Monte Carlo

CMPUT 655
Fall 2022

Resources

- Chapter 2 of Ross: classic theoretical results for MDPs
  - E.g., on the uniqueness of v*
  - Link in schedule
- Errata for the videos
  - https://docs.google.com/document/d/1zJA_QDpgULraZahBd_mvciqe8ikFoF_gT9LQ_HCqNd8/edit?usp=sharing
- Course projects list:
  - https://docs.google.com/document/d/1BbsG3roJML0NK1IfkJk9sqfawpBiO8BNyOJoKzUvV9Q/edit?usp=sharing

Planning vs learning from interaction

- Recall: the universe consists of you, a chess board and pieces AND me (but I only play chess and I never talk, never respond to you)
  - The only thing you can do in life is play chess against me!!
- There are two ways you could figure out how to beat me:
  - Play me over and over and figure out how the game works; figure out my play. **Trial and error learning** (like in a bandit, like MC)
  - **Dynamic Programming:** If you had a **book** describing the rules & how I play (Adam is part of the environment)

Using Monte Carlo to beat me in chess

- You start with some policy: say move the piece forward closest to my nearest piece

1. Play a full game till the end against me:

   1.1. Policy/strategy is **frozen** during the game

   1.2. **Record** the states of the game you see (board) and the rewards

2. After the game:

   2.1. **Update** your **value function** for all the board configurations you saw

   2.2. **Update** your **policy**/strategy

   2.3. Goto 1

Why is MC a good idea here?

- **Model free**: Don't need p(s',r|s,a) —the book explaining the rules AND how Adam plays
- It's **adaptive**:
  - What if Adam changes how he plays?? **The book would be wrong!!**
    - *Then the MC agent can change its policy in response*
  - What if Adam disappears and Rich Sutton becomes your new opponent? *MC can adapt!*
- **Scalable**: if $|\mathcal{S}|$ is big, then p(s',r|s,a) is big
- **Focused** on relevant data: DP learns the optimal policy. Even in states Adam never plays in, MC does not! It *specializes to its opponent!*

Monte Carlo is a first principles algorithm

- Consider policy evaluation: estimating v_\pi given some policy \pi
- What is the definition of v_\pi? $v_\pi(s) \doteq \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \ldots \,|\, S_t = s] = \mathbb{E}_\pi[G_t \,|\, S_t = s]$
- v_\pi is equal to the **expected return**
  - It's the expected value of the *random variable G_t*
- How do we estimate an expected value?
  - We can use a sample average!
  - Generate some samples of G_t, then compute the average of them and that's it!

A family of Monte Carlo methods

Exploring Starts MC Control

on-policy:

we learn about the optimal policy, we follow the optimal policy

Use when:
can easily start the agent in any state

Epsilon-soft MC Control

on-policy:

we learn about the epsilon-soft optimal policy, we follow the epsilon-soft optimal policy

Use when:
can't do random starts, optimal policy not essential

Off-policy MC Control

off-policy:

we learn about the optimal policy, we follow an exploratory policy (e.g., epsilon-greedy)

Use when:
no obvious restrictions

On-policy and Off-policy learning

- MC control with ϵ-soft policies is an example of an **on-policy** learning algorithm
  - the policy that is used to generate sample episodes is the same as the policy we are learning about— the ϵ-greedy policy
- Imagine we generate sample episodes using some **behavior policy**: b: S X A -> [0,1]
  - perhaps a policy that explores a lot, like uniform random action selection in each state
- The **target policy**, π: S X A -> [0,1], is **not necessarily** the policy used to generate sample episodes
  - π is the policy we wish to learn about:
    - for Policy Iteration MC exploring starts, that was π_*,
    - for MC control (with ϵ-soft policies) that was $\widetilde{\pi}_\star$

Off-policy learning

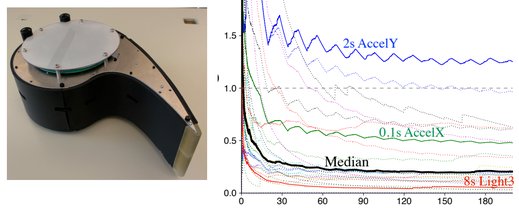
- What do we mean by learn **about** the target policy, π?
  - policy evaluation π: we learn the value function for π: v_π or q_π, OR
  - GPI: we the optimal policy π_*
  - all while generating episodes according to the behavior policy b
- This is called **off-policy** learning
- The special case where π = b is called **on-policy** learning

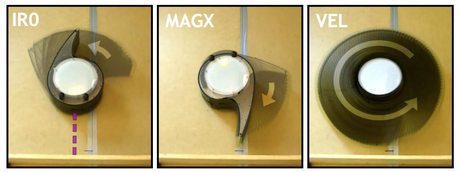
How does off-policy learning help for MC learning?

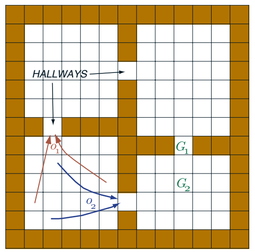
- It solves the problem of maintaining exploration in Monte Carlo GPI
  - we don't have to use unrealistic exploring starts
  - let b take care of exploration
    - just have to ensure: b(a|s) > 0 for every s,a that $\pi(a|s)>0$
  - we can learn π_* instead an ϵ-soft policy
- What famous algorithm is off-policy?

How else is off-policy learning useful?

- Massively parallel prediction learning: learn many V_π at once
- Learn many policies in parallel
- Learn models of the world
- Learn from data generated by a human
- Learn in the batch RL setting







Discussion posts

Advice

- If I mark your post with an ‘!’ I am telling you the post needs improvement. If you don’t understand ask.
  - It is your responsibility to pay attention
- Sometimes I ask you questions on your posts
- Asking the same question as others does not count. If you post late you need to read the other posts and ask a unique question

Epsilons

- Must epsilon be a constant? How could we change it?
- Examples of epsilon-soft policies
  - \pi(a|s)> 0 for all s and a
  - What is \pi(a*|s) for epsilon greedy?
- What might policies look like for continuous action spaces? Say an e-soft policy

MC methods

- What do we mean by incremental and why is it important?
- Why are optimistic initial values not effective in MC methods?
  - What type of applications are MC methods well suited for?
- **Can you use Monte Carlo updating, if you have a model?** If so, how? Is there more than one way?
- How can we handle continuing tasks?

IS and off-policy

- Don’t we get massive variance with long episodes?
  - How could this happen?
  - What approaches can mitigate this?
  - \pi(a|s)> 0 for all s and a
- “When using Monte Carlo with ES for control, Q(s, a) is calculated by averaging over all the episodes … the returns calculated from all the episodes are not from the same distribution.”

IS and off-policy

- Can the behavior policy change and why would that happen or be useful?
  - How can we design b to improve overall perf
    - … how about adapting the policy to improve learning
  - …what's wrong with random??!!
- How do we switch between on- and off-policy?
- Is off-policy better than on-policy?

Live Discussion Questions?

Terminology Review

- In Monte Carlo there are **no models, and no bootstrapping**
- **Experience**: data generate by the agent taking actions and getting reward feedback for the action it selected.
  - different from what Dynamic Programming does. DP updates the value of states using p(s',r|s,a). DP knows all the rewards in each state via p
- **Sample episodes**: starting in the start state, run policy pi (select actions according to pi) until termination, recording the states, actions, and rewards observed
- MC methods update the value estimates on an **episode-by-episode** basis. Must wait until the end of an episode to update the values of each state the agent observed

Terminology Review (2)

- **Maintaining exploration:** Why we need exploration in MC. Assume pi never takes action b in state S. If we want to estimate q(S,b) we will have no data about the reward you get from state S when pi chooses action b

- **Exploring starts:** every episode must begin in a random state, and the first action must be randomly selected, even if that action is not what pi would do

  - guarantees we visit every state-action pair

- **Epsilon-soft policies:** a stochastic policy. A policy where each action is selected with at least epsilon probability. (e.g., epsilon-greedy)

Choose your own adventure

Talk about predictive knowledge and AI
OR
Do an exercise together

AI & knowledge

- One objective of an AI agent is to know a lot
  - people know that the sun will rise, how to walk, how the desk will feel if I touch it, …
- My objective is to build learning systems that can know a lot about their world
  - learning for the sake of acquiring knowledge!!
- Determining how to represent and acquire knowledge is a classic problem in AI research

Humans know a lot

- We know we will feel an *impact* if *we reach out and touch* the wall
- We predict we will feel *pain* if we *bite our tongue*
- We know the water will feel *hot* a few seconds after *turning on the water faucet*
- We can predict the next *note* and *downbeat as we listen* to our favourite song
- We know so many things about our everyday interactions with our bodies and the world—**common sense knowledge**

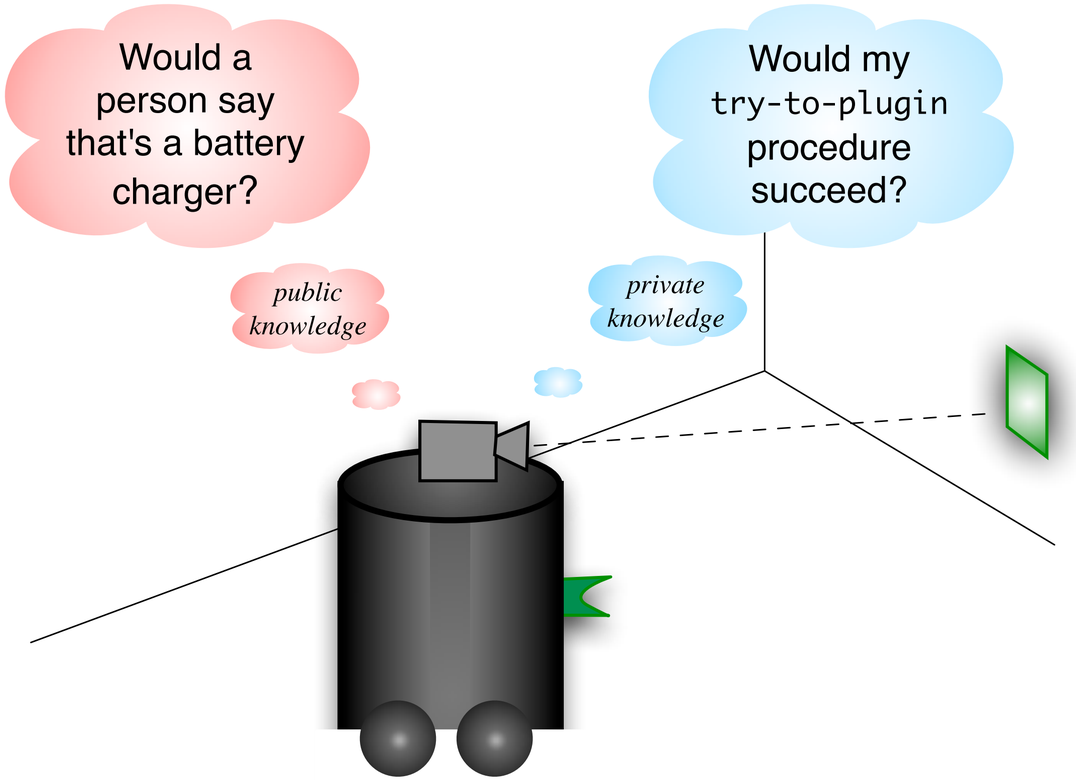
Animals seem wired to make predictions

- Animals form associations between a cue (e.g., tone), and a stimulus (e.g., shock) —classical conditioning
- Animals form predictive associations between stimuli even when the stimuli are not associated with pain, pleasure, or hunger
- Humans and animals continually make and refine large collections of predictions about what will happen next

More Examples of predictive knowledge

- How likely am I to bump into the wall over the next few seconds, if I were to walk forward?
- How long will it take me to get to the store? (declarative knowledge)
- How do I get to the printer? (procedural knowledge)

Predictive knowledge is private, personal, subjective



Public knowledge

- Logic statements:
  - e.g., sister(Amy, Jane), ∀ *x* likes (*x*, IceCream)
- Bayesian Network, dynamic Bayesian networks, POMDPs
- Publicly defined notion of correctness
  - e.g., “this thing is called a charger, because that is what a person would call it”
- Most popular type of knowledge representation used in AI systems

Public knowledge systems require humans for correctness

- Public knowledge is comprised of statements about entities in the world, e.g.:

  ➔ The label people associate with an image, correct clustering according to **you**, the location of a robot in global, objective X-Y-Θ coordinates

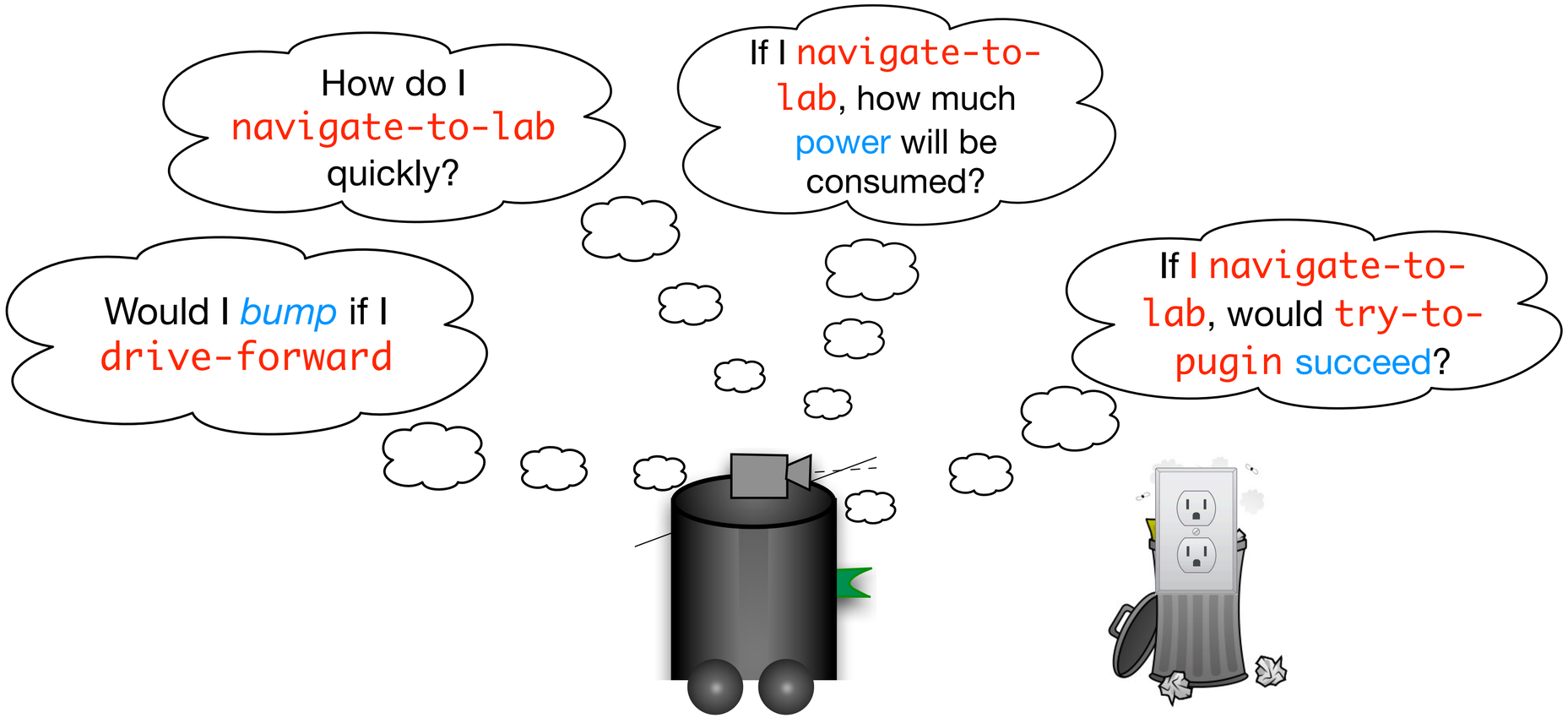
- These systems require a *publicly agreed upon* notion of *correctness*

- Humans are needed to provide consistent: labels, logical predicates, motion & observation models, …

- What happens if the labels are wrong, or the robot's body changes?

- Ultimately humans limit the scalability of public knowledge systems—we want our AI to know more than us!

A robot can learn many predictions in parallel



The predictive approach to knowledge

- Everything that the agent knows about the world is either a:

  1. **prediction** about some *outcome* in the future, conditioned on some *way of behaving*:

     ➔ I will feel *pain* if I *bite my tongue*

  2. **procedure** *(way of behaving)* for achieving some *outcome*

     ➔ *Where should I go* to maximise the *amount of coffee*

All scientific knowledge can be represented as a prediction or procedure —perhaps that is enough for building AI?!

Benefits of the predictive approach

- The knowledge can be updated from interaction with the world:
  - make a prediction, act, observe the outcome & update
- Knowledge can be maintained independent of people
- Learning can be scaled with *computation* and *data*
- Less developed than the public approach

Challenges for the predictive approach

- Building up from (potentially) low-level predictions to abstract knowledge
- Loss of human understanding
- Demonstrating how the knowledge can be used

Predictive knowledge is the key to scaling up AI

An agent can *update* it's predictive knowledge by *interacting with the world*

1. **Predictions** are kept correct with learning methods (e.g., Temporal Difference learning)
   - ➔ How can we can verify our prediction of pain? You simply bite your tongue!
   - ➔ It's the scientific method: predict, run an experiment, refine your belief
2. **Procedures** are updated with Reinforcement Learning methods (e.g., Q-learning):
   - ➔ We can improve a **procedure** for navigating to coffee via trial-and-error

The **correctness** of the knowledge is maintained through **learning**, and it does not require people, no matter how many predictions or procedures the agent stores!

Temporal Difference Learning is key to predictive knowledge

Practice exercise

Computing off-policy MC estimates

5. 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 $\gamma = 1.0$. Assume the target policy π has $\pi(A = 1|B) = 0.9$ and $\pi(A = 2|B) = 0.1$, and that the behaviour policy b has $b(A = 1|B) = 0.25$ and $b(A = 2|B) = 0.75$.

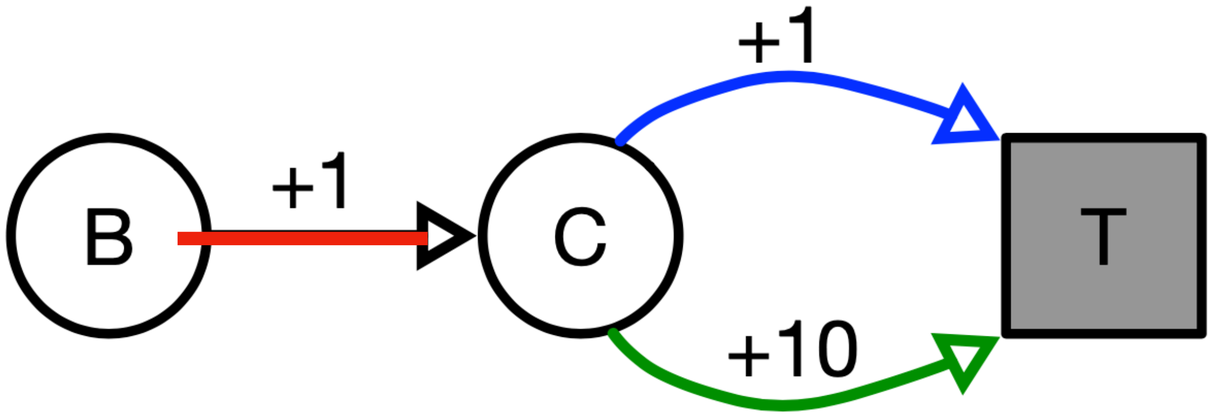
(a) What are the true values v_π?

(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?

Answers below…

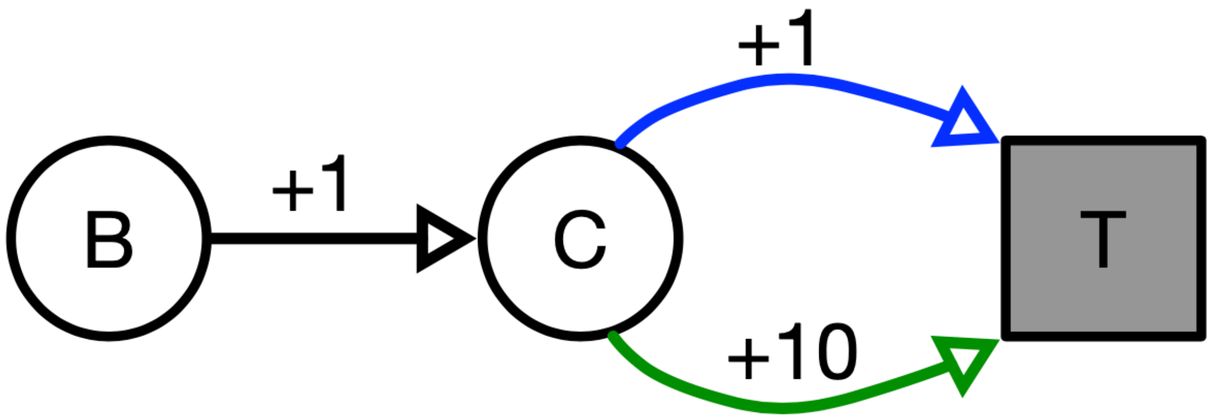
Computing off-policy MC estimates

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 $\gamma = 1.0$. 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 $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$.



Computing off-policy MC estimates

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 $\gamma = 1.0$. 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 $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$.

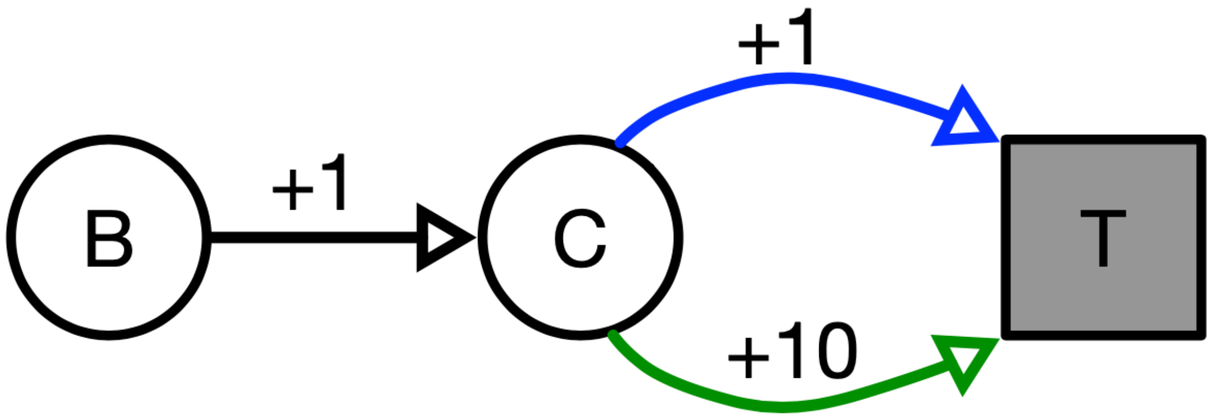


What v_\pi(C)?

hint:
v_\pi(s) = E[R] + \gamma v_\pi(s')

Computing off-policy MC estimates

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 $\gamma = 1.0$. 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 $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$.

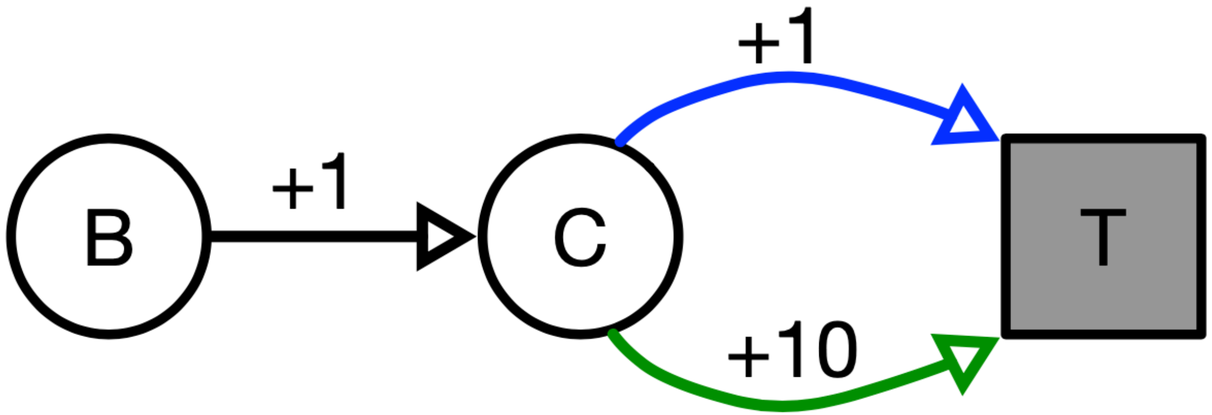


What v_\pi(C)
= 1*.9 + 10*0.1

=1.9

V_\pi(B) = ?

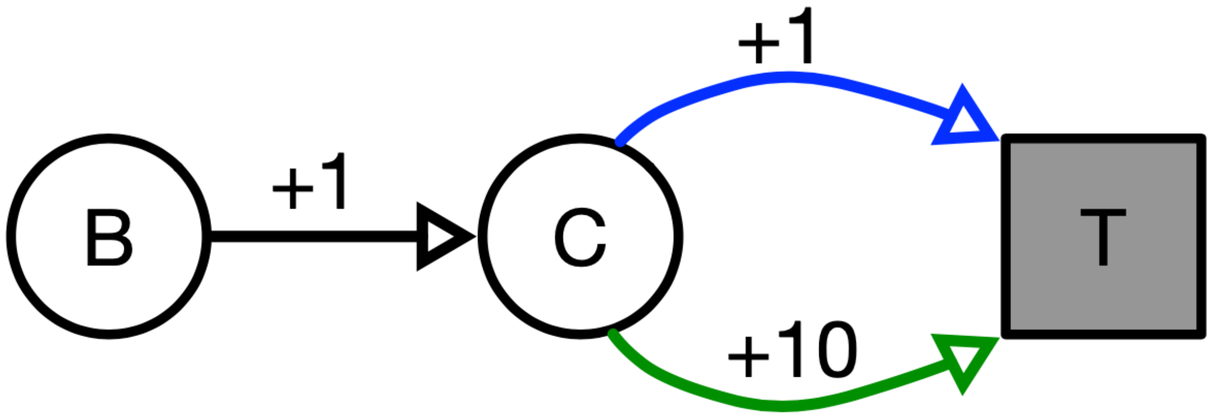
path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$



What is the return from this episode??

<S0= B, A0= 1, R1= 1, S1= C , A1= 1, R2= 1, S_2 = T>

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$



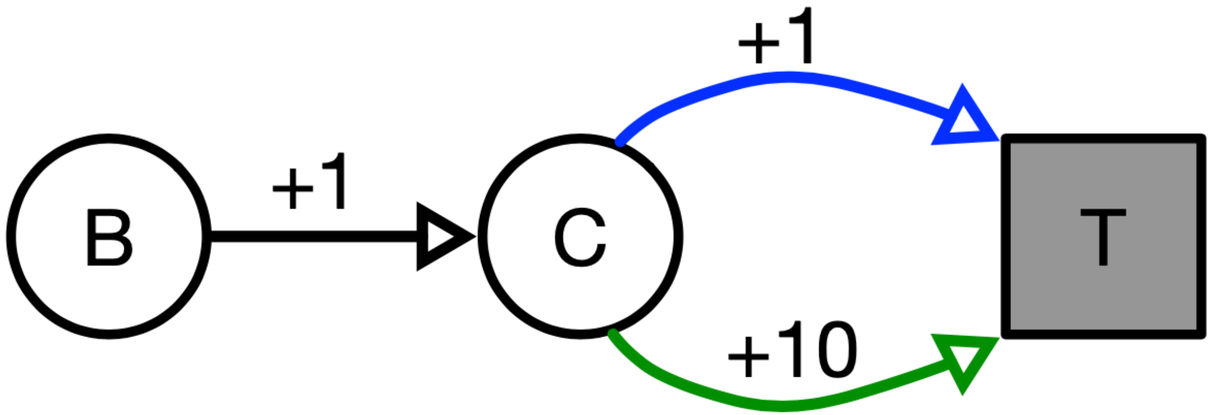
What is the return from this episode??

<S0= B, A0= 1, R1= 1, S1= C , A1= 1, R2= 1, S_2 = T>

G = 2

What is v_\pi(B) based on this one episode?

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$



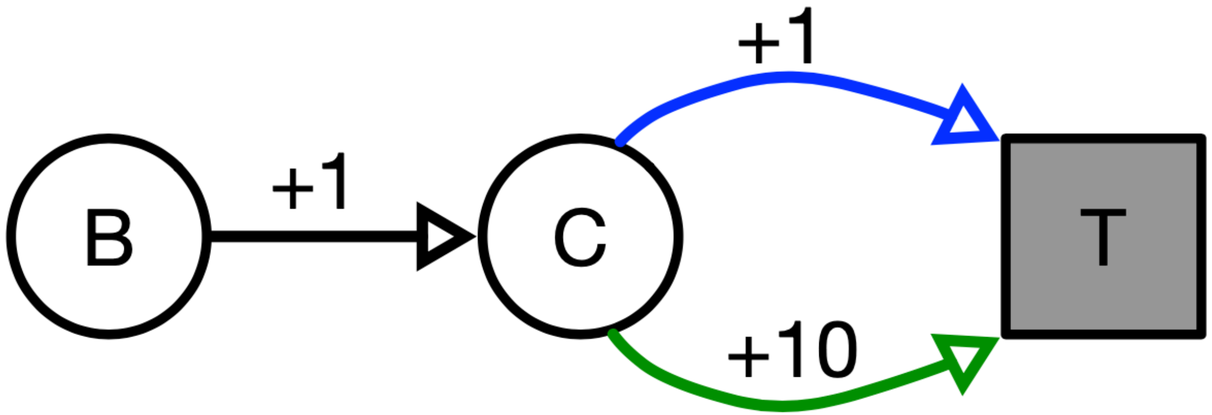
What is the return from this episode??

<S0= B, A0= 1, R1= 1, S1= C , A1= 1, R2= 1, S_2 = T>

G = 2

What is v_\pi(B) based on this one episode? 2

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$



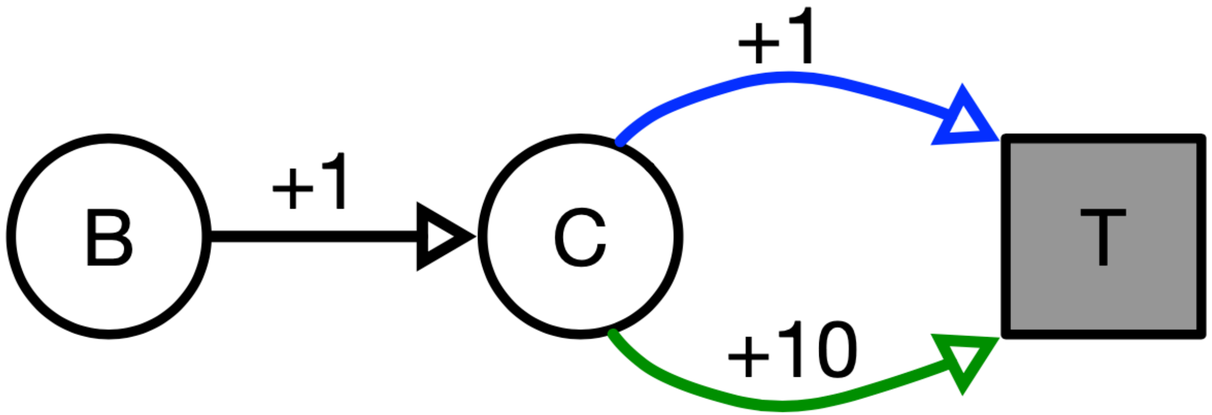
Now let's do **off-policy!!!**

Lets assume we get the following episode from policy b:

<S0= B, A0= 1, R1= 1, S1= C , **A1= 2, R2= 10**, S_2 = T>

G = 11

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$



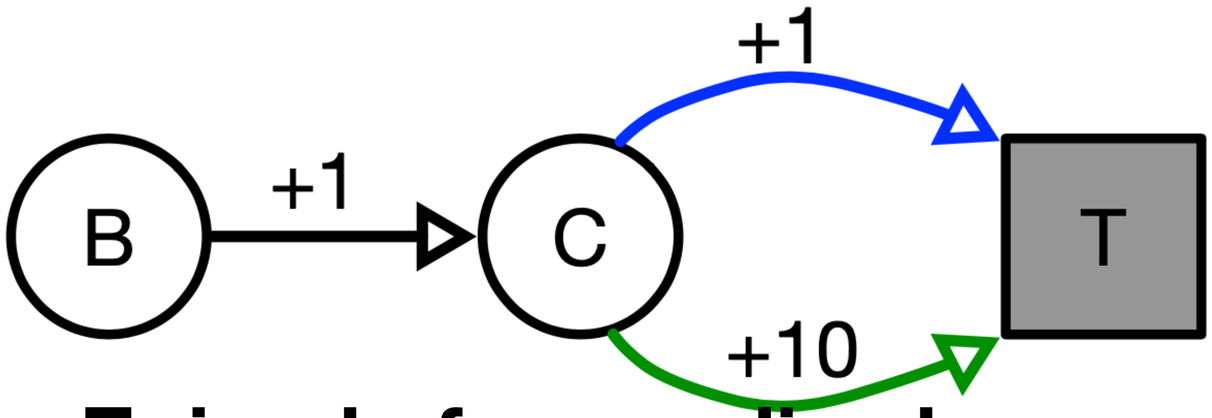
Episode from policy b:

<S0= B, A0= 1, R1= 1, S1= C , **A1= 2, R2= 10**, S_2 = T>

G = 11

What is prob of this episode under \pi?

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$



Episode from policy b:

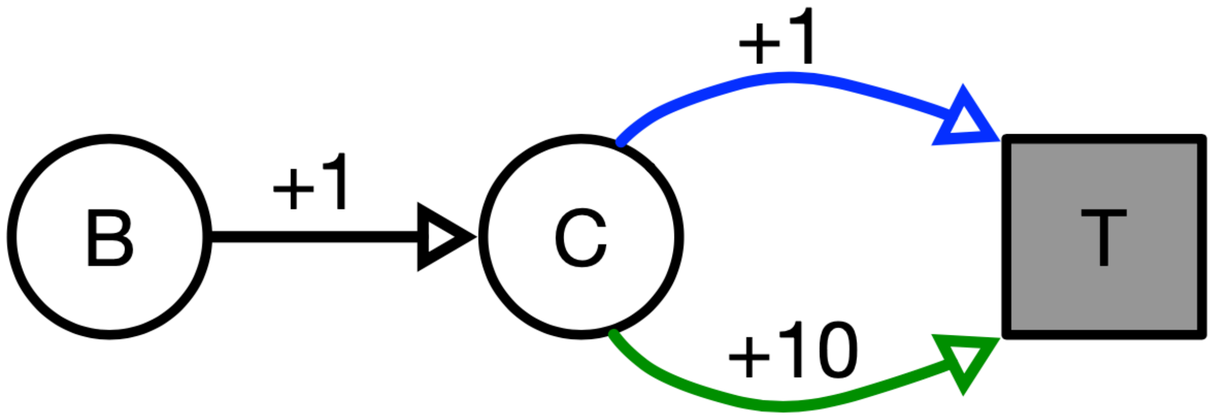
<S0= B, A0= 1, R1= 1, S1= C , **A1= 2, R2= 10**, S_2 = T>

G = 11

What is prob of this episode under \pi?

\pi(A=1| B)*\pi(A=2|C) = 1 * 0.1

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$.



Episode from policy b:

<S0= B, A0= 1, R1= 1, S1= C , **A1= 2, R2= 10**, S_2 = T>

G = 11

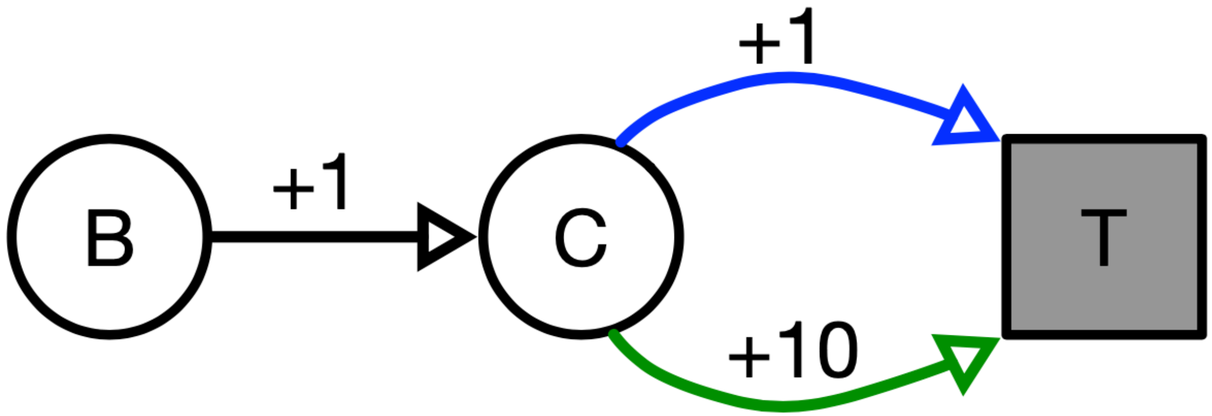
What is prob of this episode under \pi?

\pi(A=1| B)*\pi(A=2|C) = 1 * 0.1 = 0.1

What is prob of this episode under b?

b(A=1| B)*b(A=2|C) = 1 * 0.75 = 0.75

path to receive a reward $R = 10$. Assume the target policy π has $\pi(A = 1|C) = 0.9$ and $\pi(A = 2|C) = 0.1$, and that the behaviour policy b has $b(A = 1|C) = 0.25$ and $b(A = 2|C) = 0.75$.



Episode from policy b:

<S0 = B, A00 = 1, R1 = 1, S1 = C , **A1 = 2, R2 = 10**, S_2 = T>; G = 11

Prob of this episode under \pi? \pi(A=1| B)*\pi(A=2|C) = 1 * 0.1 = 0.1

Prob of this episode under b? b(A=1| B)*b(A=2|C) = 1 * 0.75 = 0.75

What would be the off-policy MC estimate of V_\pi(B) using this episode from b (using the importance sampling ratio)? **1.47**