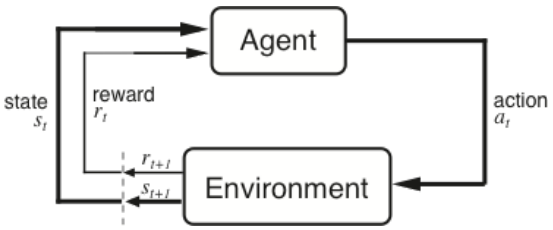
Monte Carlo Methods

Suggested reading:

Chapter 5 in R. S. Sutton, A. G. Barto: Reinforcement Learning: An Introduction
MIT Press, 1998.

Monte Carlo Methods



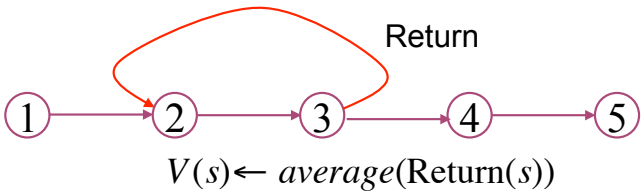
Contents:

Monte Carlo Methods

- Monte Carlo methods learn from *complete* sample returns (NOT partial returns)
  - Only defined for episodic tasks
  - Update after each episode (not step-by-step)
- Monte Carlo methods learn directly from experience
  - *On-line:* No model necessary and still attains optimality
  - *Simulated:* No need for a *full* model

Monte Carlo 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 asymptotically

First-visit Monte Carlo policy evaluation

Initialize:

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ an arbitrary state-value function

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

Repeat forever:

(a) Generate an episode using π

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

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

Append R to $Returns(s)$

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

→ See “evaluative feedback” for computing averages

Blackjack example

- *Object:* Have your card sum be greater than the dealers without exceeding 21.
- All face cards count as 10, and the ace can count either as 1 or as 11.
- *States*:
  - current sum (12-21)
  - dealer’s showing card (ace, 2-10)
  - do I have a usable ace?

- *Reward:* +1 for winning, 0 for a draw, -1 for losing (not discount)
- *Actions:* stick (stop receiving cards), hit (receive another card)
- *Policy:* Stick if my sum is 20 or 21, else hit

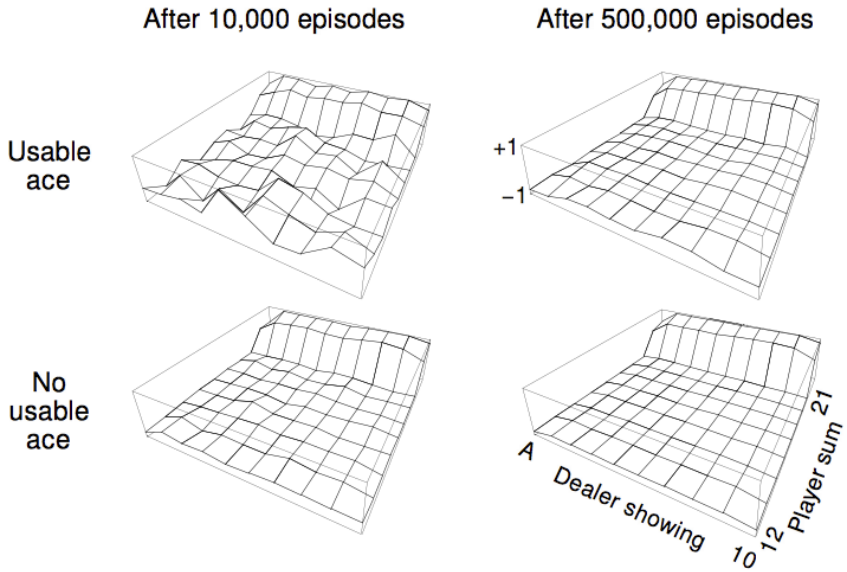
If the player holds an ace that he could count as 11 without going bust, then the ace is said to be *usable*.

Blackjack example

Dealer's fixed strategy
STICK if ≥ 17
HIT if < 17

Outcome:
if > 21 ⇒ LOSE
CLOSEST TO 21 ⇒ WIN
EQUALLY CLOSE ⇒ DRAW

Blackjack value functions



Used Policy: Stick if my sum is 20 or 21, else hit

To find the state-value function for this policy by a Monte Carlo approach, one simulates many blackjack games using the policy and averages the returns following each state. Note that in this task the same state never recurs within one episode, so there is no difference between first-visit and every-visit MC methods.

Monte Carlo vs Dynamic Programming

- Although we have complete knowledge of the environment in this task, it would not be easy to apply DP policy evaluation to compute the value function
- DP methods require the distribution of next events - in particular, they require the quantities $P^a_{ss'}$ and $R^a_{ss'}$

A **full policy-evaluation backup in DP**:

$$V_{k+1}(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V_k(s') \right]$$

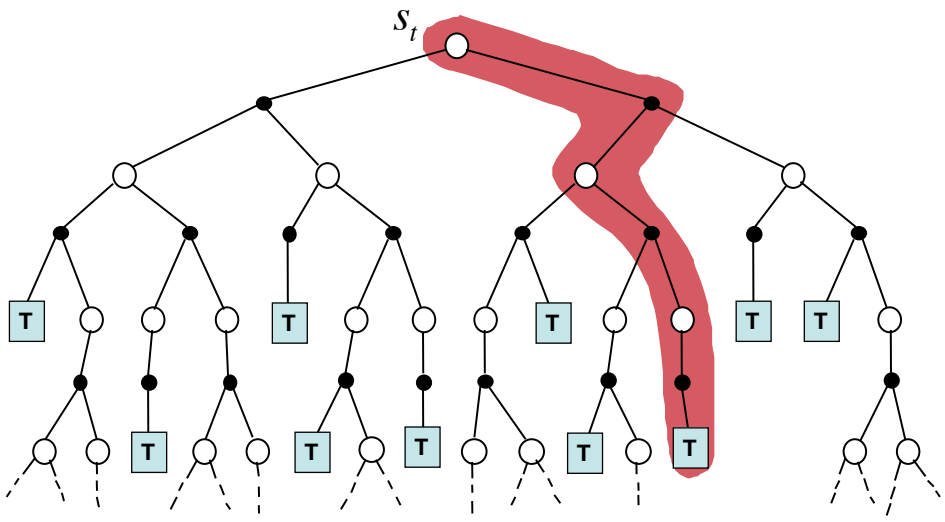
- and it is not easy to determine these for blackjack

For example, suppose the player's sum is 14 and he chooses to stick. What is his expected reward as a function of the dealer's showing card? All of these expected rewards and transition probabilities must be computed *before* DP can be applied.

Simple Monte Carlo

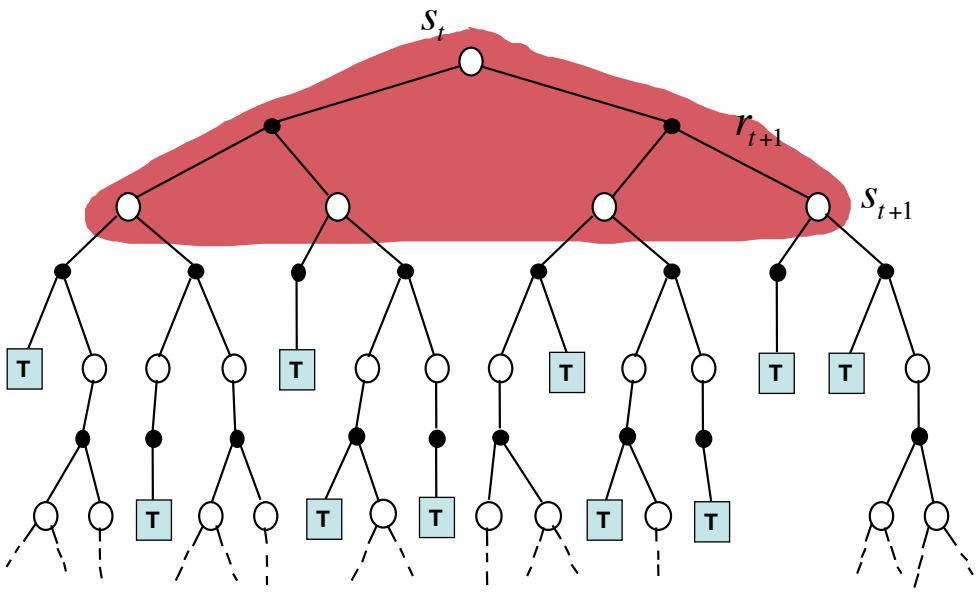
$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_t - V(s_t) \right]$$

where R_t is the actual return following state s_t.



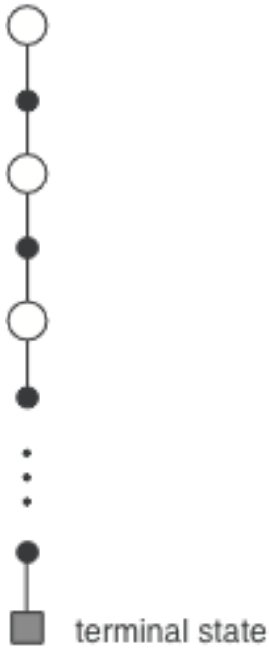
Dynamic Programming

$$V(s_t) \leftarrow E_\pi\left\{r_{t+1} + \gamma V(s_{t+1})\right\}$$



Backup diagram for Monte Carlo

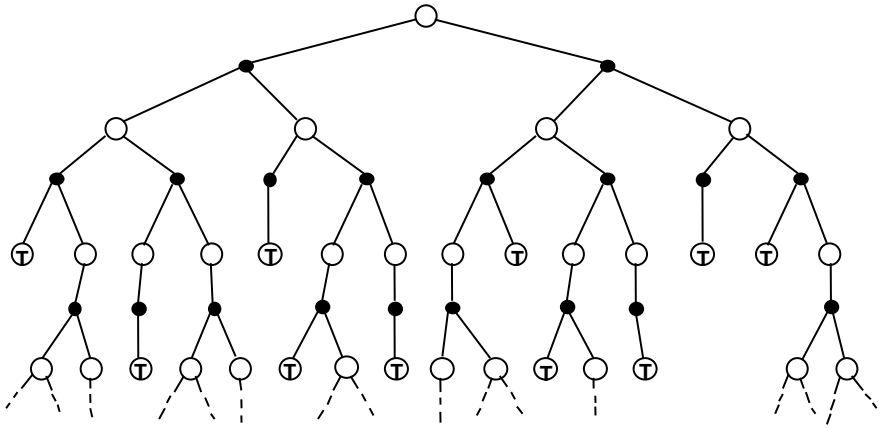
- Entire episode included
- Only one choice at each state (unlike DP)
- MC does not bootstrap
- Time required to estimate one state does not depend on the total number of states



Monte Carlo Estimation of Action Values (Q)

Monte Carlo is most useful when a model is not available

– Then it is particularly useful to learn Q^π



With a model, state values alone are sufficient to determine a policy; one simply looks ahead one step and chooses whichever action leads to the best combination of reward and next state, as in DP.
Without a model state values alone are not sufficient. One must explicitly estimate the value of each action in order for the values to be useful in suggesting a policy.

Monte Carlo Estimation of Action Values (Q)

- $Q^\pi(s,a)$ - average return starting from state *s* and action *a* following π

Problem:
If π deterministic, some/many (s, a) will never be visited

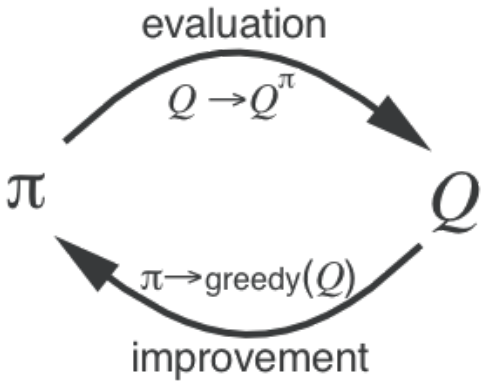
→ Explore

- *Exploring starts:* Every state-action pair has a non-zero probability of being the starting pair
- Or use ε -soft policies (as explained later)

Converges asymptotically *if* every state-action pair is visited

Monte Carlo Control

The overall idea is to proceed according to the same pattern as in DP:

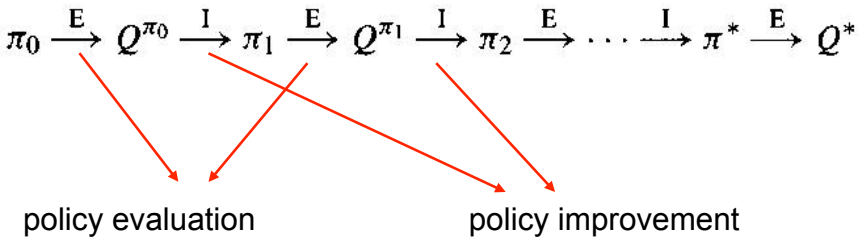


- MC policy iteration: Policy evaluation using MC methods followed by policy improvement
- Policy improvement step: greedify with respect to value (or action-value) function

Monte Carlo Control

Policy improvement is done by making the policy greedy with respect to the current value function:

$$\pi(s) = \arg\max_{a \in A(s)} Q(s,a)$$



Convergence of MC Control

Does the greedified policy meet the conditions for policy improvement?

For policy improvement construct each π_{k+1} as the greedy policy with respect to Q^{π_k} :

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

- Thus $\pi_{k+1} \geq \pi_k$ 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

Monte Carlo Exploring Starts

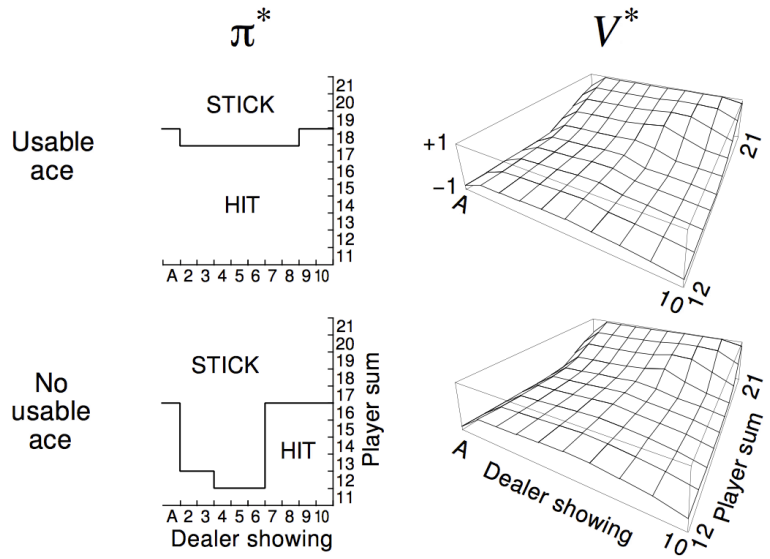
Alternate between evaluation and improvement on an episode-by-episode basis. After each episode, the observed returns are used for policy evaluation, and then the policy is improved at all the states visited in the episode.

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

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

Blackjack example continued

- Exploring starts
- Initial policy as used before (sticks on 20 or 21)



On-policy Monte Carlo Control

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

How do we get rid of exploring starts?

- *Soft* policies: $\pi(s,a) > 0$ for all s and all a

ε-soft:

$$\pi(s,a) \geq \frac{\varepsilon}{|A(s)|} \qquad \forall (s,a)$$

ε-greedy policy:

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

non-max greedy

On-policy MC Control

Generalized Policy Iteration does not require that the policy be taken all the way to a greedy policy, only that it be moved *toward* a greedy policy (i.e. ε-soft).

Evaluation is done as before.

That any ε-greedy policy with respect to Q^π is an improvement over any ε-soft policy π is assured by the policy improvement theorem.

On-policy MC Control

Let π' be the ε-greedy policy.

$$\begin{aligned} Q^\pi(s,\pi'(s)) &= \sum_a \pi'(s,a) Q^\pi(s,a) \\ &= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) + (1-\epsilon) \max_a Q^\pi(s,a) \qquad (5.2) \\ &\geq \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) + (1-\epsilon) \sum_a \frac{\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} Q^\pi(s,a) \end{aligned}$$

(the sum is a weighted average with nonnegative weights summing to 1, and as such it must be less than or equal to the largest number averaged)

$$\begin{aligned} &= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a Q^\pi(s,a) + \sum_a \pi(s,a) Q^\pi(s,a) \\ &= V^\pi(s). \end{aligned}$$

$$\pi' \geq \pi \iff V^{\pi'}(s) \geq V^\pi(s) \quad \forall s \in S$$

On-policy MC Control - proof

One also have to prove that equality can hold only when both π' and π are optimal among the ε -soft policies, that is, when they are better than or equal to all other ε -soft policies.

→ refer to the book

Policy iteration works for ε-soft policies: improvement on every step, except when the best policy has been found among the ε-soft policies.

We achieve the best policy *only* among the ε-soft policies, but, on the other hand, we have eliminated the assumption of exploring starts.

ε-soft algorithm for on-policy MC Control

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

Repeat forever:
(a) Generate an episode using π
(b) For each pair s, a appearing in the episode:
$R \leftarrow$ return following the first occurrence of s, a
Append R to $Returns(s,a)$
$Q(s,a) \leftarrow \text{average}(Returns(s,a))$
(c) For each s in the episode:
$a^* \leftarrow \arg\max_a Q(s,a)$
For all $a \in \mathcal{A}(s)$:

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

Evaluating one policy while following another

So far we have considered methods for estimating the value functions for a policy given an infinite supply of episodes generated using that policy.

Suppose now that all we have are episodes generated from a different policy:

Can we learn the value functions V^{π} or Q^{π} for a policy π, but all we have are episodes following the policy π'?

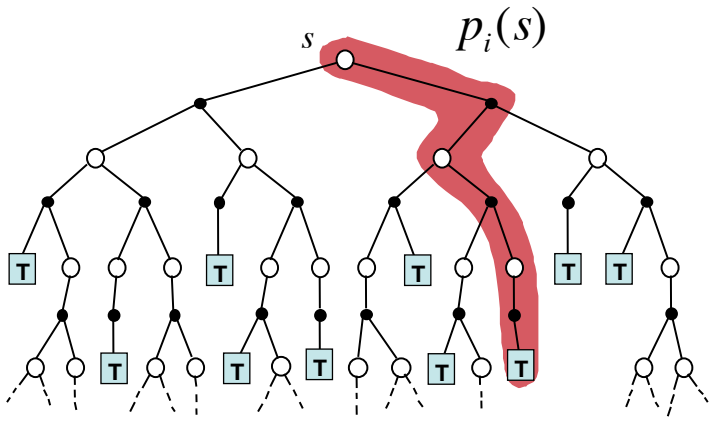
In order to use episodes from π' to estimate values for π, we require that every action taken under π is also taken, at least occasionally, under π'.

$$\pi(s,a) > 0 \quad \Rightarrow \quad \pi'(s,a) > 0$$

Evaluating one policy while following another

Consider the i^{th} first visit to state s and the complete sequence of states and actions following that visit.

Let $p_i(s)$ and $p'_i(s)$ denote the probabilities of that complete sequence happening given policies π and π' and starting from s.



Evaluating one policy while following another

Suppose we have n_s returns, $R_i(s)$, from state s, each with probability $p_i(s)$ of being generated by π and probability $p'_i(s)$ of being generated by π'. Then we can estimate

$$V^\pi(s) \approx \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}$$

which depends on the environmental probabilities $p_i(s)$ and $p'_i(s)$. However,

$$p_i(s_t) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}^{a_k}_{s_k s_{k+1}}$$

and

$$\frac{p_i(s_t)}{p'_i(s_t)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}^{a_k}_{s_k s_{k+1}}}{\prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) \mathcal{P}^{a_k}_{s_k s_{k+1}}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}.$$

Thus the weight needed, $p_i(s)/p'_i(s)$, depends only on the two policies and not at all on the environmental dynamics.

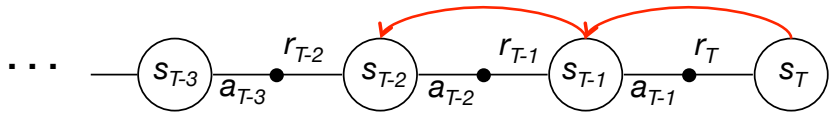
Off-policy Monte Carlo control

- **Behavior policy** (π') generates the moves
- **Estimation policy** (π) is policy being learned about
- estimation policy may be deterministic, e.g., greedy, while the behavior policy can continue to sample all possible actions

We need to:
- compute the weighted average of returns from behavior policy
- i.e. weight each return by relative probability of being generated by π and π'

Off-policy Monte Carlo control

1. Start at end of episode, work backwards



Till behavior policy and estimation policy give divergent actions, e.g. back to time *t*

2. For this chain of states and actions compute

$$\frac{p_i(s_t)}{p_i'(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}$$

Off-policy Monte Carlo control

π is deterministic (and gives the greedy actions) so $\pi(s_k, a_k)$ etc. = 1 and we know π'

$$\frac{p_i(s_t)}{p_i'(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} \qquad \frac{p_i(s_t)}{p_i'(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{1}{\pi'(s_k, a_k)}$$

3. Compute the action value function using the relative probabilities

$$Q(s,a) = \frac{\sum^{N} \frac{p_i}{p_i'} R}{\sum \frac{p_i}{p_i'}}$$

R = return for the chain of states/actions following (s, a)

Off-policy Monte Carlo control

4. Repeat this for each (*s, a*) in chain

5. Improve π (estimation policy) to be greedy

$$\pi(s) = \arg\max_a Q(s,a)$$

(Still deterministic, so still 1 for transitions within it.)

6. Generate new episodes and go back to 1. Repeat until estimation policy and Q values converge.

Takes a long time because we can only use the information from the end of the episode in each iteration.

Off-policy Monte Carlo control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$N(s,a) \leftarrow 0$; Numerator and
$D(s,a) \leftarrow 0$; Denominator of $Q(s,a)$
$\pi \leftarrow$ an arbitrary deterministic policy

Repeat forever:
(a) Select a policy π' and use it to generate an episode:
$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$
(b) $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$
(c) For each pair s, a appearing in the episode after τ:
$t \leftarrow$ the time of first occurrence (after τ) of s, a
$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$
$N(s,a) \leftarrow N(s,a) + wR_t$
$D(s,a) \leftarrow D(s,a) + w$
$Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
(d) For each $s \in \mathcal{S}$:
$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Incremental Implementation

MC can be implemented incrementally (saves memory) similar as in the armed bandit (but now: multiple states, and reward nonstationary).

Compute the weighted average of each return

$$V_n = \frac{\sum_{k=1}^{n} w_k R_k}{\sum_{k=1}^{n} w_k}$$

non-incremental

$$V_{n+1} = V_n + \frac{w_{n+1}}{W_{n+1}}\left[R_{n+1} - V_n\right]$$
$$W_{n+1} = W_n + w_{n+1}$$
$$V_0 = W_0 = 0$$

incremental equivalent

Summary

- MC has several advantages over DP:
  - Can learn *V* and *Q* directly from interaction with environment
  - No need for full models
  - No need to learn about ALL states
  - Less harm by Markovian violations (will be explained later)
- MC methods provide an alternate policy evaluation process
- One issue to watch for: maintaining sufficient exploration
  - exploring starts, soft policies
- *on-policy* and *off-policy* methods
- No bootstrapping, i.e. updating not dependent on successor states (as opposed to DP)