MACHINE INTELLIGENCE COMMUNITY

Summary of Reinforcement Learning: An Introduction Chapter 5: Monte Carlo Methods

Justin Chen* chenjus@bu.edu

Tyrone Hou* tyroneh@bu.edu

January 15, 2018

Introduction

- **Goals**:
  1. Estimating value functions
  2. Discovering optimal policies
- **Experience** - sample sequences of states, actions, and rewards from either trial-and-error or a model
- **Monte Carlo methods** solve tasks based on *averaging sample returns* for each state-action pair by learning from *experience*. As more returns are observed, the average converges to the expected value.
- This chapter only considers episodic tasks which assumes experience is divided into terminating episodes
- Value estimates and policies are updated at the end of episodes
- In this context, each state is a bandit problem where all the bandit problems are interrelated
  - Return after taking an action in one state depends on actions taken in later states in the same episode
  - All action selections are learning so the problem becomes nonstationary relative to the earlier states
  - Address nonstationarity with generalized policy iteration (GPI) but with Monte Carlo methods we learn value functions from sample returns with the MDP

5.1 Monte Carlo Prediction

- **Visit** - occurrence of state s in an episode
- **First-visit MC method** - estimates $v_\pi(s)$ as the average of returns after the first visit to a given state s
  - Each return is an independent, identically distributed estimate of $v_\pi(s)$ with finite variance.
  - Each average is an unbiased estimate
    - A statistic is an **unbiased estimator** of a population parameter is the mean of the sampling distribution of the statistic is equal to the value of the parameter

*These authors contributed equally to this work

  - Standard deviate of error decreases at rate of $1/\sqrt{n}$, n is the number of returns averaged

- **Every-visit MC method** - averages the returns following all visits to s
- Both first-visit MC and every-visit MC converge to $v_\pi(s)$ visits tends to infinity by the *law of large numbers*
- Backup diagram for Monte Carlo estimation starts with a state as the root node and continues with transitions during a single episode
- Defining characteristic of Monte Carlo methods: each state estimate is independent i.e. does not use bootstrapping
  - Computational cost computing return is *independent* of size of state space
  - Affords the ability to sample episodes and compute $v_\pi(s)$ without having to consider other states

First-visit MC prediction, for estimating $V \approx v_\pi$
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:
Generate an episode usingπ
For each state s appearing in the episode:
$G \leftarrow$ the return that follows the first occurrence of s
Append G to *Returns(s)*
$V(s) \leftarrow$ average(*Returns(s)*)

5.2 Monte Carlo Estimation of Action Values

- What if environment model is not available? That is, say we don't know the transition function $p(s', r|s, a)$.
- We can't simply estimate state values since we don't know how our actions affect the model. Instead, we can estimate *action values*.
- **Monte Carlo estimation of action values** - Average sample returns from each state-action pair visited in an episode.
  - **First-visit MC** - Average returns from *first* time a state is visited and an action taken
  - **Every-visit MC** - Average returns from *every* time a state is visited and an action taken
  - Both methods converge quadratically to $q_\pi(s, a)$ as number of visits to (s, a) approaches infinity
- As in the bandit problem, for MC estimation to be accurate we must *maintain exploration* for all state-action pairs
  - **Exploring starts** - Randomize initial state and action choice at start of each episode. This ensures all pairs are chosen in the limit of infinite episodes
  - This is like optimistic initialization for the bandit problem. We can't rely on this in general; we only guarantee exploration of starting conditions
  - May also be slow if episodes are very long
- As an alternative, we can only consider stochastic policies with non-zero probability of choosing each state-action pair (See Section 5.4)

5.3 Monte Carlo Control

- Hello, my good friend GPI.
- We use the *generalized policy iteration* (GPI) framework, in which we first approximate the value function of a given policy, then improve the policy for the value function. These kinds of *moving target* updates approach optimal values and policy. (See Chapter 4)
- For now, assume we have 1) exploring starts and 2) an infinite number of sample episodes
- Policy evaluation - Done using MC estimation of action values
- Policy improvement - Make the new policy π_{k+1} greedy w.r.t estimated value function $q_{\pi_k}(s,a)$ for the previous policy.

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

- The policy improvement theorem applies since for each state the greedy function always selects the action with maximum value; its actions are always at least as good as any other policy's.

$$\begin{aligned} q_{\pi_k}(s, \pi_{k+1}(s)) &= q_{\pi_k}(s, \underset{a}{argmax}\ q_{\pi_k}(s,a)) \\ &= \underset{a}{max}\ q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}(s, \pi_k(s)) \\ &\geq v_{\pi_k}(s) \end{aligned}$$

- Do we have to completely approximate q_{π_k} during each iteration of policy evaluation?
  - This increases the number of episodes needed to converge to optimal
  - Instead, in each evaluation step, we only move partially towards q_{π_k} before improving the policy
- Updating?

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$
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:
  Choose $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(S_0)$ from all (S, A) pairs. All pairs have probability > 0.
  Generate an episode starting in S_0, A_0 using π
  For each pair s, a appearing in the episode:
    $G \leftarrow$ the return that follows the first occurrence of s, a
    Append G to *Returns(s, a)*
    $Q(s,a) \leftarrow$ average(*Returns(s, a)*)
For each s in the episode:
  $\pi(s) \leftarrow \underset{a}{argmax} Q(s,a)$

- in the algorithm above, we alternate between policy evaluation and policy improvement steps on an episode-by-episode basis, doing one of each step per episode.
- returns are averaged over *all* state-action pairs regardless of the policy that generated them.
- It cannot converge to any sub-optimal policy π, since if it did, the value function would also converge to v_π during policy evaluation, and the policy improvement step would find a better policy.
- So does it converge to optimality, or not converge at all? *This is an fundamental open question of reinforcement learning.*

5.4 Monte Carlo Control without Exploring Starts

- How to avoid assuming exploring starts?
- **On-policy** - A method that evaluates or improves the current policy
  - **Soft on-policy control method** - A method in which for all $s \in \mathcal{S}$ and all $s \in \mathcal{A}(s)$, $\pi(a|s) > 0$ and converges to a deterministic optimal policy over time
  - ϵ-greedy:
    - All nongreedy actions selected with $\frac{\epsilon}{|\mathcal{A}(s)|}$ chance
      - Greedy probability $\times$ uniformly at random choosing an action
    - Greedy action selected with $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$
  - **ϵ-soft policies** - policies for which $\pi(a|s) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$ for all states and actions given $\epsilon > 0$
- **Off-policy** - A method that evaluates or improves a policy different from the current one to generate data
- On-policy Monte Carlo control uses GPI
  - Removing exploring starts assumption means the policy cannot improve by making it greedy w.r.t. the current value function which could prevent further exploration
  - GPI does not imply the policy must be fully greedy only that it employs exploitation
  - ϵ-greedy is guaranteed to be at least as good as any ϵ-soft policy which is guaranteed by the **policy improvement theorem**
  - For any $s \in \mathcal{S}$:

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

    - This only holds when both π' and π are better than or equal to all the ϵ-soft policies
  -

On-policy first-visit MC control (for ϵ-soft policies), estimates $\pi \approx \pi_*$
Initialize, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list
$\pi(a|s) \leftarrow$ an arbitrary ϵ-soft policy

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

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

5.5 Off-policy Prediction via Importance Sampling

- All learning control methods must deal with exploration v. exploitation trade-off
  - Agent must take sub-optimal actions to discover optimal actions
  - How can an agent learn the optimal policy while exploring?
- **Off-policy Learning** - Using two policies to indirectly learn a desired behavior by learning through exploratory behavior
- **Target Policy** - A policy that is learned and improved
- **Behavior Policy** - An exploratory policy used to generate behavior
- Off-policy uses the behavior policy as a means to learn the target policy
- On the other hand, on-policies are simpler and compromise by only learning a *near-optimal* policy
- Off-policy have more variance and are slower to converge, but are more powerful and general
- On-policies are a subset of off-policy where the behavior policy and the target policy are the same
- **Assumption of coverage** - If the probability of selecting an action under the target policy, π, is greater than zero $\pi(a|s) > 0$, then the probability of selecting an action with the behavior policy, b, must also be greater than zero, but not necessarily the same probability $b(a|s)$ and it must behave stochastically for states not encountered by π. However, π may behave deterministically when it encounters states that b does not encounter.
- Target policy is usually deterministic greedy with respect to the current action-value function estimate and continues deterministic optimal while the behavior policy remains stochastic and exploratory
- **Importance Sampling** - Estimating expected values under one distribution given samples from another
- **Importance-sampling Ratio** - Weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies

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

Importance-sampling Ratio:

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

- Trajectory probabilities, $p(S_{k+1}|S_k, A_k)$ depend on knowing MDP transition probabilities are unknown in practice
- Importance-sampling ratio only depends on π, b, and trajectory

Off-policy Monte Carlo using batched episodes with behavior policy b and target policy π

- Notation:
  - $\mathcal{T}(s)$ - Set of all time steps in which state s was visited. For first-visit methods, this would only include time steps for first visits to s in the episode.

  - T(t) - First time of termination following time t
  - G_t - Return from $(t, T(t)]$
  - $\{G_t\}_{t \in \mathcal{T}(s)}$ - Returns for state s
  - $\{\rho_{t:T(t)-1}\}_{t \in \mathcal{T}(s)}$ - Importance-sampling ratios for state s

- **Ordinary Importance Sampling** - Value function, $v_\pi(s)$, estimated as the average of returns scaled by the importance-sampling ratios

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

$$\begin{aligned} \rho_{t:T-1} G_t &= \rho_{t:T-1}(R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-t-1} R_T) \\ &= \rho_{t:T-1} R_{t+1} + \gamma \rho_{t:T-1} R_{t+2} + \ldots + \gamma^{T-t-1} \rho_{t:T-1} R_T \end{aligned}$$

  - Unbiased
  - Variance is unbounded because of the ratios can be unbounded

- **Weighted Importance Sampling** - Value function, $v_\pi(s)$, estimated as the weighted average of returns scaled by the importance-sampling ratios

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

$$V(s) \doteq 0 \text{ if denominator is } 0$$

  - Biased which asymptotically converges to zero
  - Largest weight on any individual return is bounded to one, which bounds the variance of the estimator
  - Variance converges to zero even if the variance of the ratios is infinite if the returns are bounded

5.6 Incremental Implementation

- Monte Carlo prediction can be implemented *incrementally* to reduce memory and computation time costs.

- For *on-policy* methods, we can use similar methods to those implemented for bandit algorithms (Chapter 2), except we average *returns* over state-action trajectories instead of *rewards* for taking actions.

$$\begin{aligned} Q_{n+1} &= \frac{1}{n} \sum_{i=1}^{n} R_i \\ &= Q_{n+1} + \frac{1}{n}[R_n - Q_n] \end{aligned}$$

- For *off-policy* methods, we distinguish between methods that use *ordinary* importance sampling and *weighted* importance sampling

  - Assume we have a sequence of $n-1$ returns $G_1, G_2 \ldots G_{n-1}$ for the same state, and let $W_k = \rho_{t:T(t)-1}$ be the importance sampling ratio for return G_k.
  - *with ordinary importance sampling* - Since this is a *simple average*, we apply the same method from Chapter 2, except replacing rewards with scaled returns. Then the value estimate V_n is

$$\begin{aligned} V_n &\doteq \frac{1}{n} \sum_{i=1}^{n} W_k G_k \\ &= V_n + \frac{1}{n}[W_k G_k - V_n] \end{aligned}$$

  - *with weighted importance sampling* - Since we have a *weighted average* of returns, we use a different algorithm that also keeps track of a cumulative sum C_n of weights for the first n returns.

$$\begin{aligned} V_{n+1} &\doteq \frac{\sum_{k=1}^{n} W_k G_k}{\sum_{k=1}^{n} W_k} \\ &\doteq V_n + \frac{W_n}{C_n}\Big[G_n - V_n\Big] \\ C_{n+1} &\doteq C_n + W_{n+1} \end{aligned}$$

- In general, we wish to exploit the recursive nature of importance sampling ratios and returns generated in an episode in order to speed up value estimation calculations.

Off-policy MC prediction, for estimating $Q \approx_a \pi$
Input: an arbitrary target policy π

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$

Repeat forever:
$b \leftarrow$ any policy with coverage of π
Generate an episode using b:
$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
$G \leftarrow 0$
$W \leftarrow 1$
For $t = T-1, T-2, \ldots$ down to 0:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
$W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
If W=0 then exit For loop

5.7 Off-policy Monte Carlo Control

- Off-policy methods separates concerns of policy value estimation and using the policy value estimate for control
  - **Behavior Policy** - Soft policy for generating behavior (explore all combinations of states and actions with nonzero probability)
    - Need an infinite number of returns for each pair of state and action for π to converge to π_* which is guaranteed if the behavior policy is ϵ-soft
    - Overly exploring slows down learning especially in earlier states for longer episodes, however this can be addressed with temporal-difference learning and setting $\gamma < 1$
  - **Target Policy** - Policy that evaluated and improved
- Advantage of separating the two functions is that the target can be *deterministic* while the behavior can continue to be *stochastic* to explore
- However, this method could problematically only learn from the tails of episodes if all other actions in the episode are greedy

Off-policy MC control, for estimating $\pi \approx \pi_*$
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \underset{a}{argmax} Q(S_t, a)$ (with ties broken consistently)

Repeat forever:
$b \leftarrow$ any soft policy
Generate an episode using b:
$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T, S_T$
$G \leftarrow 0$
$W \leftarrow 1$
For $t = T-1, T-2, \ldots$ down to 0:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \underset{a}{argmax}\ Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit For loop
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

5.8 *Discounting-aware Importance Sampling

- If $\gamma < 1$ and there are many time steps, only the most immediate subsequent time steps are relevant and time steps further into the future are independent of the return and expected value of 1 and contribute to variance.
- **Discounting** - Degree of partial information
- **Flat Partial Returns** - A return without discounting for a subset of contiguous time steps

$$\bar{G}_{t:h} \doteq R_{t+1} + R_{t+2} + \ldots + R_h, \quad 0 \leq t < h \leq T,$$

- T is the episode termination time step
- **Horizon** - Last time step of flat partial returns, denoted h
- Full return G_t:

$$\begin{aligned} G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T \\ &= (1-\gamma) R_{t+1} \\ &\quad + (1-\gamma)\gamma(R_{t+1} + R_{t+2}) \\ &\quad + (1-\gamma)\gamma^2(R_{t+1} + R_{t+2} + R_{t+3}) \\ &\quad \vdots \\ &\quad + (1-\gamma)\gamma^{T-t-2}(R_{t+1} + R_{t+2} + \ldots + R_{T-1}) \\ &\quad + \gamma^{T-t-1}(R_{t+1} + R_{t+2} + \ldots + R_T) \\ &= (1-\gamma) \sum_{h=t+1}^{T-1} \gamma^{h-t-1} \bar{G}_{t:h} + \gamma^{T-t-1} \bar{G}_{t:T} \end{aligned}$$

- Note that this is actually an *approximation* equal in the limit as T goes to infinity. Consider the part of G_t that is the sum of all terms with R_{t+i} for a given i. That is,

$$\begin{aligned} &(1-\gamma)(\gamma^{i-1} R_{t+i} + \gamma^i R_{t+i} + \gamma^{i+1} R_{t+i} + \cdots + \gamma^{T-t-1} R_{t+i}) \\ =&(1-\gamma) R_{t+i} \sum_{h=t+i}^{T-1} \gamma^{h-t-1} \end{aligned}$$

When $T \to \infty$, the rightmost term is an infinite geometric sum, so the entire sum of R_{t+i} terms becomes

$$(1-\gamma)\frac{\gamma^{i-1}}{1-\gamma} R_{t+i} = \gamma^{i-1} R_{t+i}$$

- **Discounting-aware Ordinary Importance-sampling estimator**

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \left((1-\gamma)\sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1}\rho_{t:h-1}\bar{G}_{t:h} + \gamma^{T(t)-t-1}\rho_{t:T(t)-1}\bar{G}_{t:T(t)}\right)}{|\mathcal{T}(s)|}$$

- **Discounting-aware Weighted Importance-sampling estimator**

$$V(s) \doteq \frac{\sum_{t\in\mathcal{T}(s)} \left((1-\gamma)\sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1}\rho_{t:h-1}\bar{G}_{t:h} + \gamma^{T(t)-t-1}\rho_{t:T(t)-1}\bar{G}_{t:T(t)}\right)}{\sum_{t\in\mathcal{T}(s)} \left((1-\gamma)\sum_{h=t+1}^{T(t)-1} \gamma^{h-t-1}\rho_{t:h-1} + \gamma^{T(t)-t-1}\rho_{t:T(t)-1}\right)}$$

5.9 *Per-reward Importance Sampling

- Another way to reduce variance in estimator is by exploiting the reward structure of the summation ratio return product $\rho_{t:T-1}G_t$:.

$$\rho_{t:T-1}G_t = \rho_{t:T-1}R_{t+1} + \gamma\rho_{t:T-1}R_{t+2} + \cdots + \gamma^{T-t-1}\rho_{t:T-1}R_T$$

- For R_{t+1},

$$\rho_{t:T-1}R_{t+1} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}\frac{\pi(A_{t+1}|S_{t+1})}{b(A_{t+1}|S_{t+1})}\cdots\frac{\pi(A_{T-1}|S_{T-1})}{b(A_{T-1}|S_{T-1})}R_{t+1}$$

- Only the first term $\frac{\pi(A_t|S_t)}{b(A_t|S_t)}$ and last reward R_{t+1} are correlated. All other terms are independent random variables with expected value one.
- $\implies \mathbb{E}[\rho_{t:T-1}R_{t+k}] = \mathbb{E}[\rho_{t:t+k-1}R_{t+k}]$ for the kth term in $\rho_{t:T-1}G_t$.
- Then $\mathbb{E}[\rho_{t:T-1}G_t] = \mathbb{E}[\tilde{G}_t]$ where

$$\tilde{G}_t = \rho_{t:t}R_{t+1} + \gamma\rho_{t:t+1}R_{t+1} + \gamma^2\rho t : t + 2R_{t+3} + \ldots + \gamma^{T-t-1}\rho_{t:T-1}R_T$$

- Since the expected values are equal, we can write the **ordinary-importance sampling estimator** using per-reward importance sampling as

$$V(s) \doteq \sum_{t\in\mathcal{J}(s)} \frac{\tilde{G}_t}{|\mathcal{J}(s)|} \tag{1}$$

5.10 Summary

- in contrast to DP and expected updates, Monte Carlo methods learn optimal value functions and policies using sample experience.
- Advantages:
  1. Can learn from direct environment interaction with no model
  2. Can be used with simulation or *sample* models that don't have access to full probability distribution of rewards and next states
  3. Can be efficiently focused to explore relevant subsections of the state space (no need to sweep the whole space)
  4. Less harmed by violation of Markov property (because *they don't bootstrap*)
- Follows generalized policy iteration, where expected values are estimated using averaged returns
- Exploration can be maintained with
  - *on-policy* methods that learn a policy sacrificing some optimality in order to continually explore
  - *off-policy* methods that learn a deterministic optimal **target policy** based on experience from another, more exploratory **behavior policy**.
- Off-policy methods almost always use *importance sampling* (ordinary or weighted) to take into account differences in action selection between target and behavior policies.

References

Sutton, Richard S., and Andrew G. Barto. “Monte Carlo Methods.” Reinforcement Learning: An Introduction, The MIT Press, 2018, pp. 75–96.