Chapter 5 - Monte Carlo Methods!

Jan 1, 2022

Monte Carlo methods are ways of solving the RL problem (i.e. agent must learn best policy to maximize reward) based on averaging sample returns. Monte Carlo methods require only sample sequences of states, actions, and rewards from actual or simulated interaction w/ an environment.

↳ An agent can learn from actual or simulated experience, which requires no prior knowledge of the environment's dynamics, but can still achieve optimal behavior.

↳ The term "Monte Carlo" refers to any estimation method whose operation involves a significant random component.

- Here, Monte Carlo methods are defined only for episodic tasks.
- Instead of computing value functions, we learn them.

5.1 - Monte Carlo Prediction:

Monte Carlo approach for learning the state-value function for a given policy.

Goal: Estimate $v_\pi(s)$ given a set of episodes following π and passing through s.

↳ Each occurrence of state s in an episode is called a visit to s.

❖ s may be visited multiple times in the same episode!

↳ The first visit to s in an episode is called the first visit.

First-Visit MC Prediction:

Estimates $v_\pi(s)$ as the average of returns following first visits to s.

↳ Since s can be visited multiple times in an episode, we only take the average of returns after the first visit.

Every-Visit MC Method:

Estimates $v_\pi(s)$ as the average of returns following all visits to s.

Pseudo-Code!

Algorithm 1: First-Visit MC Prediction

Input: policy π, positive integer *num_episodes*
Output: value function V ($\approx v_\pi$, if *num_episodes* is large enough)
Initialize $N(s) = 0$ for all $s \in \mathcal{S}$
Initialize Returns$(s) = 0$ for all $s \in \mathcal{S}$
for *episode* $e \leftarrow 1$ ***to*** $e \leftarrow num\_episodes$ **do**
  Generate, using π, an episode $S_0, A_0, R_1, S_1, A_1, R_2 \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for *time step* $t = T-1$ ***to*** $t = 0$ *(of the episode e)* **do**
    $G \leftarrow G + R_{t+1}$
    if *state* S_t ***is not*** *in the sequence* $S_0, S_1, \ldots, S_{t-1}$ **then**
      Returns$(S_t) \leftarrow$ Returns$(S_t) + G_t$
      $N(S_t) \leftarrow N(S_t) + 1$
  end
end
$V(s) \leftarrow \frac{\text{Returns}(s)}{N(s)}$ for all $s \in \mathcal{S}$
return V

↳ Only update Returns(S_t) the first time S_t is encountered in the episode

Algorithm 2: Every-Visit MC Prediction

Input: policy π, positive integer *num_episodes*
Output: value function V ($\approx v_\pi$, if *num_episodes* is large enough)
Initialize $N(s) = 0$ for all $s \in \mathcal{S}$
Initialize Returns$(s) = 0$ for all $s \in \mathcal{S}$
for *episode* $e \leftarrow 1$ ***to*** $e \leftarrow num\_episodes$ **do**
  Generate, using π, an episode $S_0, A_0, R_1, S_1, A_1, R_2 \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for *time step* $t = T-1$ ***to*** $t = 0$ *(of the episode e)* **do**
    $G \leftarrow G + R_{t+1}$
    Returns$(S_t) \leftarrow$ Returns$(S_t) + G_t$
    $N(S_t) \leftarrow N(S_t) + 1$
  end
end
$V(s) \leftarrow \frac{\text{Returns}(s)}{N(s)}$ for all $s \in \mathcal{S}$
return V

*Red highlights the difference

Blackjack Example:

Undiscounted MDP where each game = one episode.

Reward: -1 for loss, 0 for draw, +1 for win

Actions: Hit, Stand

States: 200 total

State Variables:

1. Whether player has useable ace (Y/N)
2. Sum of player's cards (12-21)
3. Card the dealer shows (Ace-10)

※ Cards are dealt w/ replacement to satisfy Markov property (and so no counting cards).

Policy: Stops hitting when sum is 20 or 21.

S (Usable Ace, Sum, Dealer)	Returns(S)	V(S)
A	Returns(A) = [1]	1
B	Returns(B) = [1]	1

A = (NoAce, 20, 10)

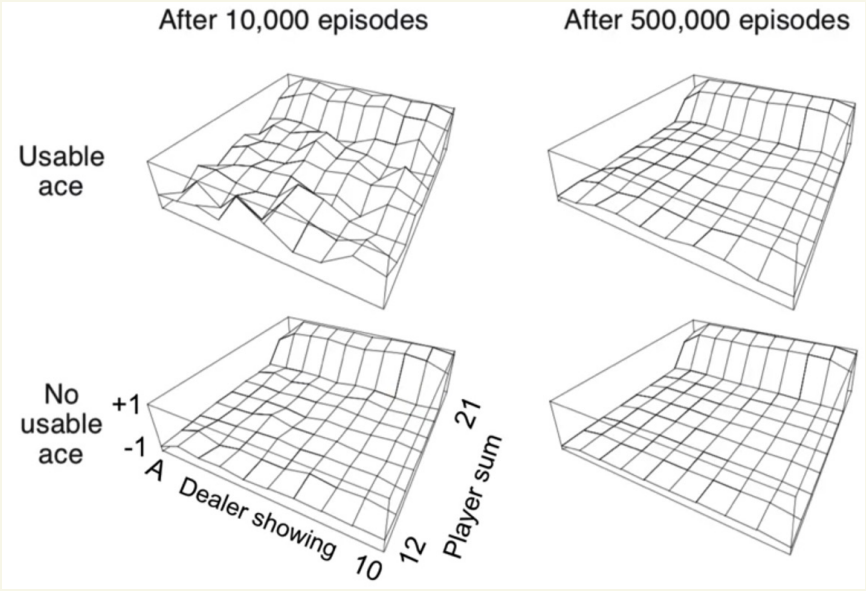
B = (NoAce, 13, 10)

※ All updates for this episode

Approximate State-Value Functions:

rougher cuz less examples

+1 for 20/21 cuz stand
↳likely to bust when 19



Implications of Monte Carlo Learning:

- no need to keep a large model of the environment (cuz learns from experience)
- the value of a state can be estimated <u>independently</u> of the other states
  ↳not like in DP
- the computation needed to update the value of each state doesn't depend on the size of the MDP
  ↳only the length of the episode and # states

5.2 - Monte Carlo Estimation of Action Values!

Without a model, we need to estimate action values instead of state values to determine a policy.

↳ Goal is to estimate q_* w/ Monte Carlo methods.

↳ A state-action pair s, a is visited if state s is visited and action a is taken in it.

⊗ The problem w/ then using every/first-visit MC methods is that many state-action pairs may not be visited.

↳ If π is deterministic, following π will only observe returns for one action per state.

↳ We need to estimate the value of all the actions from each state to compare alternatives.

↳ This is the problem of maintaining exploration.

Exploring Starts!

One way to solve this problem is to make episodes start in a state-action pair, w/ each pair having a nonzero probability of being chosen.

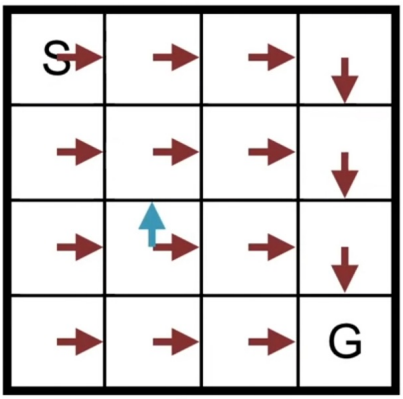
↳ As #episodes $\to \infty$, all state-action pairs will be visited.

Exploring Starts

$s_0, a_0, s_1, a_1, s_2, a_2, \ldots$

Random From π and p

※Otherwise policy would only follow the red arrows.



5.3 - Monte Carlo Control!

Now to use Monte Carlo methods to approximate optimal policies.

↳ Same pattern outlined in DP chapter.

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

E: Evaluation

I: Improvement

π_0: Random policy

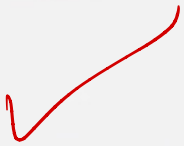
※ Since we use action-value functions, we don't need the model to construct the greedy policy.

↳ ~~$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s', r | s, a)[r + \gamma V(s')]$~~

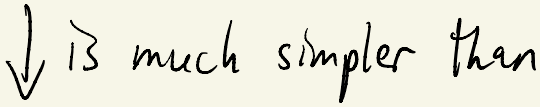
$\pi(s) \equiv \underset{a}{\operatorname{argmax}}\, q(s, a)$

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



Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
     $\Delta \leftarrow 0$
     Loop for each $s \in \mathcal{S}$:
       $v \leftarrow V(s)$
       $V(s) \leftarrow \sum_{s',r} p(s', r | s, \pi(s)) \big[r + \gamma V(s')\big]$
       $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
     old-action $\leftarrow \pi(s)$
     $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r | s, a) \big[r + \gamma V(s')\big]$
     If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Blackjack Example w/ Action Values!

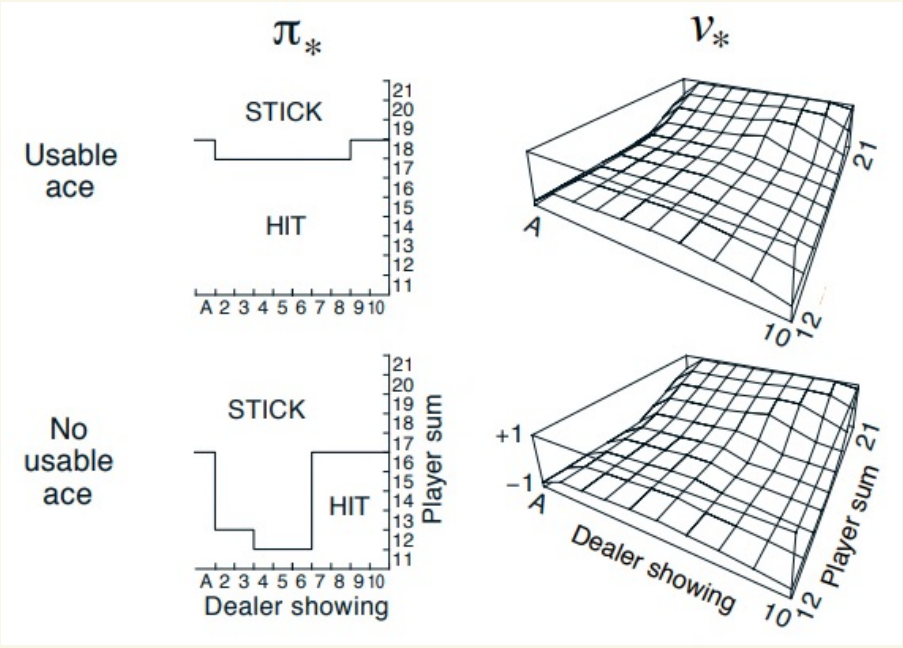
S,A	Returns(S,A)	Q(S,A)		π(S)
		Hit	Stick	
X,Stick	Returns(X,Stick) = [1]	0	1	Stick
Y,Hit	Returns(Y,Hit) = [1]	1	0	Hit

now estimate action-value pairs

X = (NoAce, 20, 8)

Y = (NoAce, 13, 8)

Over time, action-values and policy will approach optimal values due to repeated sampling + averaging.

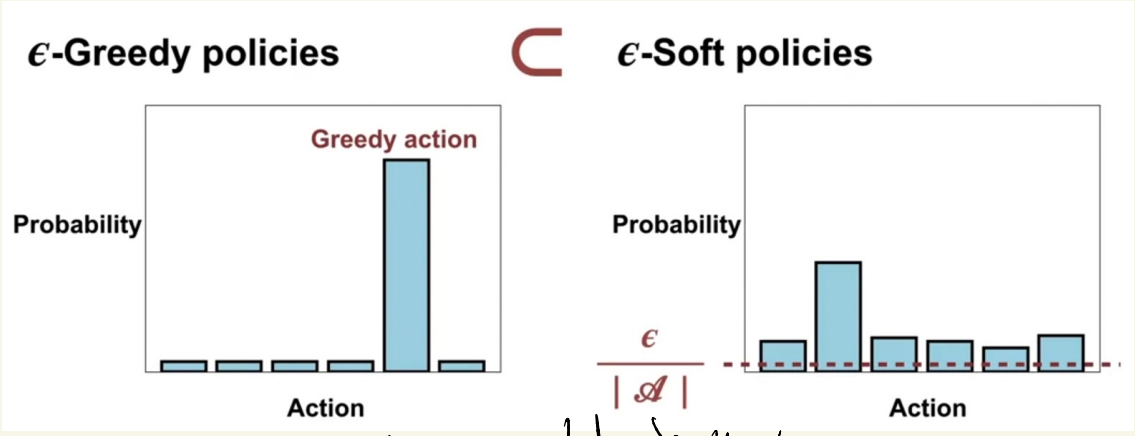


5.4 - Monte Carlo Control without Exploring Starts:

In many examples, it may be difficult to sample a random state-action pair. So how can we learn all the action-values w/t exploring starts?

Recall: ε-Greedy actions have a chance to select a random action.

↳ ε-Greedy policies are a subset of ε-Soft policies



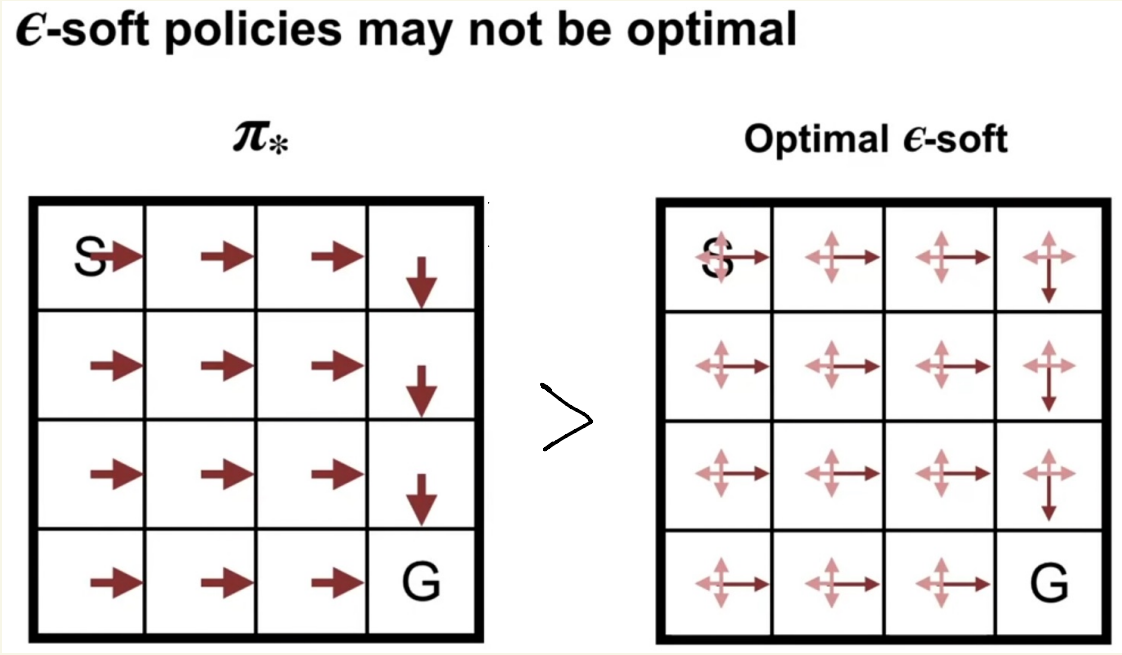
↳ Stochastic (i.e. uses probs) ↳ # actions

ε-Soft policies take each action w/ probability

at least $\frac{\varepsilon}{\#\text{ actions}} = \frac{\varepsilon}{|A(s)|}$.

↳ Forces agent to continously explore. Exploring starts is no longer needed.

- ε-Soft policies can only be used to find the optimal ε-Soft policy, not the actual optimal policy.
  - ↳ Cuz not deterministic.
- Q-learning is used to find the optimal policy (covered later).



- Performs better
- Performs reasonably well

Pseudocode:

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

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

Two Methods to Ensure All Actions are Selected:

On-policy methods attempt to evaluate/improve the policy that is used to make decisions.

↳ e.g. ε-soft policies, exploring starts

Off-policy methods evaluate/improve a policy <u>different</u> from that used to generate the data.

↳ e.g. Learning the optimal policy while following a completely random one.

5.5 - Off-Policy Prediction via Importance Sampling:

How can agents learn the optimal policy while behaving according to a non-optimal, exploratory policy?

↳ Just use two policies!

Target Policy: the one that is learned and becomes the optimal policy

Behavior Policy: the exploratory one used to generate behavior

✡ The learning is from data "off" the target policy, hence off-policy learning.

Notation:

Target Policy: $\pi(a|s)$

Behavior Policy: $b(a|s)$

↳ Behavior policy must cover target policy.
i.e. If $\pi(a|s) > 0$ for some (a,s), then $b(a|s) > 0$.

✡ On-policy: $\pi(a|s) = b(a|s)$

Derivation of Importance Sampling:

Sample: $x \sim b$

↳ Random variable x sampled from prob. dist. b.

Estimate: $E_\pi[X]$

↳ Expected value of x following π.

$$E_\pi[X] \equiv \sum_{x \in X} x\pi(x)$$

$$= \sum_{x \in X} x\pi(x)\frac{b(x)}{b(x)}$$

$$= \sum_{x \in X} x\frac{\pi(x)}{b(x)}b(x)$$

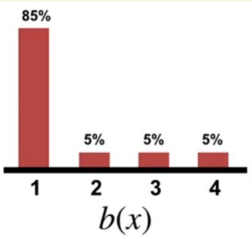
$\frac{\pi(x)}{b(x)} = \rho(x)$, importance sampling ratio

$$= E_b[X\rho(X)]$$

$$\approx \frac{1}{n}\sum_{i=1}^{n} x_i\rho(x_i) \quad \not{E}[X] \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

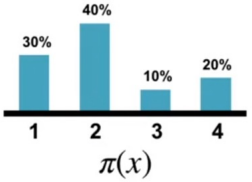
$$\therefore \boxed{E_\pi[X] \approx \frac{1}{n}\sum_{i=1}^{n} x_i\rho(x) \text{ for } x_i \sim b}$$

Example:

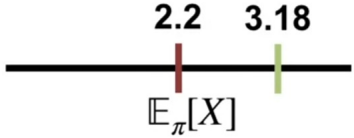


$$x = 1$$
$$b(x) = .85$$

x = [1, 3, 1]



$$\pi(x) = .3$$



$$\frac{1}{n}\sum_{1}^{n} x\rho(x) \longrightarrow \frac{(1 \times \frac{.3}{.85}) + (3 \times \frac{.1}{.05}) + (1 \times \frac{.3}{.85})}{2} = 2.24 \approx 2.2$$

Importance Sampling:

Almost all off-policy methods use importance sampling, which estimates expected values under one distribution given samples from another.

↳ We weigh returns according to the relative probabilities of their trajectories occurring under the target and behavior policies.

↳ Known as the importance-sampling ratio.

S_t; starting state

$A_t, S_{t+1}, A_{t+1} \dots S_T$; subsequent state-action trajectory

π; policy

p; state-transition probability function

$$Pr\{A_t, S_{t+1} \dots S_T \mid S_t, A_{t:T-1} \sim \pi\}$$

↳ Given agent in state S_t, what is prob. it takes A_t and ends up in state S_{t+1}, and so on.

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

Importance-Sampling Ratio;

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

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

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

✡ $\rho = \dfrac{P(\text{trajectory under } \pi)}{P(\text{trajectory under } b)}$

Pseudcode!

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 $W \neq 0$:
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
    $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

Compute $\rho_{t:T-1}$ Incrementally:

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

$$= \rho_t \rho_{t+1} \rho_{t+2} \cdots \rho_{T-2} \rho_{T-1}$$

$W_1 \leftarrow \rho_{T-1}$

$W_2 \leftarrow \rho_{T-1} \rho_{T-2}$

$\vdots$

$W_{t+1} \leftarrow W_t \rho_t$

5.7 - Off-Policy Monte Carlo Control:

Using off-policy evaluation and importance sampling for 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)
- $C(s,a) \leftarrow 0$
- $\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$ (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, $t = T-1, T-2, \ldots, 0$:
  - $G \leftarrow \gamma G + R_{t+1}$
  - $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
  - $\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken consistently)
  - If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
  - $W \leftarrow W \frac{1}{b(A_t|S_t)}$