# Monte Carlo Methods

Diego Ascarza-Mendoza

Escuela de Gobierno y Transformación Pública

## Outline

## Introduction

- Now we start developing our first learning methods for estimating value functions and discovering optimal policies.
- The special thing here is that: we **do not** assume complete knowledge of the environment.
- Monte Carlo methods only require experience: samples of states, actions, and rewards from actual or simulated interaction with an environment.
- Learning from experience is nice because it does not require any previous knowledge of the environment's dynamics and yet can lead to optimal behavior.
- Although a model is needed, the model only needs to generate sample transitions, not the complete probability distributions of all possible transitions.

## Introduction

- Monte Carlo methods are ways of solving the RL problem based on averaging sample returns.
- We are going to focus in this section to episodic tasks.
- Monte Carlo methods can be incremental in an episode-by-episode sense, but not in a step-by-step sense.
- There are similarities with the bandit problems we studied before, except that now we have multiple states, each acting like a different bandit problem.
- We are going to adapt the ideas from GPI. IN DP we computed value functions from knowledge of the MDP, here we learn value functions from sample returns with the MDP.
- As before, we will consider the prediction problem and the policy improvement. Then, we will have the control problem and its solution by GPI.

## Outline

## Monte Carlo Prediction

- The value of a state is its expected return.
- An obvious way to estimate the value of a state is simply to average the returns after visits to that state (more visits should converge to true return).
- We wish to estimate $v_\pi(s)$. Each occurrence of state $s$ in an episode is called a *visit* to $s$.
- $s$ may be visited many times. Let's call the first it is visited in an episode *the first visit* to s.
- *The first-visit MC method* estimates $v_\pi(s)$ as the average of the returns following first visits to $s$.
- The *every-visit MC method* averages the returns following all visits to $s$.
- Both first-visit MC and every-time visit MC converge to $v_\pi(s)$ as the number of visits to $s$ goes to infinity.
- Let's check an algorithm for first-visit MC prediction.

## First-visit MC prediction

Input: a policy $\pi$ to be evaluated.

1. Initialization: $V(s) \in \mathbb{R}$, arbitrarily $\forall s \in S$.

$$\text{Returns}(s) \leftarrow \quad \text{an empty list}, \quad \forall s \in \mathcal{S}$$

2. Loop forever (for each episode):
    - Generate an episode following $\pi$: $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$.
    - $G \leftarrow 0$.
    - Loop for each step of episode, $t = T-1, T-2, ..., 0$:

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

    - Unless $S_t$ appears in $S_0, ..., S_{t-1}$:
        - Append $G$ to Returns$(S_t)$.
        - $V(S_t) \leftarrow$ average$(Returns(S_t))$.

## Example - Blackjack

- Consider the game blackjack in which the goal 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.
- 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 up and the other is face down.
- If the player has 21 immediately (an ace and a 10-card), it is called a natural. He then wins unless the dealer also has a natural, in which case the game is a draw.
- If the player does not have a natural, then he can request additional cards, one by one (hits), until he either stops (sticks) or exceeds 21 (goes bust).
- If he goes bust, he loses; if he sticks, then it becomes the dealer's turn. The dealer hits or sticks according to a fixed strategy without choice: he sticks on any sum of 17 or greater, and hits otherwise.
- If the dealer goes bust, then the player wins; otherwise, the outcome—win, lose, or draw—is determined by whose final sum is closer to 21.

## Example - Blackjack

- Playing Blackjack is formulated naturally as a MDP.
- Each game is an episode.
- Rewards of $+1$, $0$, and $-1$ for winning, drawing, and losing, respectively.
- All rewards within a game are zero, we do not discount $\gamma = 1$.
- Actions: hit or stick.
- States depend on the player's cards and the dealer's showing card. Assume there is an infinite deck for simplicity (with replacement).
- If the player holds an ace that he could count as 11 without going bust, then the ace is said to be *usable*.
- Player makes a decision based on three variables: current sum, dealer's one showing card, and whether he holds a usable card.

## Example - Blackjack

- Consider the policy that sticks if the player's sum is 20 or 21, and otherwise hits.
- To find the state-value function for this policy by a MC approach, simulate many blackjack games using the policy.
- Then average the returns following each state.
- Estimates for states with a usable aces are less certain and less regular because these states are less common.
- After 500,000 the value function is very well approximated.

## Why DP is hard for this problem?

- We have complete knowledge of the environment in the blackjack task.
- Computing DP methods is hard here, though, because it requires the distribution of next events. In particular, it requires knowing $p$, and this is not easy for blackjack.
- For instance, suppose the player's sum is 14 and chooses to stick. What is the probability of ending up with a reward of $+1$ as a function of the dealer's showing card?
- All probabilities must be computed before DP can be applied, and usually these computations are often complex and error-prone.
- In contrast, generating sample games required by Monte Carlo methods is easy: Monte Carlo is amazing for its ability to work with sample episodes alone (useful when hard to know the whole dynamics).

## Some Remarks

- MC methods have estimates for each state that are independent.
- The estimate for one state does not build upon the estimate of any other state, as is the case in DP.
- In other words, MC methods do not bootstrap as we defined it in DP.
- This implies that the computational expense of estimating the value of a single state is independent of the number of states.
- This makes MC attractive when one requires the value of only one or a subset of states.
- Overall, this gives MC methods 3 advantages relative to DP:
    1. Ability to learn from actual experience.
    2. Ability to learn from simulated experience.
    3. Ability to generate many sample episodes starting from states of interest, averaging returns only from those states.

## Outline

## Monte Carlo Estimation of Action Values

- If the model is not available, it is particularly useful to estimate action values rather than state values.
- With a model, state values alone are sufficient to determine a policy: look ahead one step and choose whichever action leads to the best combination of reward and next state.
- Without a model, state values are not sufficient. One must explicitly estimate the value of each action to get a nice policy.
- Therefore, in MC methods, our primary goal is to estimate $q_{**}$.
- To do this, we first consider the policy evaluation problem of action values.
- $q_\pi(s, a)$ is the expected return when starting in state $s$, taking action $a$, and thereafter following policy $\pi$.
- This is quite similar to estimating state values, except that now we talk about visits to a state-action pair rather than a state.

## Monte Carlo Estimation of Action Values

- A stat-action pair $s, a$ is said to be visited in an episode if ever the state $s$ is visited and action $a$ is taken in it.
- The every-visit MC method estimates the value of a state action pair as the average of the returns that have followed all the visits to it.
- First-visit MC averages returns following the first time in each episode that the state was visited and the action was selected.
- These methods converge quadratically to true expected values as before.
- The problem is that many state-action pairs may never be visited. If $\pi$ is a deterministic policy, one will observe returns only for one of the actions from each state.
- To compare alternatives we need to estimate the value of all the actions from each state, not just the only we currently favor.

## Maintaining exploration

- What we described above is a general problem of *maintaining exploration* as in the bandits problem.
- For policy evaluation to work for action values, we must assure continual exploration.
- One way to do this is by specifying that episodes start in a state-action pair, and that every pair has a nonzero probability of being selected as the start.
- This guarantees that all state-action pairs will be visited an infinite number of times in the limit of an infinite number of episodes.
- This is called the assumption of *exploring starts*.

## Maintaining exploration

- Assumption of exploring starts cannon relied upon in general, particularly when learning directly from actual interaction with an environment.
- The most common alternative approach to assuring that all state-action pairs are encountered is to consider only policies that are stochastic with a nonzero probability of slecting actions in each state.
- For now, let’s retain the assumption of exploring starts to present the full Monte Carlo control method.

## Outline

## Monte Carlo Control

- Now we learn how to use Monte Carlo estimation to approximate optimal policies.
- The overall idea is to proceed according to the idea of GPI as in DP.
- Remember, in GPI, one maintains both an approximate policy and an approximate value function.
- VF is repeatedly altered to more closely approximate the value function for the current policy, and the policy is repeatedly improved with respect to the current value function.
- To start, let's consider a Monte Carlo version of classical policy iteration.
- In this method, we perform alternating complete steps of policy evaluation and policy improvement, beginning with arbitrary $\pi_0$ and ending with optimal policy and value functions:
$$\pi_0 \xrightarrow{\text{E}} q_{\pi 0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} q_{\pi 1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \ldots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} q_*,$$
- $\xrightarrow{\text{E}}$ denotes complete evaluation and $\xrightarrow{\text{I}}$ denotes complete policy improvement.

## Monte Carlo Control

- Policy evaluation is done exactly as described before: many episodes are experienced, with the approximate action-value function approaching the true function asymptotically.
- For the moment, assume we observe an infinite number of episodes and that, in addition, the episodes are generated with exploring starts.
- Under these assumptions, MC methods will compute each $q_{\pi k}$ exactly for arbitrary $\pi_k$.
- Policy improvement is done by making the policy greedy with respect to the current value function.
- Since we have an action-value function, no model is needed to construct the greedy policy.
- For any $q$, the greedy policy is the one that, for each $s \in \mathcal{S}$, deterministically chooses an action with maximal action-value:

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

## Monte Carlo Control

- Policy improvement can then be done by constructing each $\pi_{k+1}$ as the greedy policy with respect to $q_{\pi k}$.
- The policy improvement theorem then applies to $\pi_k$ and $\pi_{k+1}$ because, for all $s \in \mathcal{S}$,

$$\begin{aligned} q_{\pi k}(s, \pi_{k+1}(s)) &= q_{\pi k}(s, \arg\max_a q_{\pi k}(s,a)) \\ &= \max_a q_{\pi k}(s,a) \\ &\geq q_{\pi k}(s, \pi_k(s)) \\ &\geq v_{\pi k}(s). \end{aligned}$$

- The theorem assures us that each $\pi_{k+1}$ is uniformly better than $\pi_k$, or just as good as $\pi_k$, in which case they are both optimal.
- This, in turn, assures that the overall process converges to the optimal policy and value function. This is how MC methods can be used to find an optimal policy without knowledge of the environment dynamics.

## Monte Carlo Control

- We made two unlikely assumptions above to guarantee convergence of Monte Carlo method: exploring starts and that policy evaluation could be done with infinite number of episodes.
- To obtain a practical algorithm, we will have to remove both assumptions.
- Lets start focusing on removing the second assumption. This was also a problem in DP. In both DP and MC there are two ways to solve this issue (infinite number for convergence):
  1. Hold firm to the idea of approximating $q_{\pi k}$ in each policy evaluation.
  2. Give up trying to complete policy evaluation before returning to policy improvement.
- In 1. measurements and assumptions are made to obtain bounds on the magnitude and probability of error in the estimates (may require too many episodes to be useful in practice on any but the smallest problems).
- In 2. we give up trying complete policy evaluation before turning to policy improvement. One extreme case of this was value iteration.

## Monte Carlo Control

- For MC policy iteration, it is natural to 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.
- A simple algorithm to go along these lines, which we call *Monte Carlo ES* for Monte Carlo with Exploring Starts is given in the next slide.

## Monte Carlo ES, for estimating $\pi \approx \pi_*$ (algorithm)

1. Initialize:
   - $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$.
   - $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$.
   - $\mathit{Returns}(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$.
2. 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 $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$.
   - $G \leftarrow 0$.
   - Loop for each 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 $\mathit{Returns}(S_t, A_t)$.
     - $Q(S_t, A_t) \leftarrow$ average($\mathit{Returns}(S_t, A_t)$).
     - $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$.

## Monte Carlo ES, for estimating $\pi \approx \pi_*$

- In the previous algorithm, all the returns for each state-action pair are accumulated and averaged, irrespective of what policy was in force.
- It is easy to see that Montecarlo ES cannot converge to any suboptimal policy. If it did, the value function would eventually converge to the value function for that policy, and that would in turn cause the policy to change.
- Convergence to this optimal fixed point seems inevitable as the changes to the action-value function decrease over time, but has not yet been formally proved.
- Let's solve blackjack!

## Outline

## Monte Carlo without exploring starts

- We want to avoid the assumption of exploring starts.
- There are two ways to ensure this:
    1. On-policy methods.
    2. Off-policy methods.
- On-policy attempts to evaluate or improve the policy that is used to make decisions.
- Off-policy attempts to improve a policy different from that used to generate the data.
- Montecarlo ES method is an example of on-policy method.
- In on-policy control methods the policy is generally *soft*, meaning that $\pi(a|s) > 0 \; \forall s \in \mathcal{S}$ and $\forall a \in \mathcal{A}(S)$, but gradually shifted closer and closer to a deterministic optimal policy.
- The on-policy method in this section uses $\epsilon - greedy$ policies: choose the action that has maximal estimated action value, but with probability $\epsilon$ choose randomly another one.

## Monte Carlo without exploring starts

- In other words, all nongreedy actions are given the minimal probability of selection $\frac{\epsilon}{|\mathcal{A}(s)|}$, and the remaining bulk of the probability, $1 - \epsilon + \frac{\epsilon}{|A(s)|}$ is given to the greedy action.
- $\epsilon - \mathit{greedy}$ policies are examples of $\epsilon - \mathit{soft}$ policies, defined as policies for which $\pi(a|s) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$ for all states and actions, and $\epsilon > 0$.
- The overall idea of on-policy MC control is still that of GPI.
- We use first-visit MC methods to estimate the action-value function for the current policy.
- We can not improve policy by making it greedy wrt current value function since that would prevent further exploration of nongreedy actions.

## Monte Carlo without exploring starts

- The good news is that GPI does not require that the policy be taken all the way to a greedy policy.
- It only requires that it moves towards a greedy policy. In on-policy method we will move it only to an $\epsilon$-greedy policy.
- For any $\epsilon-$soft policy, $\pi$, any $\epsilon-$greedy policy wrt $q_\pi$ is guaranteed to be better than or equal to $\pi$.
- Let's take a look at the algorithm.

## On-policyt first-visit MC control (for $\epsilon$-soft policies)

```
1. Set algorithm parameter $\epsilon > 0$
2. Initialize:
    • $\pi$ an arbitrary $\epsilon-$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 $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$.
3. Repeat forever (for each episode):
    • Generate an episode following $\pi$: $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$.
    • $G \leftarrow 0$.
    • Loop for each episode, $t = T-1, T-2, ..., 0 :$
        • $G \leftarrow \gamma G + R_{t+1}$.
        • Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1, ..., S_{t-1}, A_{t-1}$:
        • Append $G$ to $Returns(S_t, A_t)$.
        • $Q(S_t, A_t) \leftarrow$ average($Returns(S_t, A_t)$).
        • $A^* \leftarrow \arg\max_a Q(S_t, a)$.
        • For all $a \in \mathcal{A}(S_t)$ :
```

$$\pi(a|S_t) = \begin{cases} 1 - \epsilon + \epsilon/|\mathcal{A}(\mathcal{S}_t)| & \text{if a=A}^*, \\ \epsilon/|\mathcal{A}(\mathcal{S}_t)| & \text{if a} \neq A^*. \end{cases}$$

## On-policy first-visit MC control

- That any $\epsilon$-greedy policy wrt $q_\pi$ is an improvement over any $\epsilon$-soft policy $\pi$ is assured by the policy improvement theorem.
- Let $\pi'$ be the $\epsilon$-greedy policy. The conditions of the PIT apply because $\forall s \in \mathcal{S}$:

$$\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_a \pi'(a|s) q_\pi(s,a) && (1)\\
&= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\epsilon) \max_a q_\pi(s,a) && (2)\\
&\geq \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + (1-\epsilon) \sum_a \frac{\pi(a|s) - \frac{\epsilon}{|\mathcal{A}(s)|}}{1-\epsilon} q_\pi(s,a). && (3)\\
&= \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) - \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) + \sum_a \pi(a|s) q_\pi(s,a) && (4)\\
&= v_\pi(s). && (5)
\end{aligned}$$

- Thus, by the PIT, $\pi \geq \pi'$ ($v_{\pi'}(s) \geq v_\pi(s)$ $\forall s \in \mathcal{S}$.

## On-policy first-visit MC control

- Now' let's prove that equality holds only when both $\pi'$ and $\pi$ are optimal among the $\epsilon-$soft policies (they are better than or equal to all other $\epsilon-$soft policies).
- Consider a new environment that is just like the original environment, except with the requirement that policies be $\epsilon-$soft "moved-inside" the environment.
- New environment has the same action and state set as the original and behaves as follows:

1. If in state $s$ and taking action $a$, then with probability $1-\epsilon$ the new environment behaves exactly like the old environment.
2. With probability $\epsilon$ it replicates the action at random, with equal probabilities, and then behaves like the old environment with the new, random action.
3. The best one can do in this new environment with general policies is the same as the best one could do in the original environment with $\epsilon$-soft policies.
4. Let $\tilde{v}_*$ and $\tilde{q}_*$ denote the optimal value functions for the new environment. A policy is optimal among $\epsilon-$soft policies if and only if $v_\pi = \tilde{v}_*$.

## On-policy first-visit MC control

- $\tilde{v}_*$ has to be the unique solution to the Bellman optimality equation with altered transition probabilities:

$$\begin{aligned}\tilde{v}_*(s) &= \max_a \sum_{s',r} \left[ (1-\epsilon) p(s',r|s,a) + \sum_{a'} \frac{\epsilon}{|\mathcal{A}(s)|} p(s',r|s,a') [r + \gamma \tilde{v}_*(s')] \right] \\ &= (1-\epsilon) \max_a \sum_{s',r} p(s',r|s,a)[r+\gamma \tilde{v}_*(s')] \\ &+ \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s',r} p(s',r|s,a)[r+\gamma \tilde{v}_*(s')].\end{aligned}$$

When equality holds and the $\epsilon-$soft policy $\pi$ is no longer improved, then we also know from equation 5 that:

$$\begin{aligned}v_\pi(s) &= (1-\epsilon) \max_a q_\pi(s,a) + \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s,a) \\ &= (1-\epsilon) \max_a \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')] \\ &+ \frac{\epsilon}{|\mathcal{A}(s)|} \sum_a \sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')].\end{aligned}$$

## On-policy first-visit MC control

- The equation above is exactly the one as in our 'parallel world', except for the substitution of $v_\pi$ for $\hat{v}_*$. Since $\hat{v}_*$ is unique, $v_\pi = \hat{v}_*$.
- Thus, we have shown that policy iteration forks for $\epsilon-$soft policies.
- Using the natural notion of greedy policy for $\epsilon-$soft policies, one is assured of improvement on every step, except when the best policy has been found among the $\epsilon-$soft policies.
- This independent of how the action-value functions are determined at each stage, but assumes that they are computed exactly.
- Now we only achieve the best policy among the $\epsilon$-soft policies, but we have eliminated the assumption of exploring starts.

## Outline

## Off-policy prediction via importance sampling

- Remember we have the following dilemma in learning methods: We need to seek action values conditional on subsequent *optimal* behavior.
- However, to do this, we need to behave non-optimally in order to explore all actions. How can we learn about optimal policy while behaving according to an exploratory policy?
- In the on-policy approach, we deal with this by learning action values not for the optimal policy, but for a near-optimal policy that still explores.
- There is another straightforward approach: use two policies, 1) one for which you learn about and become optimal policy, and 2) other that is exploratory just to generate behavior.
- The policy being learned about is called *target policy*. The one used to generate behavior is called the *behavior policy*.
- We say thar learning is from data ”off” the target policy, and the overall process is termed *off-policy learning*.

## Off-policy prediction via importance sampling

- Off-policy methods require additional concepts and notation, and because the data is due to a different policy, these methods are often of greater variance and slower to converge.
- Off-policy methods are more general and robust though. On-policy methods are a special case of off-policy methods when target and behavior policies are the same.
- They can be used, for instance, to learn data generated by a conventional non-learning controller, or from a human expert.

## The prediction problem

- To consider the prediction problem, we depart from fixed target and behavior policies. This is, we want to estimate $v_\pi$ or $q_\pi$, but all we have are policies from another policy $b$, with $b \neq \pi$.
- $\pi$ would be the target policy, and $b$ would be the behavior policy. These are fixed and given.
- To use episodes from $b$ to estimate values for $\pi$, we need that every action taken under $\pi$ is also, at least occasionally, taken under $b$: $\pi(a|s) >) \rightarrow b(a|s) > 0$ (asssumption of *coverage*).
- In control, the target policy is typically the deterministic greedy policy with respect to the current estimate of the action-value function.
- This policy becomes a deterministic optimal policy while the behavior policy remains stochastic and more exploratory, for example, and $\epsilon-$greedy policy.
- Let's consider first the case in which $\pi$ is unchanging and given.

## Importance Sampling

- Off-policy methods utilize *importance sampling*, which is a technique for estimating expected values under one distribution given samples from another.
- We use this by weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies, called the *importance-sampling ratio*.
- Given a starting state $S_t$, the probability of the subsequent state-action trajectory $A_t, S_{t+1}, A_{t+1}, ..., S_T$, occurring under any $\pi$ is:

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

- Where $p$ is the state-transition probability function. Therefore, the relative probability of the trajectory under the target and behavior policies is:

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} \tag{6}$$

## Importance Sampling

- The ratio $\rho_{t:T-1}$ transforms the returns to have the right expected value:
$$\mathbb{E}[\rho_{t:T-1} G_t | S_t = s] = v_\pi(s). \tag{7}$$
- Now we can provide a MC algorithm that averages returns from a batch of observed episodes following policy $b$ to estimate $v_\pi(s)$.
- It is convenient to number time steps in a way that increases across episode boundaries.
- For instance, If the first episode of the batch ends in a terminal state at time 100, then the next episode begins at $t = 101$.
- This enables us to use time-step numbers to refer to particular steps in particular episodes.
- Define the set of all time steps in which state $s$ is visited by $\mathcal{T}(s)$ (for an every-visit method).
- For a first-visit methods, $\mathcal{T}(s)$ would only include time steps that were first visits to $s$ within their episodes.

## Importance Sampling

- Let $T(t)$ denote the first time of termination following time $t$, and $G_t$ denote the return after $t$ up through $T(t)$.
- Then $\{G_t\}_{t\in\mathcal{T}(s)}$ are the returns that pertain to state $s$, and $\{\rho_{t:T(t)-1}\}_{t\in\mathcal{T}(s)}$ are the corresponding importance-sampling ratios.
- To estimate $v_\pi(s)$, we simply scale the returns by the ratios and average the results:

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}. \tag{8}$$

- When importance sampling is done in this way (sample average), it is called *ordinary importance sampling*.
- An important alternative is *weighted importance sampling*, which uses a weighted average defined as:

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\mathcal{T}(s)} \rho_{t:T(t)-1}}, \tag{9}$$

## Sample vs weighted importance sampling

- Consider the estimates of their first-vist methods after observing a single return from state $s$.
- In the weighted-average estimate, $V(s) = G_t$. Reasonable estimate, but its expectation is $v_b(s)$ rather than $v_\pi(s)$ and is biased in this sense.
- However, the first-visit version of the ordinary importance-sampling estimator is always $v_\pi(s)$ in expectation, but it can be extreme.
- Suppose the ratio were ten. O-I-S estimate would be ten times the expected return!
- Formally, the difference between those two is expressed in their biases and variances:
    1. Ordinary importance sampling is unbiased, whereas weighted importance sampling is biased.
    2. Variance of ordinary importance sampling is in general unbounded, whereas in the weighted estimator, the largest weight on any single return is one.
- In practice, a weighted estimator usually has a dramatically lower variance and is strongly preferred. However, ordinary importance sampling is easier to extend to the approximate methods.

## Example

- Suppose there is a nonterminal state $s$ and two actions, right and left.
- Right action causes a deterministic transition to termination, whereas the left action transitions, with probability 0.9 back to $s$ or, with probability 0.1, on to termination.
- The rewards are $+1$ on the latter transition and otherwise zero.
- Consider the target policy that always selects left.
- all episodes under this policy consist of some number (possibly zero) of transitions back to s followed by termination with a reward and return of $+1$.
- Thus, with $\gamma = 1$, the value of $s$ under the target policy is 1.
- Suppose we want to learn the value of this policy with one that chooses right and left with equal probabilities with off-policy data.

## Outline

## Incremental Implementation

- MC prediction methods can be implemented incrementally, on an episode-by-episode basis, using extensions of what we learned with bandit problems.
- While in bandits we averaged rewards, in MC methods we average *returns*.
- For on-policy MC methods, the same methods can be used.
- For off-policy MC methods, we need to separately consider those that use ordinary and those that use weighted importance sampling.
- For the latter, we have to form a weighted average of the returns, and a slightly different incremental algorithm is required.

## Incremental Implementation

- Suppose we have a sequence of returns $G_1, G_2, ..., G_{n-1}$, all starting in the same state and each with a corresponding weight $W_i$ (e.g., $W_i = \rho_{t_i} : T(t_i) - 1)$. We want to estimate:

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \quad n \geq 2, \tag{10}$$

- and keep it up-to-date as we obtain a single additional return $G_n$. In addition to keeping track of $V_n$, we must maintain for each state the cumulative sum $C_n$ of the weights given to the first $n$ returns.
- The update rule for $V_n$ is:

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}\left[G_n - V_n\right], \quad n \geq 1, \tag{11}$$

and

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

Where $C_0 \doteq 0$. Let's look at the following algorithm or the weighted importance sampling for the off-policy case (can be for the on-policy setting, target equal to behavior).

## Off-policy MC prediction, for estimating $Q \approx q_\pi$

- Input: an arbitrary target policy $\pi$ and initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$$Q(s,a) \in \mathbb{R} \quad \text{(arbitrarily)}$$
$$C(s,a) \leftarrow 0$$

- Loop forever (for each episode):

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

## Outline

## Off-policy Monte Carlo Control

- Off-policy MC control methods use one of the techniques presented in the preceding two sections.
- They follow the behavior policy while learning about and improving the target policy.
- These techniques require that the behavior policy has a nonzero probability of selecting all actions that might be selected by the target policy (coverage).
- To explore all possibilities, we require that the behavior policy be soft.
- Let's look at an algorithm for an off-policy MC control method, based on GPI and weighted importance sampling for $\pi_*$ and $q_*$. $b$ can be anything, but to assure convergence of $\pi$ to the optimal policy, an infinite number of returns must be obtained for each pair of state and action.

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

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

$$Q(s,a) \in \mathbb{R} \quad \text{(arbitrarily)}$$
$$C(s,a) \leftarrow 0$$
$$\pi(s) \leftarrow \arg\max_a Q(s,a) \quad \text{with ties broken consistently}$$

- Loop forever (for each episode):

1. $b \leftarrow$ any policy with coverage of $\pi$
2. Generate an episode following $b : S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$.
3. $G \leftarrow 0$, $W \leftarrow 1$.
4. Loop for each step of the episode, $t = T-1, T-2, ..., 0$, while $W \neq 0$:

$$G \leftarrow \gamma G + R_{t+1}, \quad C(S_t, A_t) \leftarrow C(S_t, A_t) + W,$$
$$Q(S_t, A_t) \leftarrow C(S_t, A_t) + \frac{W}{C(S_t, A_t)} |G - Q(S_t, A_t)|$$
$$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a) \quad \text{(with ties broken consistently).}$$

If $_t \neq \pi(S_t)$ proceed to next episode

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

## Problems with this method

- A potential problem is that this method learns only from the tails of episodes, when all of the remaining actions in the episode are greedy.
- If nongreedy actions are common, then learning will be slow, particularly for states appearing in the early portions of long episodes.
- There is no complete assessment of how serious this problem is.
- The next chapter is supposed to address at least partially this issue.