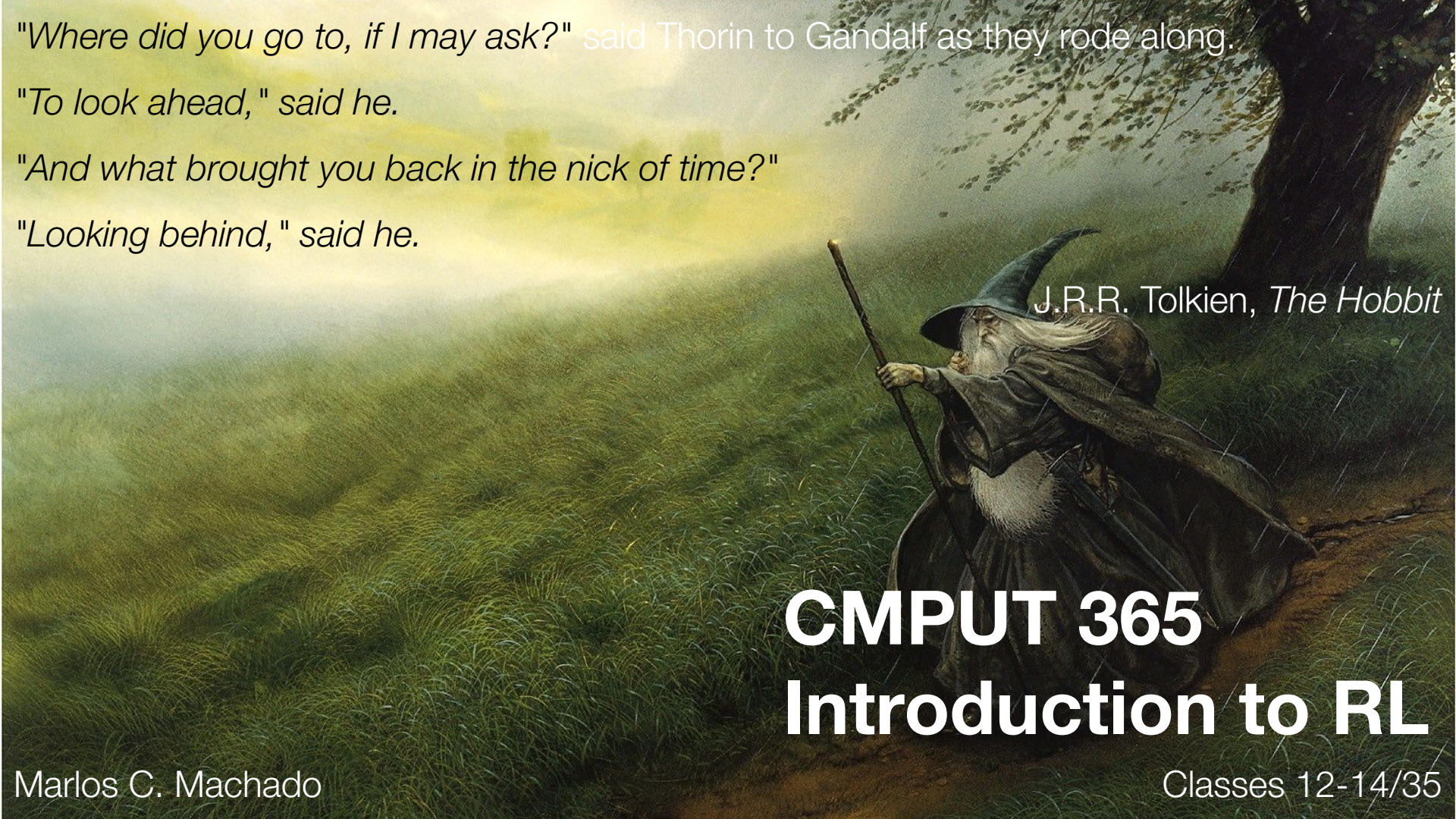


CMPUT 365
Introduction to RL

Marlos C. Machado

Classes 12-14/35

Reminder

You **should be enrolled in the private session** we created in Coursera for CMPUT 365.

I **cannot** use marks from the public repository for your course marks.

You **need** to **check**, **every time**, if you are in the private session and if you are submitting quizzes and assignments to the private section.

There were **43 pending invitations** last time I checked!

If you have any questions or concerns, **talk with the TAs** or email us `cmput365@ualberta.ca`.

Reminders and Notes

- What I plan to do today:
  - Overview of Monte Carlo Methods for Prediction & Control (Chapter 5 of the textbook).
- On the midterm:
  - We haven’t marked it yet, we have started.
- What I recommend YOU to do for next class:
  - Read Chapter 5 up to Section 5.5.
  - Graded Quiz (Off-policy Monte Carlo), which is due on Friday.
  - *Programming Assignment is not graded this week.*

Please, interrupt me at any time!

https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Interlude

An overview

- Main features of a reinforcement learning problem:
  - Trial-and-error learning
  - Exploration
  - Delayed credit assignment

An overview

- Main features of a reinforcement learning problem:
  - Trial-and-error learning
  - Exploration
  - Delayed credit assignment

A flavour of RL: Bandits (Chapter 2)

An overview

- Main features of a reinforcement learning problem:
  - Trial-and-error learning
  - Exploration
  - Delayed credit assignment

But what does that mean?
What is this sequential decision-making problem we are trying to solve?
What does solution mean here?

A problem formulation: MDPs (Chapter 3)

An overview

- Main features of a reinforcement learning problem:
  - Trial-and-error learning
  - Exploration
  - Delayed credit assignment

- What about the solution?

A first solution: Dynamic Programming (Chapter 4)

An overview

- Main features of a reinforcement learning problem:
  - Trial-and-error learning
  - Exploration
  - Delayed credit assignment

- What about the solution?
  - Dynamic programming! ← We need to know p(s', r | s, a) and it can be computationally expensive to solve the system of linear equations.

Our first learning algorithm: Monte Carlo Methods (Chapter 5)

Chapter 5

Monte Carlo Methods

Monte Carlo Methods – Why?

- This is our **first learning** method.
- We do not assume complete knowledge of the environment.
- “Monte Carlo methods **require only experience** — sample sequences of states, actions, and rewards from actual or simulated interaction with an environment.” 🤯
- It works! And different variations are used everywhere in the field (n-step returns, TD(λ), MCTS–AlphaGo/AlphaZero–, etc).
- … but we still “need” a model, albeit only a sample model.

MC Methods are ways of solving the RL problem based on avg. sample returns (similar to bandits, but instead of rewards we are sampling returns).

Monte Carlo 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: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

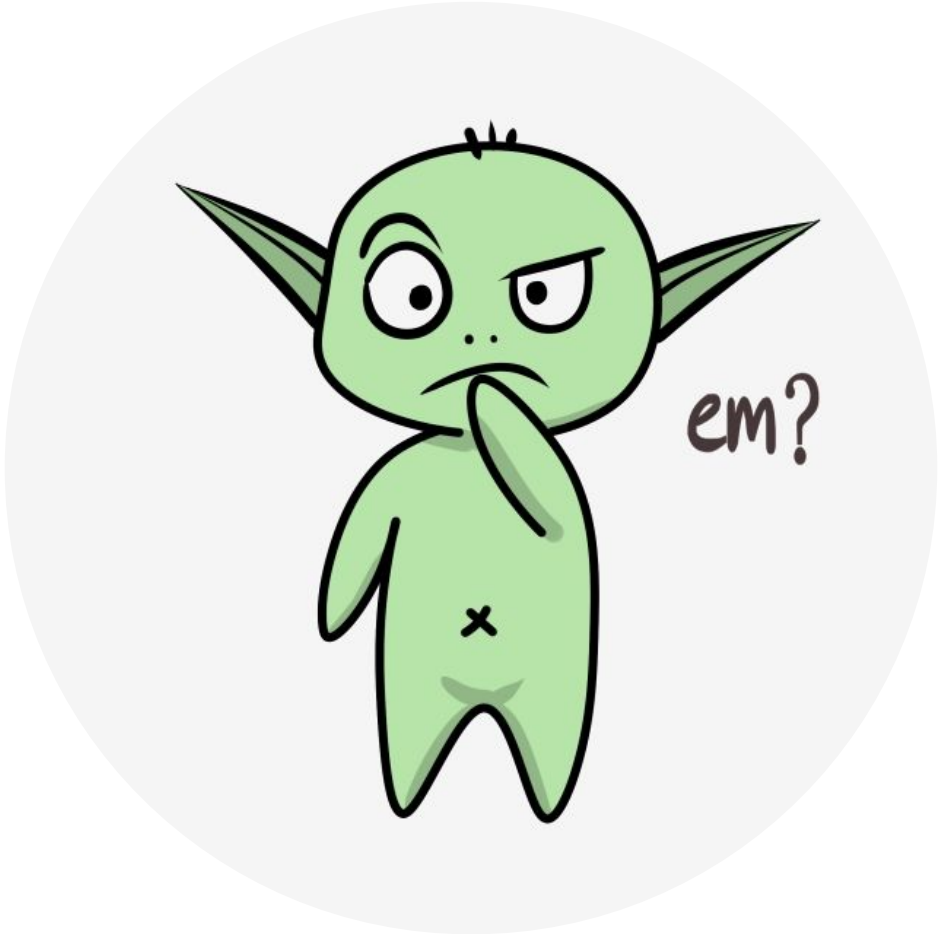
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

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

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

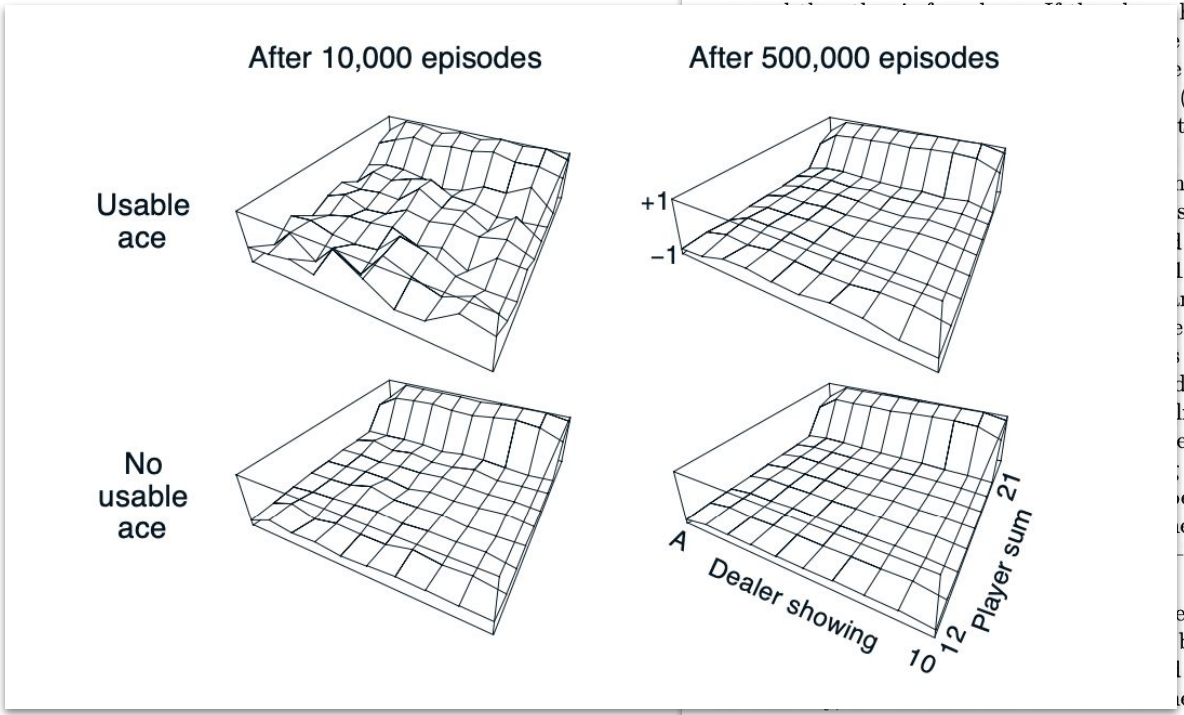
Append G to $Returns(S_t)$

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



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Blackjack: An example



Example 5.1: Blackjack The object of the popular casino card game of *blackjack* is to obtain cards the sum of whose numerical values is as great as possible without exceeding 21. All face cards count as 10, and an ace can count either as 1 or as 11. We consider the version in which each player competes independently against the dealer. The game begins with two cards dealt to both dealer and player. One of the dealer's cards is face

has 21 immediately (an ace and a 10-card),

dealer also has a natural, in which case the

a natural, then he can request additional

(*sticks*) or exceeds 21 (*goes bust*). If he goes

the dealer's turn. The dealer hits or sticks

he sticks on any sum of 17 or greater, and

he player wins; otherwise, the outcome—win,

sum is closer to 21.

as an episodic finite MDP. Each game of

, and 0 are given for winning, losing, and

me are zero, and we do not discount ($\gamma = 1$);

returns. The player's actions are to hit or

cards and the dealer's showing card. We

deck (i.e., with replacement) so that there is

already dealt. If the player holds an ace that

en the ace is said to be *usable*. In this case

it as 1 would make the sum 11 or less, in

because, obviously, the player should always

e basis of three variables: his current sum

–10), and whether or not he holds a usable

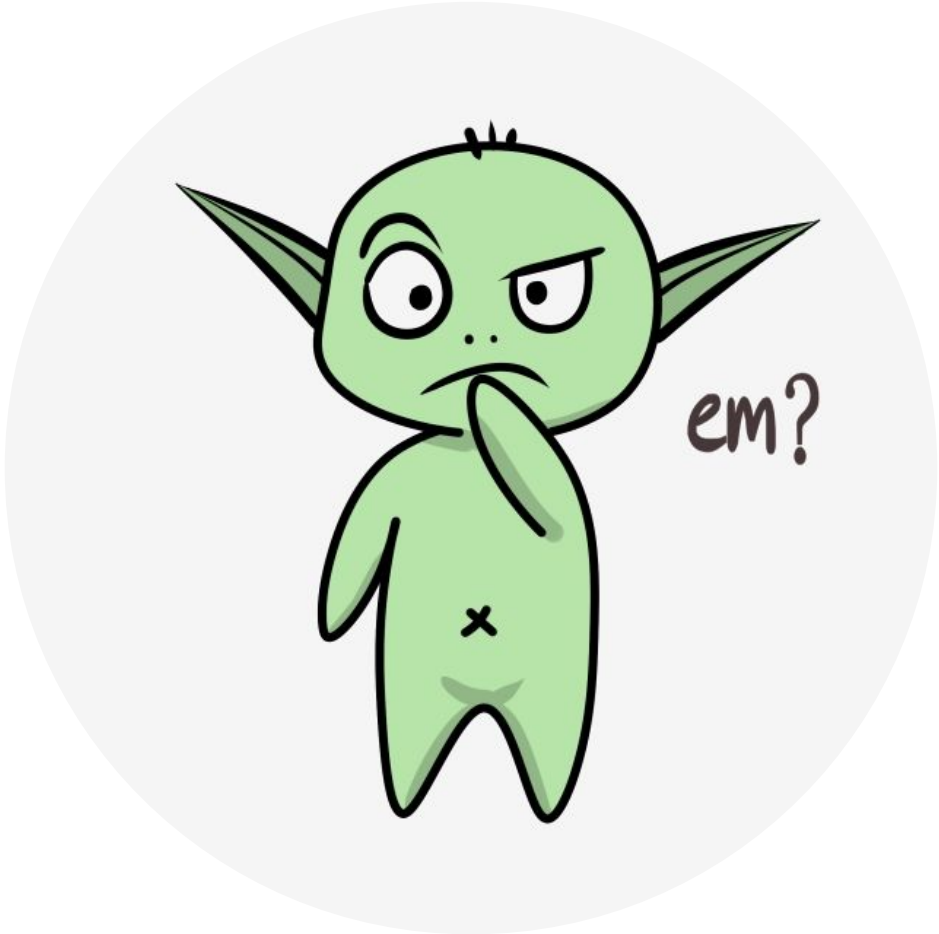
er's sum is 20 or 21, and otherwise hits. To

by a Monte Carlo approach, one simulates

averages the returns following each state.

e state-value function shown in Figure 5.1.

The estimates for states with a usable ace are less certain and less regular because these states are less common. In any event, after 500,000 games the value function is very well approximated.



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

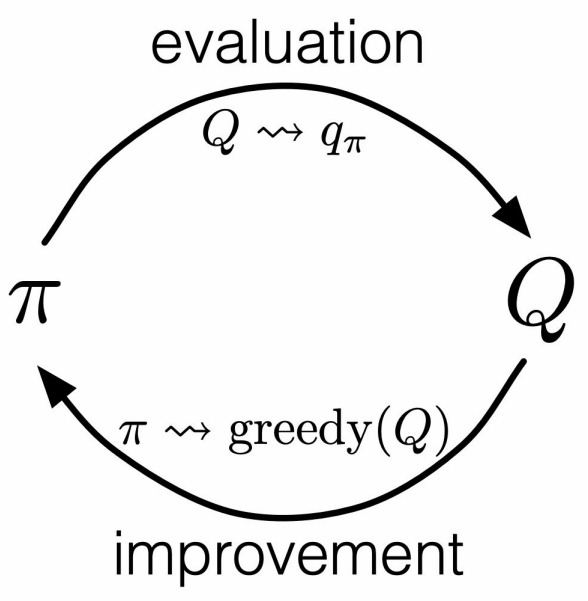
Some useful information / reminders about MC Methods

- Often it is much easier to get samples than to get the distribution of next events. Recall the Blackjack example in the textbook.
- Monte Carlo methods do not *bootstrap* (the estimate for one state does not build upon the estimate of any other state).
- First/every-visit MC converge to $v_\pi(s)$ as the number of visits to s goes to infinity. In first-visit MC, each return is i.i.d. and has finite variance ¯_(ツ)_/¯
- The computational cost of estimating the value of a single state is independent of the number of states.

Monte Carlo Estimation of Action Values

- If we don't have access to a model, we need to estimate *action* values.
- Same as before, but now we visit state-action pairs ¯_(ツ)_/¯
  But to estimate q_* we need to estimate the value of *all* actions from each state.
  Solution? Exploration! … or exploring starts 😟

Monte Carlo Control



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

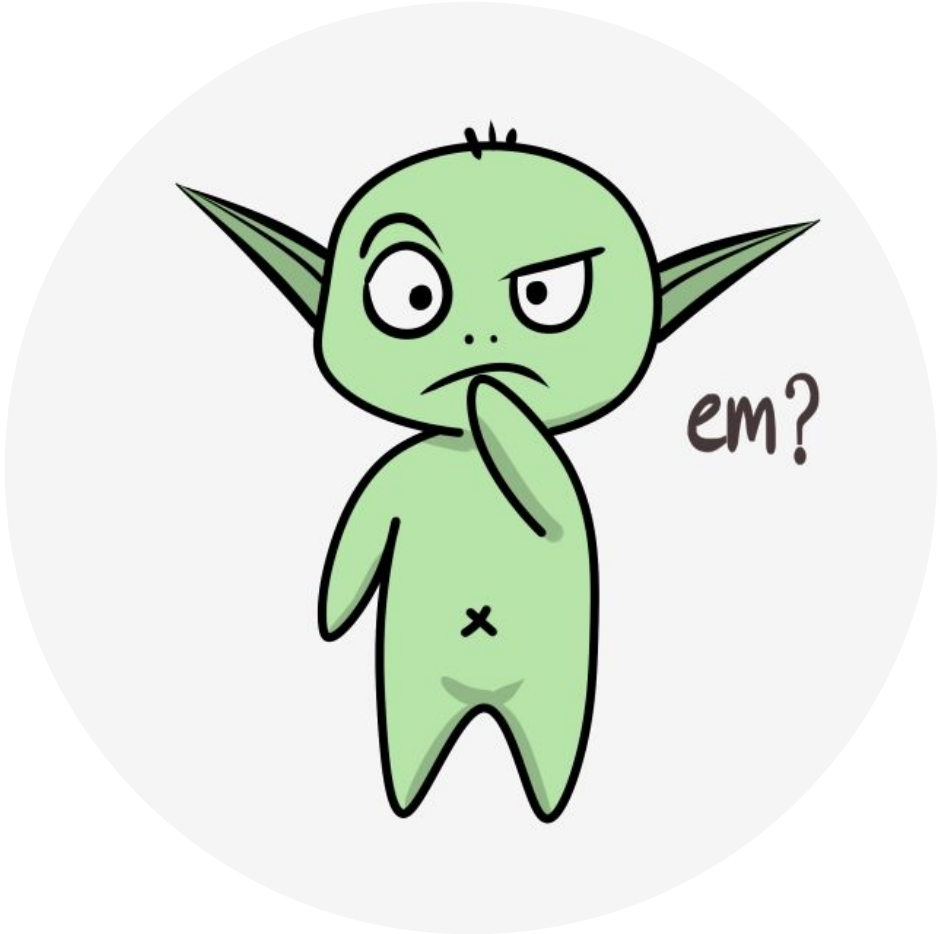
$$\pi(s) \doteq \arg\max_a q(s,a)$$

Monte Carlo ES

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



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

MC Control without Exploring Starts

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

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

We need to ensure that the probability we select each action is not zero.

MC Control without Exploring Starts

On-policy: You learn about the policy you used to make decisions.

Off-policy: You learn about a policy that is different from the one you used to make decisions.

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 $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

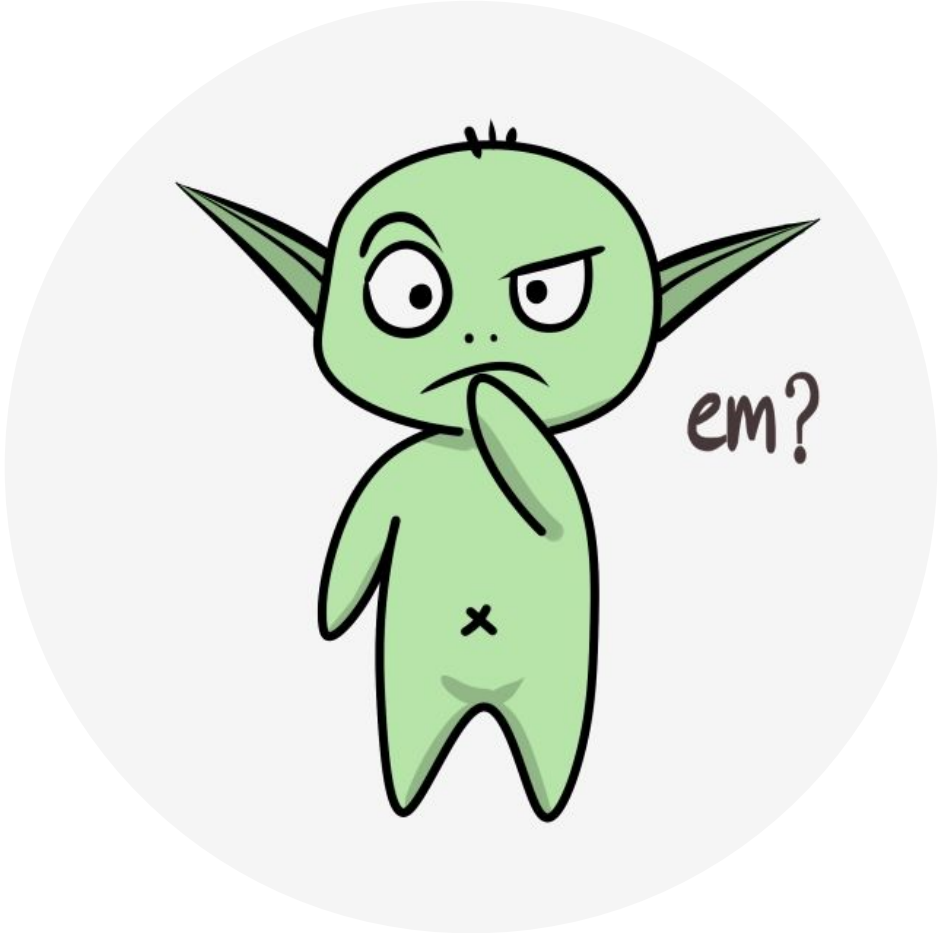
Append G to $Returns(S_t, A_t)$

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

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

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

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



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Learning with exploration

- *On-policy first-visit MC control (for ε-soft policies)* seems great!
- … but how can we learn about the optimal policy while behaving according to an exploratory policy? We need to behave non-optimally in order to explore 🤔.
- So far we have been *on-policy*, which is a compromise: we learn about a near-optimal policy, not the optimal one.
- But what if we had two policies? We use one for exploration but we learn about another one, which would be the optimal policy?

Behaviour policy

Target policy

That's off-policy learning!

Pros and cons of off-policy learning

Pros

- It is more general.
- It is more powerful.
- It can benefit from external data
  - and other additional use cases.

Cons

- It is more complicated.
- It has much more variance.
  - Thus it can be much slower to learn.
- It can be unstable.

Check Example 5.5 in the textbook about Infinite Variance

What's the actual issue?

Let π denote the target policy, and let b denote the behaviour policy.

We want to estimate $\mathbb{E}_{\pi}[G_t]$, but what we can actually directly estimate is $\mathbb{E}_{b}[G_t]$.

In other words, $\mathbb{E}[G_t \mid S_t = s] = v_b(s)$.

Importance Sampling

A general technique for estimating expected values under one distribution given samples from another. It is based on re-weighting the probabilities of an event.

Importance Sampling

$$\mathbb{E}_\pi[X] \doteq \sum_{x \in X} x\pi(x)$$

Importance Sampling

In RL, the probability of a trajectory is:

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

Importance Sampling

In RL, the probability of a trajectory is:

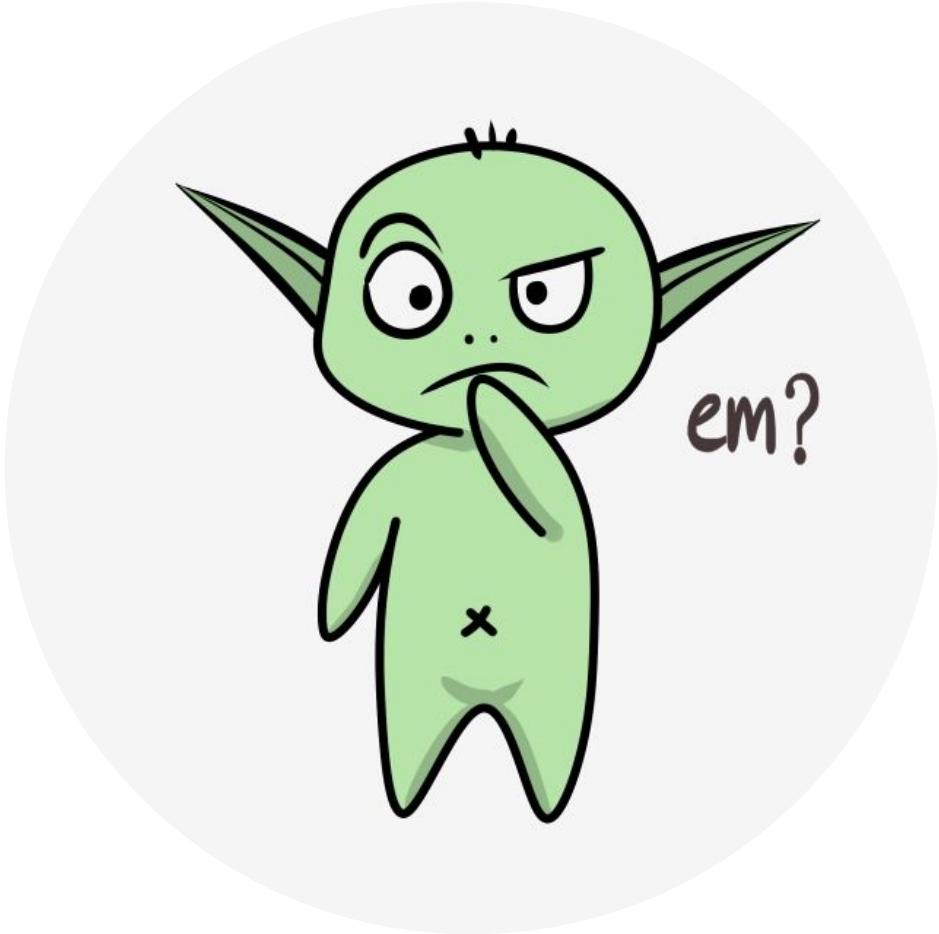
$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}$$

the relative prob. of the traj. under the target and behavior policies (the IS ratio) is:

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

We require *coverage*: b(a|s) > 0 when π(a|s) > 0

The IS ratio does not depend on the MDP, that is, on p(s', r | s, a)!



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

The solution

The ratio $\rho_{t:T-1}$ transforms the returns to have the right expected value:

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

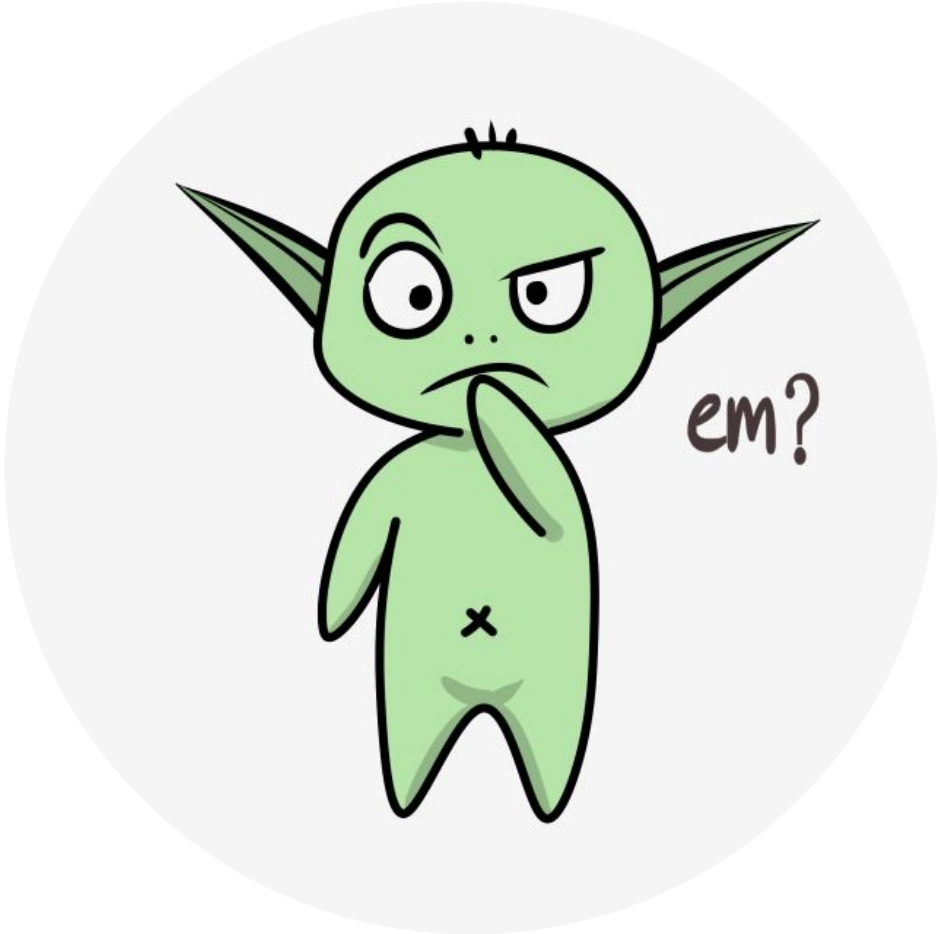
Ordinary importance sampling:

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

Set of all time steps in which state s is visited.

Weighted importance sampling:

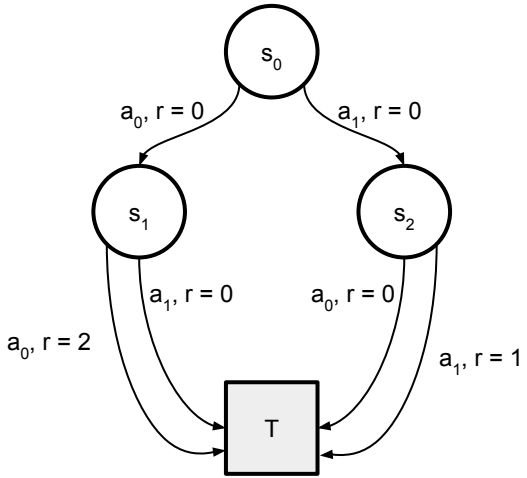
$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1}}$$



https://pngtree.com/freepng/question-expression-cartoon-illustration_4545209.ht

Practice Exercise 1

Consider the three-state MDP below with terminal state T and γ = 1. Suppose you observe three episodes: **{s_0, s_1, T}** with a return of 2, **{s_0, s_1, T}** with a return of 2, **{s_0, s_2, T}** with a return of 1. **What is the (every-visit) Monte-Carlo estimator of the value for each of the states, s_0, s_1, s_2?** How would the Monte-Carlo estimates change if r(s_0, a_1, s_2) = 1?



Practice Exercise 1

Consider the three-state MDP below with terminal state T and γ = 1. Suppose you observe three episodes: **{s_0, s_1, T}** with a return of 2, **{s_0, s_1, T}** with a return of 2, **{s_0, s_2, T}** with a return of 1. **What is the (every-visit) Monte-Carlo estimator of the value for each of the states, s_0, s_1, s_2?** How would the Monte-Carlo estimates change if r(s_0, a_1, s_2) = 1?

Trajectories:

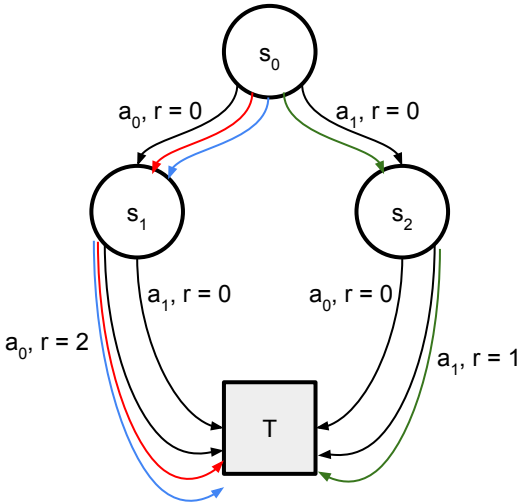
s_0, a_0, 0, s_1, a_0, 2, T
s_0, a_0, 0, s_1, a_0, 2, T
s_0, a_1, 0, s_2, a_1, 1, T

Returns:

0 + 2 = 2
0 + 2 = 2
0 + 1 = 1

States Visited / Return:

s_0, s_1, T / 2
s_0, s_1, T / 2
s_0, s_2, T / 1



Monte-Carlo Estimate

Returns from s_2: [1] → $V(s_2)$ = avg([1]) = 1
Returns from s_1: [2, 2] → $V(s_1)$ = avg([2, 2]) = 2
Returns from s_0: [1, 2, 2] → $V(s_0)$ = avg([1, 2, 2]) = 5/3

Practice Exercise 1

Consider the three-state MDP below with terminal state T and γ = 1. Suppose you observe three episodes: **{s_0, s_1, T}** with a return of 2, **{s_0, s_1, T}** with a return of 2, **{s_0, s_2, T}** with a return of 1. What is the (every-visit) Monte-Carlo estimator of the value for each of the states, s_0, s_1, s_2? **How would the Monte-Carlo estimates change if r(s_0, a_1, s_2) = 1?**

Trajectories:

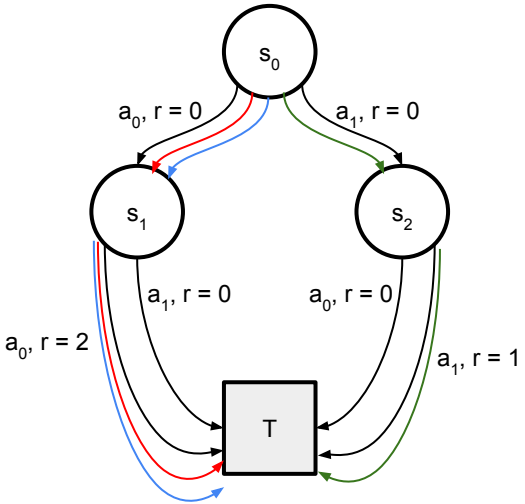
s_0, a_0, 0, s_1, a_0, 2, T
s_0, a_0, 0, s_1, a_0, 2, T
s_0, a_1, 0, s_2, a_1, 1, T

Returns:

0 + 2 = 2
0 + 2 = 2
0 + 1 = 1

States Visited / Return:

s_0, s_1, T / 2
s_0, s_1, T / 2
s_0, s_2, T / 1



Monte-Carlo Estimate

Returns from s_2: [1] → $V(s_2)$ = avg([1]) = 1
Returns from s_1: [2, 2] → $V(s_1)$ = avg([2, 2]) = 2
Returns from s_0: [1, 2, 2] → $V(s_0)$ = avg([1, 2, 2]) = 5/3

Practice Exercise 1

Consider the three-state MDP below with terminal state T and γ = 1. Suppose you observe three episodes: **{s_0, s_1, T}** with a return of 2, **{s_0, s_1, T}** with a return of 2, **{s_0, s_2, T}** with a return of 1. What is the (every-visit) Monte-Carlo estimator of the value for each of the states, s_0, s_1, s_2? **How would the Monte-Carlo estimates change if r(s_0, a_1, s_2) = 1?**

Trajectories:

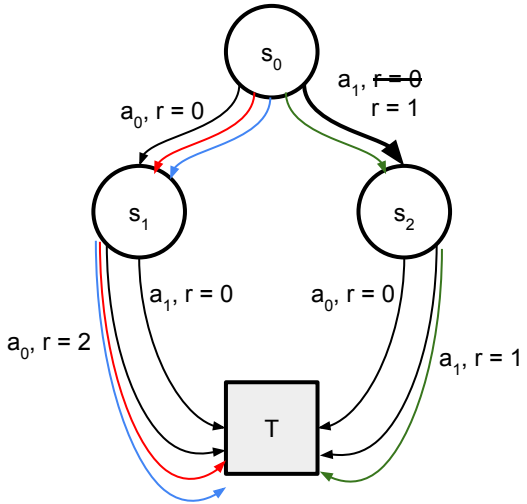
$s_0, a_0, 0, s_1, a_0, 2, T$
$s_0, a_0, 0, s_1, a_0, 2, T$
$s_0, a_1, 0, s_2, a_1, 1, T$

Returns:

0 + 2 = 2
0 + 2 = 2
0 + 1 = 1

States Visited / Return:

s_0, s_1, T / 2
s_0, s_1, T / 2
s_0, s_2, T / 1



Monte-Carlo Estimate

Returns from s_2: [1] → $V(s_2)$ = avg([1]) = 1
Returns from s_1: [2, 2] → $V(s_1)$ = avg([2, 2]) = 2
Returns from s_0: [1, 2, 2] → $V(s_0)$ = avg([1, 2, 2]) = 5/3

Practice Exercise 1

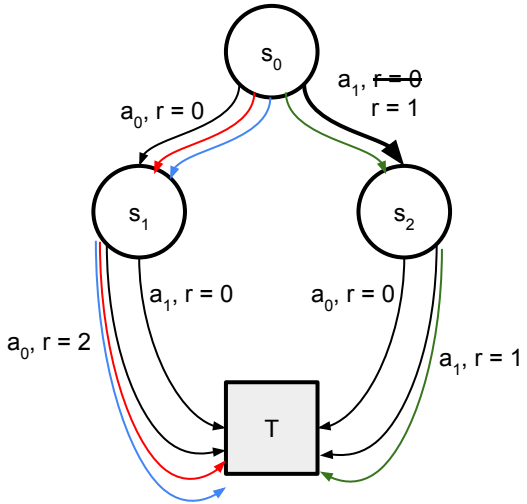
Consider the three-state MDP below with terminal state T and γ = 1. Suppose you observe three episodes: **{s_0, s_1, T}** with a return of 2, **{s_0, s_1, T}** with a return of 2, **{s_0, s_2, T}** with a return of 1. What is the (every-visit) Monte-Carlo estimator of the value for each of the states, s_0, s_1, s_2? **How would the Monte-Carlo estimates change if r(s_0, a_1, s_2) = 1?**

Trajectories: **Returns:**

$s_0, a_0, 0, s_1, a_0, 2, T$ — $0 + 2 = 2$
$s_0, a_0, 0, s_1, a_0, 2, T$ — $0 + 2 = 2$
~~$s_0, a_1, 0, s_2, a_1, 1, T$~~ — ~~$0 + 1 = 1$~~
$s_0, a_1, \mathbf{1}, s_2, a_1, 1, T$ — $1 + 1 = 2$

States Visited / Return:

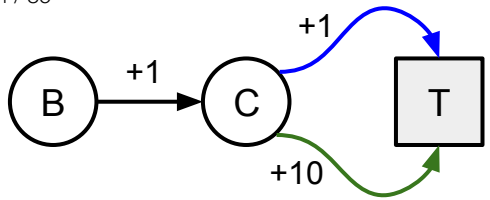
s_0, s_1, T / 2
s_0, s_1, T / 2
~~s_0, s_2, T / 1~~
s_0, s_2, T / 2



Monte-Carlo Estimate

Returns from s_2: [1] → $V(s_2)$ = avg([1]) = 1
Returns from s_1: [2, 2] → $V(s_1)$ = avg([2, 2]) = 2
~~Returns from s_0: [1, 2, 2] → $V(s_0)$ = avg([1, 2, 2]) = 5/3~~
Returns from s_0: [2, 2, 2] → $V(s_0)$ = avg([2, 2, 2]) = 2

Practice Exercise 2



Off-policy Monte Carlo Prediction allows us to use sample trajectories to estimate the value function for a policy that may be different than the one used to generate the data. Consider the following MDP, with two states, B and C, with 1 action in state B and two actions in state C, with γ = 1.0. In state C both actions transition to the terminating state with A = 1 following the blue path to receive a reward R = 1, and A = 2 following the green path to receive a reward R = 10. Assume the target policy π has π(A = 1 | C) = 0.9 and π(A = 2 | C) = 0.1, and that the behaviour policy *b* has *b*(A = 1 | C) = 0.25 and *b*(A = 2 | C) = 0.75.

a) What are the true values v_π?

b) Imagine you got to execute π in the environment for one episode, and observed the episode trajectory S_0= B, A_0= 1, R_1= 1, S_1= C, A_1= 1, R_2= 1. What is the return for B for this episode? Additionally, what are the value estimates V_π, using this one episode with Monte Carlo updates?

c) But you do not actually get to execute π; the agent follows the behaviour policy *b*. Instead, you get one episode when following *b*, and observed the episode trajectory S_0= B, A_0= 1, R_1= 1, S_1= C, A_1= 2, R_2 = 10. What is the return for B for this episode? Notice that this is a return for the behaviour policy, and using it with Monte Carlo updates (without importance sampling rations) would give you value estimates for *b*.

d) But we do not actually want to estimate the values for behaviour b, we want to estimate the values for π. So we need to use importance sampling rations for this return. What is the return for B using this episode, but now with importance sampling ratios? Additionally, what is the resulting value estimate for V_π using this return?

Practice Exercise 2