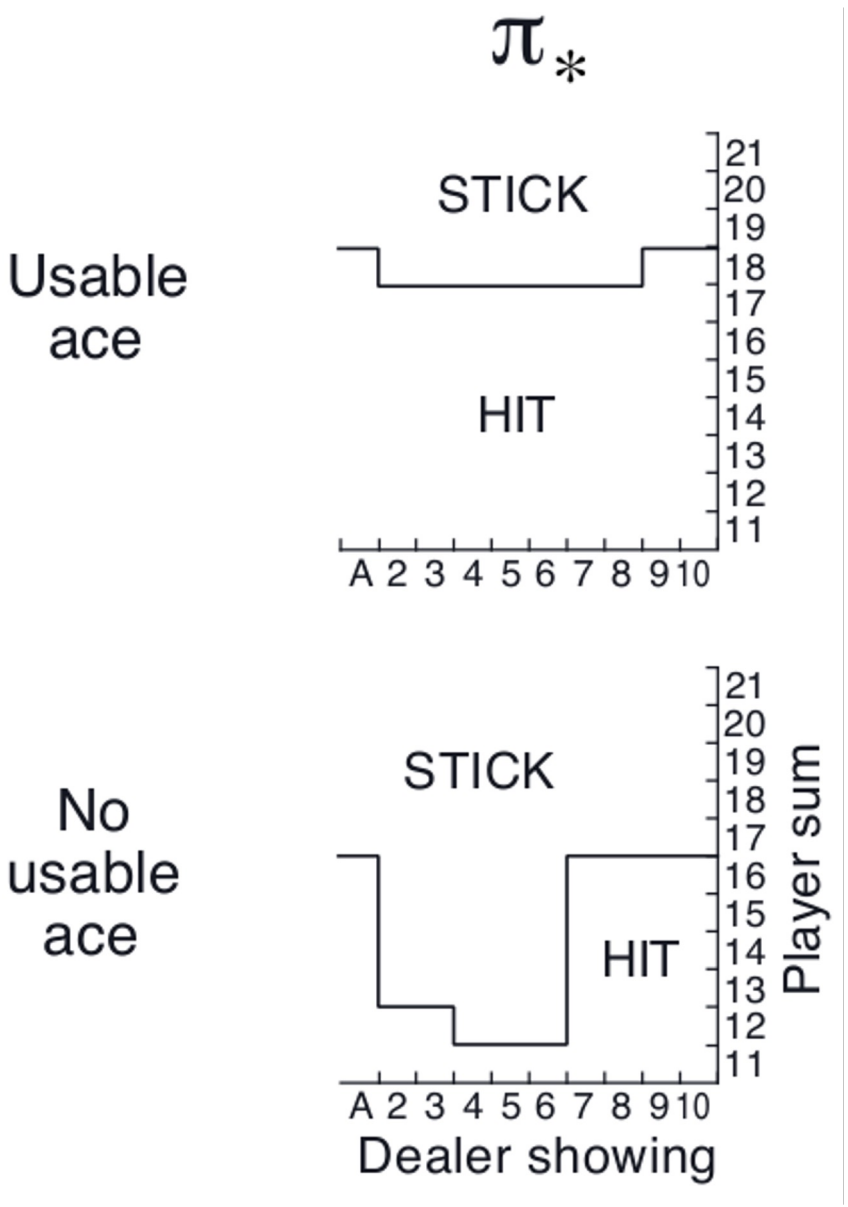
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

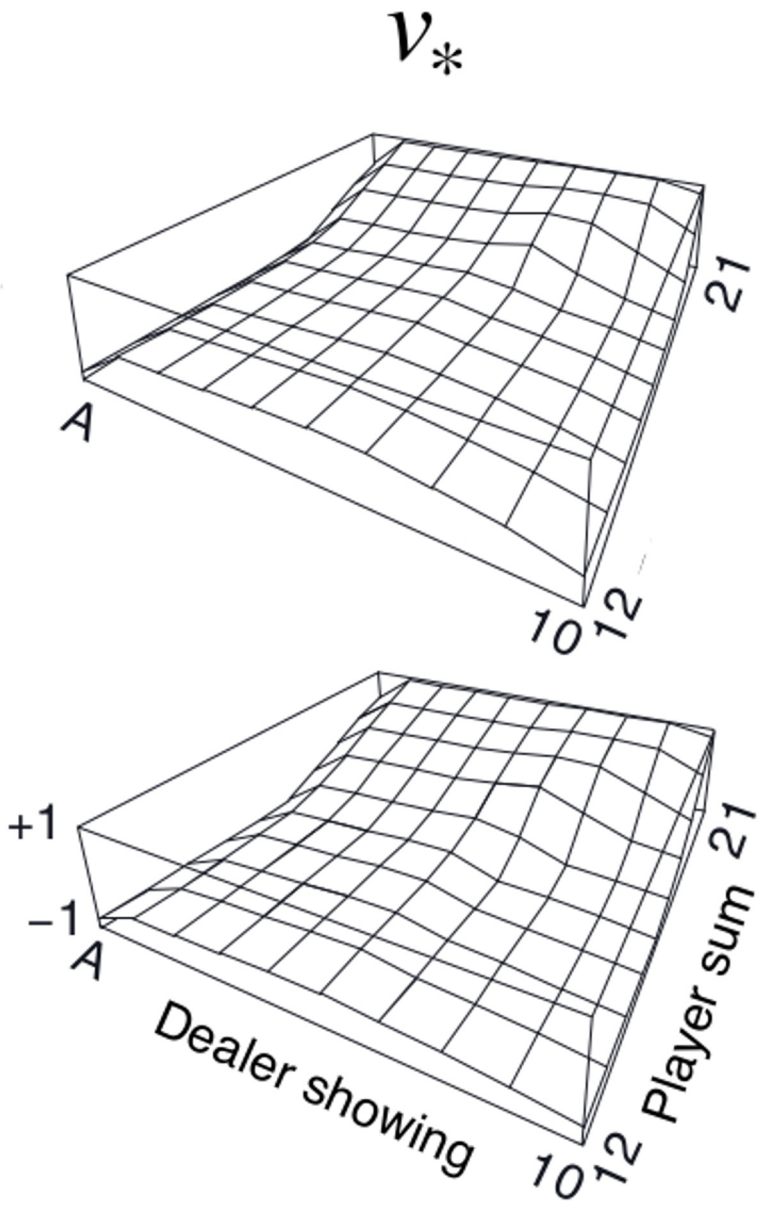
Exploring starts:

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

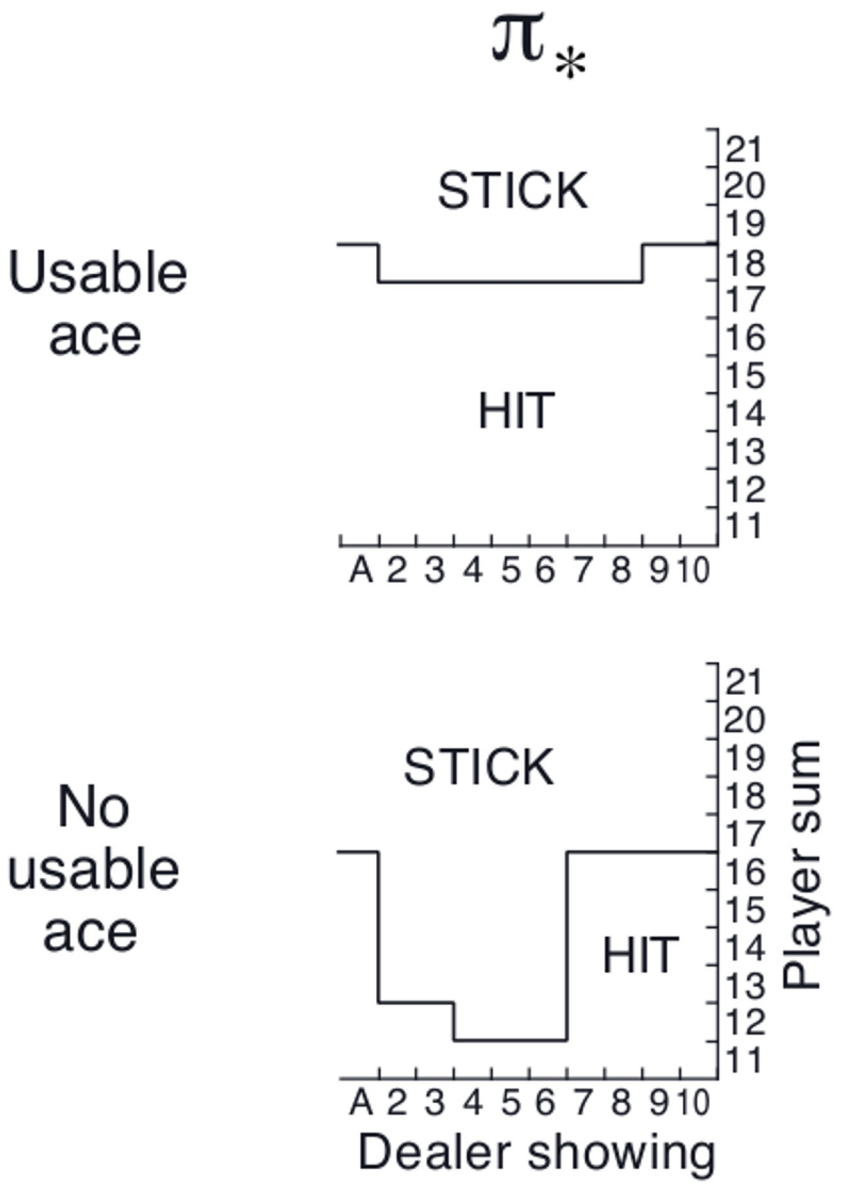
Why do you need exploring starts?

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 (as in MCES)

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] of episodes for MC
- To
  - 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 .

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 (as in MCES)

(not formally proven –
"one of the most fundamental open theoretical questions in reinforcement learning.")

Exploration in MC control

So far, we have avoided talked about exploration during RL

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0
Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
Append G to $Returns(S_t, A_t)$
$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

Exploration in MC control

So far, we have avoided talked about exploration during RL

Monte Carlo ES (Exploring Start

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all s

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in$

$Returns(s,a) \leftarrow$ empty list, for all

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability > 0

Generate an episode from S_0, A_0, following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

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?

Exploration in MC control

So far, we have avoided talked about exploration during RL

Monte Carlo ES (Exploring Start

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all s

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in$

$Returns(s,a) \leftarrow$ empty list, for all

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all ... have probability > 0

Generate an episode from S_0, A_0, following π: ...

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T$

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

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?

Exploration in MC control

So far, we have avoided talked about exploration during RL

Monte Carlo ES (Exploring Start

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all s

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in$

$Returns(s,a) \leftarrow$ empty list, for all

Loop forever (for each episode):

Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all ... have probability > 0

Generate an episode from S_0, A_0, following π: ...

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T$

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1, \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow$ average$(Returns(S_t$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

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

Remember: Multi-armed bandit

Given estimates $Q_t(a)$ for all actions, what action to take?

Two natural possibilities:

1. Greedy action selection: $A_t \doteq \arg\max_a Q_t(a)$
   Exploitation

2. Do something else
   Exploration

How to decide between these two?
One simple possibility: ε-greedy
1. With probability 1-ε, take greedy action (exploit)
2. With probability ε, take an action uniformly at random
   (explore; might pick greedy action too)

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

MC control with ε-greedy policies

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

E-greedy exploration

MC control with ε-greedy policies

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Converges to an optimal ε-soft policy (see Sec. 5.4)

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

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

On-policy vs. off-policy

On-policy first-visit MC control (for ε-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

On-policy vs. off-policy

On-policy: Learn about the policy currently executing

- If behavior policy is suboptimal, then Q is suboptimal
- But if behavior policy does not explore, cannot be optimal!

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

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

On-policy vs. off-policy

On-policy: Learn about the policy currently executing

- If behavior policy is suboptimal, then Q is suboptimal
- But if behavior policy does not explore, cannot be optimal!

$$A^* \leftarrow \arg\max_a Q(S_t, a) \qquad \text{(with ties broken arbitrarily)}$$
$$\text{For all } a \in \mathcal{A}(S_t):$$
$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy *b*

On-policy vs. off-policy

On-policy: Learn about the policy currently executing

- If behavior policy is suboptimal, then Q is suboptimal
- But if behavior policy does not explore, cannot be optimal!

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)
For all $a \in \mathcal{A}(S_t)$:

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

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy *b*

- Separate exploration from estimation (of optimal values)
- Useful if we can be exploratory during "training",
  but need to act optimally when it counts (at "test")

On-policy vs. off-policy

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy b

- Useful if we can be exploratory during "training",
but need to act optimally when it counts (at "test")

Typical setting:
Behavior policy: b = ε-greedy
Target policy: π = greedy

Off-policy methods

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy b

- Useful if we can be exploratory during "training",
but need to act optimally when it counts (at "test")

Typical setting:
Behavior policy: b = ε-greedy
Target policy: π = greedy

Necessary: Coverage: If $\pi(a|s) > 0$, then $b(a|s) > 0$

Off-policy methods

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy b

- Useful if we can be exploratory during "training",
but need to act optimally when it counts (at "test")

Typical setting:
Behavior policy: b = ε-greedy
Target policy: π = greedy

Necessary: Coverage: If $\pi(a|s) > 0$, then $b(a|s) > 0$
- Anything the target policy may do,
the behavior policy must try as well
- Satisfied for b = ε-greedy and π = greedy

Off-policy methods

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy b

- Useful if we can be exploratory during "training",
but need to act optimally when it counts (at "test")

Typical setting:
Behavior policy: b = ε-greedy
Target policy: π = greedy

Necessary: Coverage: If $\pi(a|s) > 0$, then $b(a|s) > 0$
- Anything the target policy may do,
the behavior policy must try as well
- Satisfied for b = ε-greedy and π = greedy

Ideal: b and π are not too different (experience is relevant)

Importance sampling

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy *b*

- Using experiences from *b*, we know how to
estimate Q_b using Monte-Carlo methods

$$Q_b(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} [G_t | S_t = s, A_t = a]$$

Importance sampling

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy *b*

- Using experiences from *b*, we know how to
estimate Q_b using Monte-Carlo methods

$$Q_b(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} [G_t | S_t = s, A_t = a]$$

- How to estimate Q_π using Monte-Carlo methods?

Importance sampling

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy *b*

- Using experiences from *b*, we know how to
estimate Q_b using Monte-Carlo methods

$$Q_b(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} [G_t | S_t = s, A_t = a]$$

- How to estimate Q_π using Monte-Carlo methods?

More general question:
Suppose x ~ p(x) , interested in $\mathbb{E}_p[f(X)]$

Monte-Carlo estimate: $\mathbb{E}_p[f(X)] = \sum_x p(x) f(x) \approx \frac{1}{N} \sum_i f(x^{(i)})$

How can we use the data { $x^{(i)}$ } to estimate $\mathbb{E}_q[f(X)]$?

Importance sampling

More general question:

Suppose x ~ p(x) , interested in $\mathbb{E}_p[f(X)]$

Monte-Carlo estimate: $\mathbb{E}_p[f(X)] = \sum_x p(x)f(x) \approx \frac{1}{N}\sum_i f(x^{(i)})$

How can we use the data { $x^{(i)}$ } to estimate $\mathbb{E}_q[f(X)]$?

$$\mathbb{E}_p[f(X)] = \sum_x p(x)f(x) \approx \frac{1}{N}\sum_i f(x^{(i)})$$

Importance sampling

More general question:

Suppose x ~ p(x) , interested in $\mathbb{E}_p[f(X)]$

Monte-Carlo estimate: $\mathbb{E}_p[f(X)] = \sum_x p(x)f(x) \approx \frac{1}{N}\sum_i f(x^{(i)})$

How can we use the data { $x^{(i)}$ } to estimate $\mathbb{E}_q[f(X)]$?

$$\mathbb{E}_p[f(X)] = \sum_x p(x)f(x) \approx \frac{1}{N}\sum_i f(x^{(i)})$$

$$\mathbb{E}_q[f(X)] = \sum_x q(x)f(x) = \sum_x p(x)\frac{q(x)}{p(x)}f(x) \approx \frac{1}{N}\sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})}f(x^{(i)})\right]$$

Importance sampling

More general question:

Suppose x ~ p(x) , interested in $\mathbb{E}_p[f(X)]$

Monte-Carlo estimate: $\mathbb{E}_p[f(X)] = \sum_x p(x)f(x) \approx \frac{1}{N}\sum_i f(x^{(i)})$

How can we use the data { $x^{(i)}$ } to estimate $\mathbb{E}_q[f(X)]$?

$$\mathbb{E}_p[f(X)] = \sum_x p(x)f(x) \approx \frac{1}{N}\sum_i f(x^{(i)})$$

$$\mathbb{E}_q[f(X)] = \sum_x q(x)f(x) = \sum_x p(x)\frac{q(x)}{p(x)}f(x) \approx \frac{1}{N}\sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})}f(x^{(i)})\right]$$

- p is where the data actually came from (e.g., behavior)
- q is what we want to estimate (e.g., target)

q(x) / p(x) = importance sampling ratio

Importance sampling

Off-policy: Learn the value of some target policy π
while executing / using experience from behavior policy *b*

- Using experiences from *b*, we know how to
estimate Q_b using Monte-Carlo methods

$$Q_b(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} [G_t | S_t = s, A_t = a]$$

- How to estimate Q_π using Monte-Carlo methods?

$$\frac{1}{N} \sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})} f(x^{(i)}) \right]$$

Here: f = return of the sampled trajectory (G_t)
p = probability of trajectory under behavior policy *b*
q = probability of trajectory under target policy π

Importance sampling: Probability of trajectory

How to estimate Q_π using Monte-Carlo methods?

$$\frac{1}{N}\sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})} f(x^{(i)})\right]$$

f = return of the sampled trajectory (G_t)
p = probability of trajectory under behavior policy *b*
q = probability of trajectory under target policy π

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: Importance-sampling ratio

How to estimate Q_π using Monte-Carlo methods?

$$\frac{1}{N}\sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})} f(x^{(i)})\right]$$

f = return of the sampled trajectory (G_t)
p = probability of trajectory under behavior policy *b*
q = probability of trajectory under target policy π

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

where p here is the state-transition probability function defined by (3.4). Thus, the relative probability of the trajectory under the target and behavior policies (the importance-sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k,A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k,A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}. \tag{5.3}$$

Importance sampling: Importance-sampling ratio

How to estimate Q_π using Monte-Carlo methods?

$$\frac{1}{N}\sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})} f(x^{(i)})\right]$$

f = return of the sampled trajectory (G_t)
p = probability of trajectory under behavior policy *b*
q = probability of trajectory under target policy π

where p here is the state-transition probability function defined by (3.4). Thus, the relative probability of the trajectory under the target and behavior policies (the importance-sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)\cancel{p(S_{k+1}|S_k,A_k)}}{\prod_{k=t}^{T-1} b(A_k|S_k)\cancel{p(S_{k+1}|S_k,A_k)}} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}. \tag{5.3}$$

Importance sampling

How to estimate Q_π using Monte-Carlo methods?

$$\frac{1}{N}\sum_i \left[\frac{q(x^{(i)})}{p(x^{(i)})} f(x^{(i)})\right]$$

f = return of the sampled trajectory (G_t)
p = probability of trajectory under behavior policy *b*
q = probability of trajectory under target policy π

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

$$\mathbb{E}[G_t|S_t=s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1}G_t \mid S_t=s] = v_\pi(s)$$

Ordinary vs weighted importance sampling

$$\mathbb{E}[G_t | S_t = s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

Ordinary (simple) average of importance-weighted returns:

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

Ordinary vs weighted importance sampling

$$\mathbb{E}[G_t | S_t = s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

Ordinary (simple) average of importance-weighted returns:

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

← number of samples
(trajectories from s in either
first-visit or every-visit sense)

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

Ordinary vs weighted importance sampling

$$\mathbb{E}[G_t | S_t = s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

Ordinary (simple) average of importance-weighted returns:

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

← number of samples (trajectories from s in either first-visit or every-visit sense)

Can be very high variance since probability of the trajectory under behavior policy *b* can be arbitrarily different from the probability of trajectory under target policy π

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

Ordinary vs weighted importance sampling

$$\mathbb{E}[G_t | S_t = s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

Ordinary (simple) average of importance-weighted returns:

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

← number of samples (trajectories from s in either first-visit or every-visit sense)

Weighted average of importance-weighted returns:

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

← sum of weights

Ordinary vs weighted importance sampling

$$\mathbb{E}[G_t | S_t = s] = v_b(s)$$

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

Ordinary (simple) average of importance-weighted returns:

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

← number of samples (trajectories from s in either first-visit or every-visit sense)

Weighted average of importance-weighted returns:

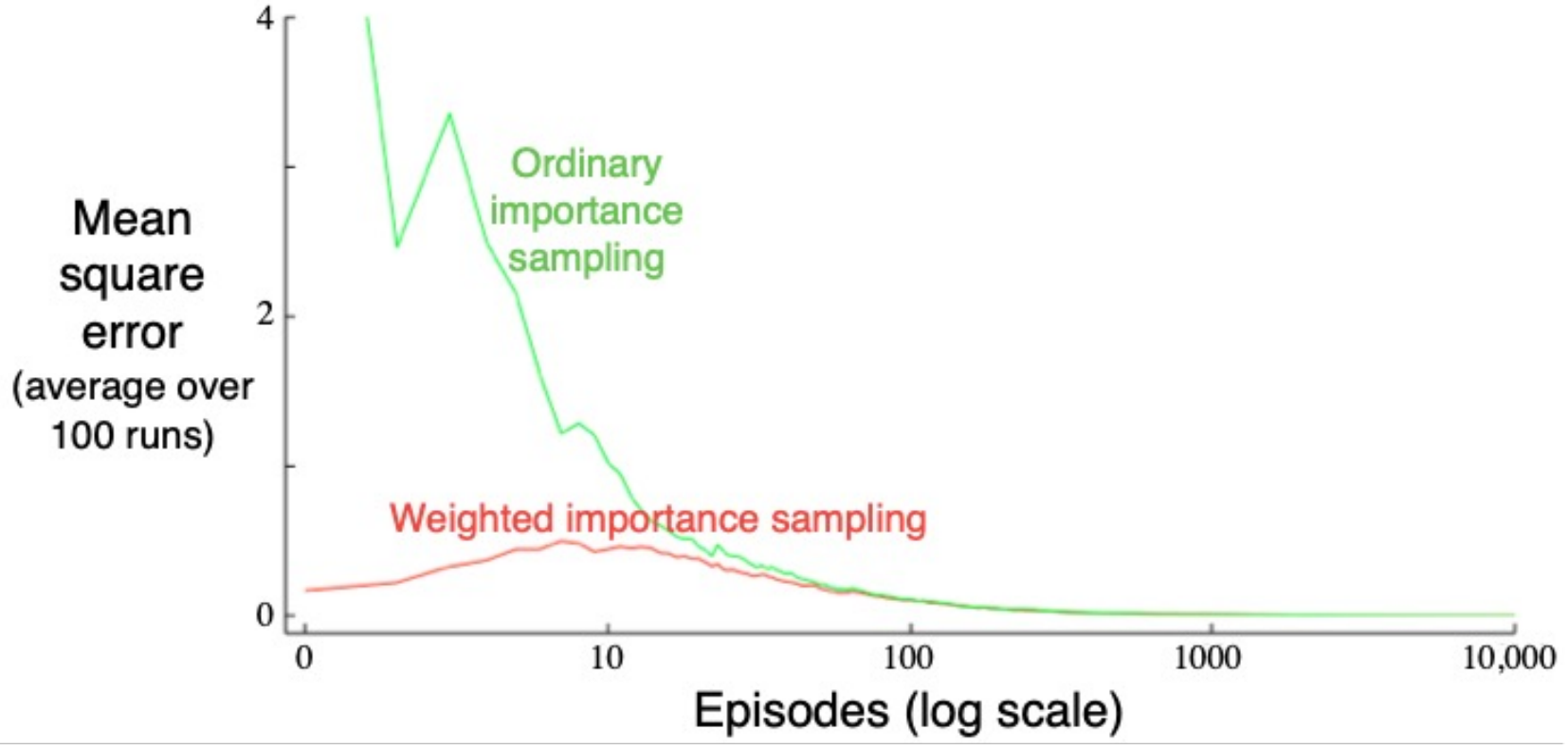
$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$

← sum of weights

As a result, this has bias, but lower variance

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



Importance sampling

- What can we do with the importance sampling ratio?

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

Remember this?

A simple bandit algorithm

Initialize, for $a = 1$ to k:

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

Repeat forever:

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1-\varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$

Incremental estimate of $Q(A) = \frac{1}{n}\sum_{i=1}^{n} R_i$

Incremental update in ordinary IS

Ordinary (simple) average of importance-weighted returns:

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

Incremental update in ordinary IS

We want to estimate this:

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

Actually, we can use our previous Q-value:

$$Q_{n+1} = Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]$$

Incremental update in ordinary IS

We want to estimate this:

$$Q_t(a) \doteq \frac{\text{sum of rewards when } a \text{ taken prior to } t}{\text{number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot \mathbb{1}_{A_i=a}}{\sum_{i=1}^{t-1} \mathbb{1}_{A_i=a}}$$

Actually, we can use our previous Q-value:

$$Q_{n+1} = Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]$$

Apply exactly the same trick to ordinary IS:

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

$$V_{n+1} = V_n + (1/n)[W_n G_n - V_n]$$

Incremental update in weighted IS

Non-incremental weighted IS estimate:

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \qquad n \geq 2$$

Incremental update in weighted IS

Non-incremental weighted IS estimate:

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \qquad n \geq 2$$

Incremental version (cumulative weight update):

$$C_0 \doteq 0$$

$$C_{n+1} \doteq C_n + W_{n+1}$$

Incremental update in weighted IS

Non-incremental weighted IS estimate:

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \qquad n \geq 2$$

Incremental version (V_1 can be arbitrary):

$$C_0 \doteq 0$$

$$C_{n+1} \doteq C_n + W_{n+1}$$

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}\Big[G_n - V_n\Big], \qquad n \geq 1$$

Incremental, weighted version

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$

Keep a running sum of weights

Loop forever (for each episode):
$b \leftarrow$ any policy with coverage of π
Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Incremental, weighted version

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, while $W \neq 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
        $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Incremental, weighted version

Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy π
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

Loop forever (for each episode):
    $b \leftarrow$ any policy with coverage of π
    Generate an episode following b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$, 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

- Combine importance sampling for estimation with policy improvement
- Another MC learning method
- Use soft behavior policy to learn deterministic optimal policy
- Converges in the limit

Off-policy Monte-Carlo control

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

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \in \mathbb{R}$ (arbitrarily)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):
$b \leftarrow$ any soft policy
Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
$G \leftarrow 0$
$W \leftarrow 1$
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Deterministic policy, so prob is either 0 or 1

Off-policy Monte-Carlo control

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

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \in \mathbb{R}$ (arbitrarily)

$C(s,a) \leftarrow 0$

$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):

$b \leftarrow$ any soft policy

Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

$W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)

$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Off-policy Monte-Carlo control

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

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \in \mathbb{R}$ (arbitrarily)

$C(s,a) \leftarrow 0$

$\pi(s) \leftarrow \arg\max_a Q(s,a)$ (with ties broken consistently)

Loop forever (for each episode):

$b \leftarrow$ any soft policy

Generate an episode using b: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

$W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken consistently)

If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)

$W \leftarrow W \frac{1}{b(A_t|S_t)}$

Deterministic policy, so prob is either 0 or 1

MC Summary

- MC methods estimate value function by doing rollouts
- Can estimate either the action value function, $Q(s, a)$, or the state 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
- Off-policy methods typically have higher-variance but can be more sample-efficient (reuse samples)
- Off-policy versions of other RL methods also typically use some form of importance sampling