Reinforcement Learning

Monte Carlo methods

Jordi Casas Roma

jordi.casas.roma@uab.cat

September 12, 2025

UAB
Universitat Autònoma de Barcelona

Table of Contents

Table of Contents

Preliminaries

Main idea

Informal definition of **Monte Carlo methods** (MC):

Informal definition

Monte Carlo methods are ways to solve the reinforcement learning problem using a very simple idea:

- the average is a good estimator of the expected value.

Therefore, they are based on estimating the value functions $v_\pi(s)$ or $q_\pi(s, a)$ from sample averages of the return of each state (or action-state pair).

Preliminaries

Main idea

The differences with **Dynamic Programming** (DP), where complete knowledge of the dynamics of the MDP was assumed, are:

- Learning from real experience does not require "a priori" knowledge of the dynamics of the environment to achieve optimal behaviours (policies).
- Although we need a model of the environment to generate the transitions between states, it is not necessary to know the exact expression of the probability distributions $p(r, s'|s, a)$ that characterize the dynamics of the MDP to generate these transitions.

Preliminaries

Main idea

We define MC methods only for **episodic tasks**:

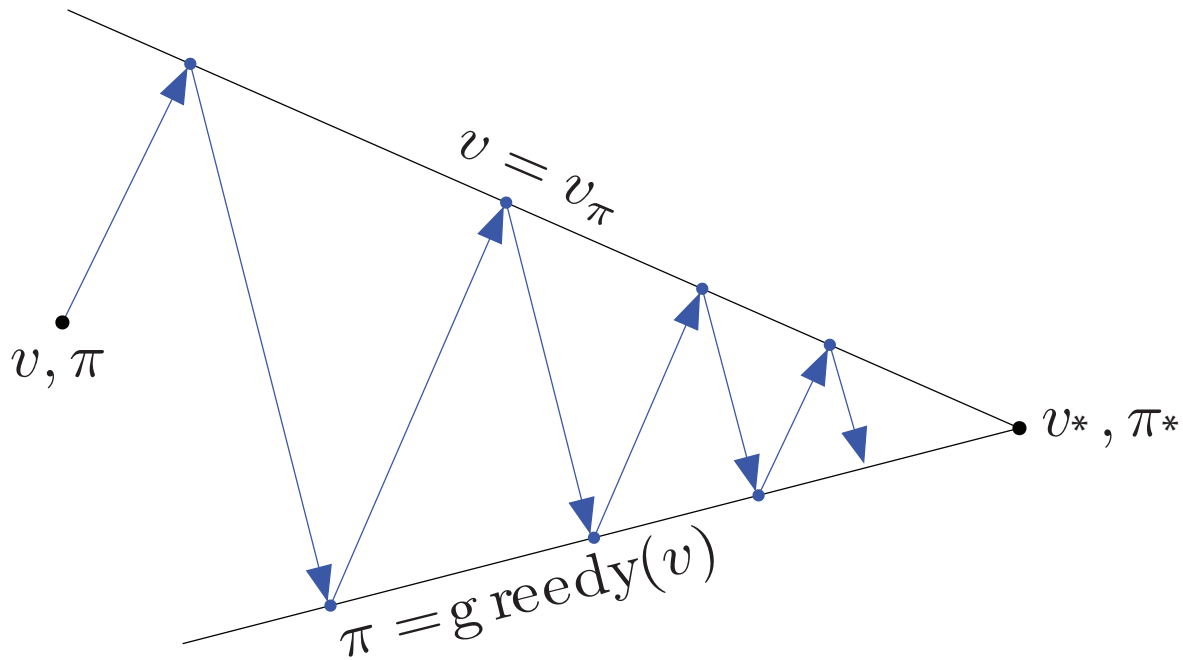
- The experience is divided into episodes.
- All episodes end eventually, no matter what actions are selected.

1. Therefore, the value function and the policy are updated when an episode ends.
2. In this way, MC methods can be considered incremental, but only in the sense of episode by episode (not in step by step, like it occurs in dynamic programming).

Preliminaries

Main idea

We adopt the concept of **General Policy Iteration (GPI)** seen in the Dynamic Programming lesson.



(Sutton & Barto, 2018)

Preliminaries

Main idea

In this way, throughout this lesson, we study the following concepts:

- **Prediction problem**:
  - How to obtain $v_\pi(s)$ and $q_\pi(s, a)$ for a given arbitrary policy.
- **Policy improvement process**.
- **Control problem** and its solution using GPI (General Policy Iteration).

$$\pi_0 \xrightarrow{E} v_{\pi 0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi 1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} v_{\pi_+}$$

Table of Contents

Prediction
Introduction

What is the objective of **prediction methods**?

Prediction

Given an **arbitrary policy**, obtain the **state** and **action value functions**.

I.e.:

$$v_\pi(s) \text{ and } q_\pi(s,a) \leftarrow \pi$$

Table of Contents

Prediction

State Value Function

The **value function of a state** $v_\pi(s)$ is the expected return from that state:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \tag{1}$$

Therefore, if we want to **estimate the value function from sample sequences** of states, actions and rewards, a procedure can be:

- Generate many episodes under a given policy π:
  $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, A_{T-1}, R_T, S_T$
- Extract those tuples that contain the state s, obtain the return G_t for each episode, and average to obtain an approximation of the expected value.

Prediction

State Value Function

This procedure can be carried out from **two approaches**:

- In the **First-visit MC method**, $v_\pi(s)$ is estimated as the average of the first-visit returns to the state s.
  - Only those first-visit returns from state s (for each episode) are averaged.
- In the **Every-visit MC method**, $v_\pi(s)$ is estimated as the average of the returns of all visits to state s.
  - In this case, if state s is visited multiple times in a single episode, the returns for each visit are averaged.

Prediction

First-visit MC method

First visit MC prediction method for $v_\pi(s)$.

In the first time step t that state s is visited in an episode, we must:

- Increment the counter $N(s) \leftarrow N(s) + 1$.
- Increase total return $Returns(s) \leftarrow Returns(s) + G_t$.
- The value of $v_\pi(s)$ is estimated by the average return $V(s) = Returns(s)/N(s)$.
- We can expect that $V(s) \rightarrow v(s)$ as $N(s) \rightarrow \infty$.

Prediction

First-visit MC method

Algorithm 1 First-visit MC prediction method to compute $v_\pi(s)$

Require: Init $N(s) = 0, \forall s \in S$
Require: Init $return\_sum(s) = 0, \forall s \; inS$
Require: Init $V(s) = 0, \forall s \in S$

for $i = 1$ to num_episodes **do**
  Generate an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \cdots, A_{T-1}, R_T, S_T$ using π
  for $t = 0$ to $T - 1$ **do**
    if S_t is a first visit **then**
      $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
      $N(S_t) = N(S_t) + 1$
      $return\_sum(S_t) = return\_sum(S_t) + G_t$
      $V(S_t) = return\_sum(S_t)/N(S_t)$
    end if
  end for
end for
return V

Prediction

Every-visit MC method

Algorithm 2 Every-visit MC prediction method to compute $v_\pi(s)$

Require: Init $N(s) = 0, \forall s \in S$
Require: Init $return\_sum(s) = 0, \forall s \; in S$
Require: Init $V(s) = 0, \forall s \in S$

for $i = 1$ to num_episodes **do**
  Generate an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \cdots, A_{T-1}, R_T, S_T$ using π
  for $t = 0$ to $T - 1$ **do**
    $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $N(S_t) = N(S_t) + 1$
    $return\_sum(S_t) = return\_sum(S_t) + G_t$
    $V(S_t) = return\_sum(S_t)/N(S_t)$
  end for
end for
return V

Table of Contents

Prediction
Action Value Function

Action Value Function:

- The **policy evaluation problem** for the action value function is to estimate $q_\pi(s, a)$ under policy π.
- The method is analogous to that explained for the state value function $v_\pi(s)$ with the difference that, instead of averaging the returns of each state in different episodes, now averages must be made for each state-pair action.

Prediction

Action Value Function

Similarly, this procedure can be carried out from **two approaches**:

- In the **First-visit MC method**, $q_\pi(s, a)$ is estimated as the average of the first-visit returns to the state s and action a.
  - Only those first-visit returns from state s and action a (for each episode) are averaged.
- In the **Every-visit MC method**, $q_\pi(s, a)$ is estimated as the average of the returns of all visits to state s and action a.
  - In this case, if state s and action a is visited multiple times in a single episode, the returns for each visit are averaged.

Prediction
Action Value Function

Problems related to the Action Value Function:

- The main problem with evaluating a policy π to estimate $q_\pi(s, a)$ is that many state-action pairs will not be visited.
- Specifically, if π is deterministic, we will only get one action for each state.
- If we do not have returns to average, the MC estimation of $q_\pi(s, a)$ for non-selected actions will not improve with experience, as they are never selected.
- This problem is relevant, given that the purpose of estimating $q_\pi(s, a)$ is to help us choose between the various actions of each state and estimate optimal policies.

Prediction
Action Value Function

There are mainly two ways to **deal with this problem**:

1. Specify that each episode begins in a given state-action pair and guarantee that each pair has a non-zero probability of being selected at the beginning of an episode.
   - This method is called "exploring starts".
   - The problem is that it is not easy to implement in real environments.
2. Only use random policies where $\pi(a|s) \neq 0$, $\forall(s, a)$.

Prediction

First-visit MC method

Algorithm 3 First-visit MC prediction method to compute $q_\pi(s, a)$

Require: Init $N(s, a) = 0, \forall s \in S, a \in A$
Require: Init $return\_sum(s, a) = 0, \forall s \; inS, a \in A$
Require: Init $Q(s, a) = 0, \forall s \in S, a \in A$

for $i = 1$ to num_episodes **do**
  Generate an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \cdots, A_{T-1}, R_T, S_T$ using π
  for $t = 0$ to $T - 1$ **do**
    if (S_t, A_t) is a first visit **then**
      $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
      $N(S_t, A_t) = N(S_t, A_t) + 1$
      $return\_sum(S_t, A_t) = return\_sum(S_t, A_t) + G_t$
      $Q(S_t, A_t) = return\_sum(S_t, A_t)/N(S_t, A_t)$
    end if
  end for
end for
return Q

Prediction

Every-visit MC method

Algorithm 4 Every-visit MC prediction method to compute $q_\pi(s, a)$

Require: Init $N(s, a) = 0, \forall s \in S, a \in A$
Require: Init $return\_sum(s, a) = 0, \forall s \ inS, a \in A$
Require: Init $Q(s, a) = 0, \forall s \in S, a \in A$

for $i = 1$ to num_episodes **do**
  Generate an episode: $S_0, A_0, R_1, S_1, A_1, R_2, \cdots, A_{T-1}, R_T, S_T$ using π
  for $t = 0$ to $T - 1$ **do**
    $G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $N(S_t, A_t) = N(S_t, A_t) + 1$
    $return\_sum(S_t, A_t) = return\_sum(S_t, A_t) + G_t$
    $Q(S_t, A_t) = return\_sum(S_t, A_t)/N(S_t, A_t)$
  end for
end for
return Q

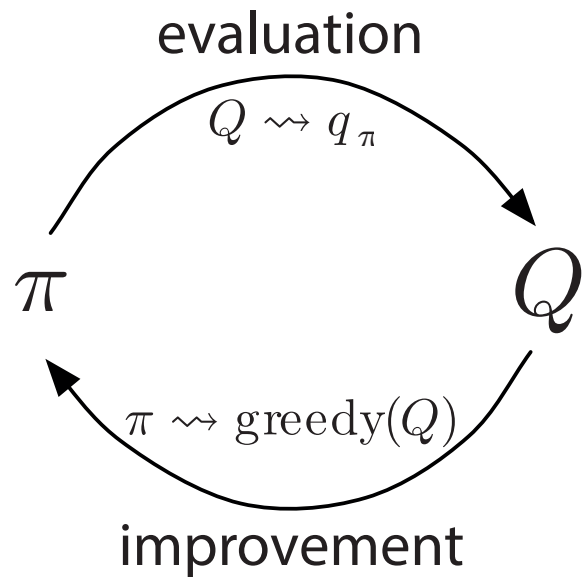
Table of Contents

Control
Introduction

What is the objective of **control methods**?

Control

Given an **arbitrary state** or **action value functions**, get the policy that optimizes them using GPI (General Policy Iteration).



Control
Introduction

General Policy Iteration (GPI):

- The GPI maintains an estimate of the policy and value function.
- They are updated alternately, so that in each iteration, the value function is approximated following the selected policy (**evaluation phase**), and the policy is improved based on the new estimated value function (**improvement phase**).
- These 2 phases alternate, so that both gradually evolve towards their optimal value:

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

- $\xrightarrow{E}$ indicates an **evaluation** of the policy and,
- $\xrightarrow{I}$ indicates an **improvement** of the policy.

Control
Evaluation phase

Evaluation phase:

- The **evaluation phase** is carried out as we have explained in the previous section.
- If the number of episodes tends to $+\infty$ and it is guaranteed that all state-action pairs will be visited (for example using the method of initial explorations),
  - it can be stated that the Monte Carlo estimates tend to q_{π_k} for any π_k.

Control

Improvement phase

Improvement phase:

- The **improvement phase** is implemented by applying a **greedy** policy from the current action value function.
- That is to say:

$$\pi_{k+1}(s) = \arg\max_{a} q_{\pi_k}(s, a) \tag{2}$$

Table of Contents

Control

Control using Monte Carlo methods with "exploring starts"

Control using Monte Carlo methods with "exploring starts":

- **Monte Carlo method with "exploring starts"** is based on forcing each episode to start in a certain state-action pair, and guaranteeing that each state-action pair probability > 0 of being selected at the beginning of an episode.

Control

Control using Monte Carlo methods with "exploring starts"

Algorithm 5 Control using Monte Carlo methods with "exploring starts" to estimate $\pi \approx \pi_*$ (first-visit)

Require: $\pi(s) \in \mathcal{A}(s)$ randomly $\forall s \in \mathcal{S}$
Require: $Q(s, a) \in \mathbb{R}$ randomly $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $Returns(s, a) \leftarrow$ empty list, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

for $i = 1$ to num_episodes **do**
  Select $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(s)$ randomly, such that the probability of each possible pair > 0
  generate an episode from S_0, A_0 following π: $S_0, A_0, R_1, \ldots, R_T, S_T$
  $G \leftarrow 0$
  for each step, $t = T - 1, T - 2, \ldots, 0$ **do**
    $G \leftarrow R_{t+1} + \gamma G$
    if S_t, A_t not in $S_0, A_0, S_1, A_1, \cdots, S_{t-1}, A_{t-1}$ **then**
      Add G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
      $\pi(S_t) \leftarrow arg\max_a Q(S_t, a)$
    end if
  end for
end for
return π

Control

Control using Monte Carlo methods with "exploring starts"

Algorithm 6 Control using Monte Carlo methods with "exploring starts" to estimate $\pi \approx \pi_*$ (every-visit)

Require: $\pi(s) \in \mathcal{A}(s)$ randomly $\forall s \in \mathcal{S}$
Require: $Q(s, a) \in \mathbb{R}$ randomly $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $Returns(s, a) \leftarrow$ empty list, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

for $i = 1$ to num_episodes **do**
  Select $S_0 \in \mathcal{S}$ and $A_0 \in \mathcal{A}(s)$ randomly, such that the probability of each possible pair > 0
  generate an episode from S_0, A_0 following π: $S_0, A_0, R_1, \ldots, R_T, S_T$
  $G \leftarrow 0$
  for each step, $t = T - 1, T - 2, \ldots, 0$ **do**
    $G \leftarrow R_{t+1} + \gamma G$
    Add G to $Returns(S_t, A_t)$
    $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
    $\pi(S_t) \leftarrow arg\max_a Q(S_t, a)$
  end for
end for
return π

Control

Control using Monte Carlo methods with "exploring starts"

Problem!

The main problem with **Monte Carlo methods with "exploring starts"** is that it is not usual to have all the state-action pairs (s, a) at the beginning of an episode.

Therefore, this method can be "forced" in simulated environments, but they can rarely be applied in real-life problems.

Table of Contents

Control

Control using Monte Carlo methods without "exploring starts"

There are two different alternative strategies that also allow the exploration of the environment:

- **On-policy methods**: They evaluate or improve the policy used to make decisions.
- **Off-policy Methods**: They evaluate or improve a policy different from the one used to generate the data.

Control
On-policy methods

On-policy methods:

On-policy methods

On-policy methods try to evaluate or improve the policy that is used both to generate the data and to make decisions.

On-policy MC control for ε-soft policies

- In *on-policy* control methods, the policy to be evaluated is generally **soft**, which means that $\pi(a|s) > 0$ for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}(s)$.
- Thus, if we execute a **"large number" of episodes**, all state-action pairs will be visited.
- The ε-**greedy policies** behave like a *greedy* policy most of the time (selecting maximum values of action value function $q_\pi(s, a)$), but with probability ε select an action at random:

$$\pi(a|s) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|} & \text{if} \quad a = arg\max_a Q(s, a) \\ \frac{\varepsilon}{|\mathcal{A}(s)|} & \text{if} \quad a \neq arg\max_a Q(s, a) \end{cases} \tag{3}$$

where $|\mathcal{A}(s)|$ is the number of available actions for state s.

An example of ε-greedy (soft) policy

- Let's suppose the following parameters:
  - $\varepsilon = 0.1$
  - $|\mathcal{A}(s)| = 10$
- From the equation 3 we get:

$$\pi(a|s) = \begin{cases} 1 - 0.1 + \frac{0.1}{10} = 0.91 & \text{if} \quad a = arg\max_a Q(s,a) \\ \frac{0.1}{10} = 0.01 & \text{if} \quad a \neq arg\max_a Q(s,a) \end{cases}$$

- Thus, there is a 91% chance of selecting the best action, and only a 1% chance of selecting each of the other 9 actions.

Control

On-policy first-visit MC control for ε-soft policies

Algorithm 7 On-policy first-visit MC control for ε-soft policies

Require: $\varepsilon > 0$
Require: $\pi \leftarrow$ policy ε-soft
Require: $Q(s,a) \in \mathbb{R}$ randomly $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $Returns(s,a) \leftarrow$ empty list, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

for $i = 1$ to num_episodes **do**
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  for each step, $t = T-1, T-2, \ldots, 0$ **do**
    $G \leftarrow R_{t+1} + \gamma G$
    if S_t, A_t not in $S_0, A_0, S_1, A_1, \cdots, S_{t-1}, A_{t-1}$ **then**
      Add G to $Returns(S_t, A_t)$
      $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
      $A^* \leftarrow arg\max_a Q(S_t, a)$
      $\forall a \in \mathcal{A}(S_t)$:

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

    end if
  end for
end for
return π

Control

On-policy every-visit MC control for ε-soft policies

Algorithm 8 On-policy every-visit MC control for ε-soft policies

Require: $\varepsilon > 0$
Require: $\pi \leftarrow$ policy ε-soft
Require: $Q(s,a) \in \mathbb{R}$ randomly $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $Returns(s,a) \leftarrow$ empty list, $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$

1: **for** $i = 1$ to num_episodes **do**
2: Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, A_{T-1}, R_T, S_T$
3: $G \leftarrow 0$
4: **for** each step, $t = T-1, T-2, \ldots, 0$ **do**
5: $G \leftarrow R_{t+1} + \gamma G$
6: Add G to $Returns(S_t, A_t)$
7: $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
8: $A^* \leftarrow arg\max_a Q(S_t, a)$
9: $\forall a \in \mathcal{A}(S_t)$:

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

10: **end for**
11: **end for**
12: **return** π

Control

On-policy MC control for ε-soft policies

On-policy MC control for ε-soft policies:

- The algorithm follows the **GPI strategy**: it uses the first-visit or every-visit MC method to estimate the action value function for the current policy (**evaluation phase**), but instead of using a *greedy* method for the improvement (which would prevent exploration by always selecting the actions with maximum $q_{\pi_k}(s, a)$), the ε-*greedy* method is used (**improvement phase**).
- This method converges to the optimal policy, since GPI does not require that the policy is *greedy*-selected in each iteration, simply that it tends to greedy policies.
- However, if we want to guarantee convergence to the optimal policy (and reduce the variance), we need to reduce exploration, so a common tactic is to progressively reduce the value of ε.

On-policy methods:

Important characteristic of on-policy methods

On-policy methods addressed the exploration versus exploitation dilemma by substituting the use of optimal policies with quasi-optimal policies that facilitate the exploration process (such as the ε-*greedy* policies).

Off-policy methods:

Off-policy methods

Off-policy methods are based on using **two policies**:

- one that is improved and becomes the optimal policy, and
- another that is more exploratory and is used to generate the data.

Control
Off-policy methods

Off-policy methods:

Therefore, there are two policies:

- **Target policy** ($\pi(s|a)$)
  - It is the policy that is learned about and that eventually becomes a deterministic optimal policy.
- **Behaviour policy** ($b(s|a)$)
  - It is the policy used to generate behaviour, to generate the data. It is usually a more exploratory stochastic policy (for example, a ε-*greedy* policy).

Control
Off-policy methods

Importance sampling:

1. The goal is to estimate q_π or v_π from a sequence of episodes generated by a policy $b \neq \pi$.
2. In order to estimate $\pi(a|s)$ from episodes generated by $b(a|s)$, the so-called coverage assumption must be satisfied:
   - 2.1 According to this assumption, we require that all actions taken under policy π are at least occasionally taken under policy b.
   - 2.2 That is, $\pi(a|s) > 0$ implies $b(a|s) > 0$ for all $s \in S$ and $a \in A(s)$.

Importance sampling:

Importance sampling

Importance sampling is a technique used to estimate expected values of a probability distribution by using samples from a different distribution.

Importance sampling:

If we start from an initial state S_t, the probability that the subsequent trajectory of state-action pairs $A_t, S_{t+1}, A_{t+1}, \ldots, S_T$ occurs under policy π is calculated as:

$$\begin{aligned} Pr\{A_t, S_{t+1}, A_{t+1}, \cdots, S_T | S_t, A_{t:T-1} \sim \pi\} = \\ = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) = \\ = \prod_{k=1}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k) \end{aligned} \tag{4}$$

Importance sampling:

Similarly, the probability of the same subsequent trajectory occurs under policy b is calculated as:

$$\begin{aligned} Pr\{A_t, S_{t+1}, A_{t+1}, \cdots, S_T | S_t, A_{t:T-1} \sim b\} = \\ = b(A_t|S_t)p(S_{t+1}|S_t, A_t)b(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) = \\ = \prod_{k=1}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k) \end{aligned} \tag{5}$$

Control
Off-policy methods

Importance sampling:

Based on equations 4 and 5, the importance sampling ratio is defined as the relative probability of the trajectory under the behaviour and target policies:

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

- $\pi(A_k|S_k)$ is the prob of taking action A_k in state S_k under policy π,
- $b(A_k|S_k)$ is the prob of taking action A_k in state S_k under policy b.

This ratio ρ_t represents the importance sampling weight, reflecting how likely the trajectory is under the target policy π relative to the behavior policy b. It adjusts for the difference in policies when estimating values using samples generated by a different policy.

Control
Off-policy methods

Importance sampling:

Since the goal of prediction is to estimate the expected return under the target policy π, but the returns G_t we have were obtained under the behaviour policy b, averaging these returns would give us an estimate of $v_b(s)$, because in this case:

$$\mathbb{E}[G_t|S_t = s] = v_b(s) \tag{7}$$

Control
Off-policy methods

Importance sampling:

If we multiply the return G_t by the importance sampling ratio ρ_t in the previous equation, we obtain the state value function under the target policy π:

$$v_\pi(s) = \mathbb{E}[\rho_t G_t | S_t = s] \tag{8}$$

1. G_t is the return starting from time step t,
2. ρ_t (sometimes called $\rho_{t:T-1}$) is the cumulative importance sampling ratio from time step t to $T-1$.

By weighting the returns with the importance sampling ratio, we correct for the difference between the two policies and obtain an unbiased estimate of the value function $v_\pi(s)$.

Control
Off-policy methods

Importance sampling: Ordinary importance sampling

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_t G_t}{|\mathcal{T}(s)|} \tag{9}$$

- $\mathcal{T}(s)$: The **set of all time steps** at which the state s is visited across all episodes. This set is calculated differently depending on whether we are using first-visit or every-visit Monte Carlo (MC) methods.
- G_t: The **return** (cumulative reward) calculated starting from time step t until the time of episode termination, $T - 1$.
- $\{G_t\}_{t \in \mathcal{T}(s)}$: The **set of returns** associated with the state s.
- ρ_t: The **importance sampling ratio**.

Control
Off-policy methods

Importance sampling: Weighted importance sampling

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_t G_t}{\sum_{t \in \mathcal{T}(s)} \rho_t} \tag{10}$$

- $\mathcal{T}(s)$: The **set of all time steps** at which the state s is visited across all episodes. This set is calculated differently depending on whether we are using first-visit or every-visit Monte Carlo (MC) methods.
- G_t: The **return** (cumulative reward) calculated starting from time step t until the time of episode termination, $T - 1$.
- $\{G_t\}_{t \in \mathcal{T}(s)}$: The **set of returns** associated with the state s.
- ρ_t: The **importance sampling ratio**.

Control
Off-policy methods

Off-Policy Prediction: Incremental Implementation

- Monte Carlo prediction methods can be implemented incrementally, episode by episode.
- In the specific case of off-policy prediction using weighted importance sampling, we will see below how to perform an incremental (episode by episode) implementation.

Control
Off-policy methods

Off-Policy Prediction: Incremental Implementation

- Let's suppose we have a sequence of returns $G_1, G_2, \ldots, G_{n-1}$, all initiated from the same state s corresponding to different episodes (in the case of first-visit methods) or not (in every-visit methods), and each with a corresponding random weight W_i (for example, $W_i = \rho_{t_i}$).
- Then, the estimation that should be implemented will be:

$$V_n \doteq \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, \quad n \geq 2 \tag{11}$$

Control
Off-policy methods

Off-Policy Prediction: Incremental Implementation

- From this point n, each time a new return G_n is calculated, the update rule for V_n becomes:

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}[G_n - V_n], \ n \geq 1 \tag{12}$$

Where:

- $C_{n+1} \doteq C_n + W_{n+1}$
- $C_0 \doteq 0$

Control

Off-policy MC prediction (*policy evaluation*)

Algorithm 9 Off-policy MC prediction

Require: An arbitrary target policy π
Require: $Q(s, a) \in \mathbb{R}$ randomly $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $C(s, a) \leftarrow 0$

for $i = 1$ to num_episodes **do**
  $b \leftarrow$ any policy that covers π
  Generate an episode following b: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  for each step, $t = T-1, T-2, \ldots, 0$ AND $W \neq 0$ **do**
    $G \leftarrow R_{t+1} + \gamma G$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
  end for
end for

return Q

Control
Off-policy methods

Off-Policy Prediction: Incremental Implementation

Ordinary importance sampling vs Importance sampling weighted

In the case of **every-visit methods** (which are more commonly used in practice due to their simplicity of implementation, as they do not require certain checks), both types of sampling present a non-zero bias.

However, both biases decrease as the number of episodes increases.

Control
Off-policy methods

Off-Policy Prediction:

- The method relies on GPI and weighted importance sampling to estimate π_* and q_*.
- The target policy is a *greedy* policy with respect to Q, which is an estimate of q_π.
- Behaviour policy (b) can be any policy, but with the objective of ensuring the convergence of π to the optimal policy, policies that cause the visit of all state-action pairs (s, a) are usually used.

Control
Off-policy methods

To conclude:

It is worth noting the two main performance differences between off-policy methods and the on-policy methods:

- **Off-policy methods** have higher variance and take longer to converge compared to on-policy methods.
- On the other hand, they are more powerful and allow the introduction of external expertise, which can be incorporated through the behaviour policy $b(s|a)$.

Control

Off-policy MC control

Algorithm 10 Off-policy MC control

Require: $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $Q(s,a) \in \mathbb{R}$ randomly $\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$
Require: $C(s,a) \leftarrow 0$
Require: $\pi(s) \leftarrow arg\max_a Q(S_t, a)$

for $i = 1$ to num_episodes **do**
  $b \leftarrow$ any *soft*-policy
  Generate an episode following b: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, A_{T-1}, R_T, S_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  for each step, $t = T-1, T-2, \ldots, 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow arg\max_a Q(S_t, a)$
    if $A_t \neq \pi(S_t)$ **then**
      Exit inner *loop* (proceed to next episode)
    end if
    $W \leftarrow W \frac{1}{b(A_t|S_t)}$
  end for
end for

return π

Table of Contents

Biliography
References

Some relevant references:

1. **R. S. Sutton, A. G. Barto**. (2018). *Reinforcement Learning: An Introduction (Second edition)*. MIT Press, Cambridge, MA.
2. **M. Lapan**. (2024). *Deep Reinforcement Learning Hands-On (Third Edition)*. Packt Publishing.