RL Ch5 Monte Carlo Methods

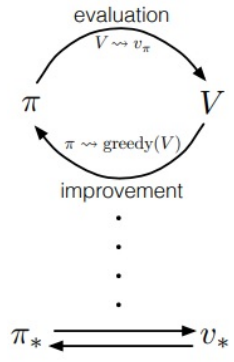
Le-Rong Hsu

June 2023

1 Review: General Policy Iteration

Policy iteration consists of two parts: policy evaluation and policy improvement. In particular, policy improvement is simply done by applying greedy algorithm on the policy:

Initialize some policy π and value function V, and assume the dynamics is known. In policy evaluation, we approximate the value $v_\pi(s)$ of the policy π by the estimate $V(s)$ for all states $s \in \mathcal{S}^+$. Next, we improve the policy by applying greedy algorithm on V to determine the best action under the state s. The iteration terminates if the policy is stable. The idea can be generalized to general policy iteration (GPI), that is, the policy is always improved with respect to the value function and the value function is always driven toward the value function for the policy.



However, we usually do not know the environment, so we need some method which is able to approximate the optimal policy without the dynamics.

2 Introduction

- Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging **sample returns**.
- In this chapter, we define Monte Carlo methods only for episodic tasks. That is, we assume experience is divided into episodes, and that all episodes eventually terminate no matter what actions are selected.
- To handle the nonstationarity, we adapt the idea of general policy iteration (GPI). We consider the prediction problem (policy evaluation) then policy improvement, and finally the control problem and solution by GPI.

Remark. The *prediction* problem is the problem estimating the value function v_π for π, and the *control* problem is the problem finding an optimal policy.

3 Monte Carlo Prediction

Suppose we wish to estimate $v_\pi(s)$, the value of a state s under policy π, given a set of episodes obtained by following π and passing through s. Each occurrence of state s in an episode is called a *visit* to s. The *first-visit MC method* estimates $v_\pi(s)$ as the average of the returns following first visit to s, whereas the *every-visit MC method* averages the returns following all visits to s.

These two methods are very similar but have slightly different theoretical properties. First-visit MC has been most widely studied, dating back to the 1940s, and is the one we focus on in this chapter. Every-visit MC will be discussed in chapters 9 and 12.

First-visit MC is shown in the box. Every-visit MC would be the same except without checking S_t having occurred earlier in the episode.

Remember that our goal is to estimate $v_\pi(s) = \mathbb{E}[G_t|S_t = s]$, where $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$. Note that we terminate at T since the task is assumed to be episodic.

Algorithm 1 First-visit MC prediction for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:

- $V(s) \in \mathbb{R}$ for all $s \in \mathcal{S}$
- $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

while true (loop forever)(loop for each episode) **do**
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for $t = T-1, T-2, ..., 1$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t does not appear in $S_0, S_1, \ldots, S_{t-1}$ **then**
      Append G to $Returns(S_t)$
      $V(S_t) \leftarrow$ average($Returns(S_t)$)
    end if
  end for
end while

We also provide the incremental implementation version:

Algorithm 2 First-visit MC prediction (incremental implementation) for estimating $V \approx v_\pi$

Input: a policy π to be evaluated
Initialize:

- $V_0(s) \in \mathbb{R}$ for all $s \in \mathcal{S}$
- $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

while true (loop forever)(loop for each episode) **do**
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for $t = T-1, T-2, ..., 1$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t does not appear in $S_0, S_1, \ldots, S_{t-1}$ **then**
      Append G to $Returns(S_t)$
      $n \leftarrow \text{len}(Returns(S_t))$
      $V(S_t) \leftarrow V(S_t) + \frac{1}{n}(G - V(S_t))$
    end if
  end for
end while

Remark. According to the definition of sample average, the n in the equation is the number of visits to the state S_t (S_t is some state s). The incremental implementation is obtained by rewriting sample average.

Both first-visit MC and every-visit MC converge to v_π as the number of visits goes to ∞. Recall that $G_t = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{T-1} R_T$. For first-visit case, the estimates $\text{avg}(Returns)(S_t) = \frac{1}{n}\sum_{i=1}^{n} G_i$ (n: the length of the list) is identically distributed and $v_\pi(S_t) = \mathbb{E}[G_t | S_t = s]$. By law of large numbers, the average of $\text{avg}(Returns)(S_t)$ converges to the expected value of G_t; that is, $v_\pi(s)$. This also gives the reason that for loop is backward. The backward for loop can avoid extra computation and memory cost compared to forward for loop.

Every-visit MC is less straightforward, but its estimates also converge quadratically to $v_\pi(s)$.

Recall that in DP methods, the dynamics of environment is assumed. All the probabilities must be calculated before DP can be applied, and such computations are often complex and error-prone. In contrast, generating the sample games is easy. The ability of Monte Carlo methods to work with sample episodes alone can be a significant advantage even when one has complete knowledge of the environment's dynamics.

Note that the computational expense of estimating $v_\pi(s)$ is independent of the number of states. This make Monte Carlo methods particularly attractive when one requires the value of one or a subset of states.

An important fact about Monte Carlo methods is that the estimates for each state are independent. The estimate for one state does not build upon the estimate of any other state, i.e. $V(S_{t+1})$ does not depend on $V(S_t)$.

4 Monte Carlo Estimation of Action Values

Model is anything the agent uses to predict the environment's response to its actions. A model-based method uses a model to plan actions before they are taken. A model-free method does not need a model; they studies the associations between actions and rewards. Thus, if a model is not available, then it's useful to estimate *action* values rather than *state* values. The first-visit MC method averages the returns following the first time in each episode that the state was visited and the action was selected.

Monte Carlo methods for action values are the same just as presented for state values, except now we talk about visits to a state-action pair s, a. A state-action pair is visited if ever the state s is visited and action a is taken in it.

However, it's possible that many state-action pairs are not selected. This forces us to consider combining *exploration* to the algorithm. For policy evaluation work for action values, we must assure continual exploration. One way to do this is randomly choose a state-action pair as the start, and each pair has nonzero probability of being selected as the start. We call this the assumption of *exploring starts*.

The assumption cannot be relied in general, particularly when learning directly from actual interaction with an environment. The most common alternative approach is to consider only policies that are stochastic with a nonzero probability of selecting all actions in each state. **We will discuss two important variants of this approach in later sections**.

5 Monte Carlo Control

We now consider how Monte Carlo estimation can be used in control, that is, to approximate optimal policies. We are now ready to consider how Monte Carlo estimation can be used in control, that is, to approximate optimal policies. The overall idea is according to **GPI**.

To begin, let's consider a Monte Carlo version of classical policy iteration. Assume that we do observe an infinite number of episodes and that the episodes are generated with exploring starts. We perform with alternating complete steps of policy evaluation and policy improvement, beginning with an arbitrary policy π and ending with the optimal policy and optimal action-value function:

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

Policy evaluation is done exactly as described in the proceeding section. On the other hand, **policy improvement is done by making the policy greedy with respect to the current value function**.

It's easy to see that Monte Carlo ES cannot converge to any suboptimal policy. If it did, then the value function would eventually converge to the value function for that policy, and that in turn would cause the policy the change. Stability is achieved only when both the policy and value function are optimal.

As stated in the algorithm, the greedy policy chooses an action with maximal action-value:

$$\pi(s) = \arg\max_a q(s, a). \tag{1}$$

We now show that the greedy policy is better or equal to the original policy: fixed $s \in \mathcal{S}$,

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

This in turn assures us that the overall process converges to the optimal policy and optimal value function.

Remember that we made two unlikely assumptions above to easily obtain the convergence for Monte Carlo method. To obtain a practical algorithm **we will have to remove both assumptions**. We postpone consideration of the first assumption until the next section.

Now we focus on the assumption that policy evaluation operates on an infinite number of episodes. One way is to apply the idea of value iteration, in which we update the value function and immediately improve the policy.

For Monte Carlo policy iteration it's 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 visited states in the episode.

Algorithm 3 Monte Carlo ES (Exploring Starts) for estimating $\pi \approx \pi^*$

Initialize:

- $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$
- $Q(s,a) \in \mathbb{R}$ for all $s \in \mathcal{S}, a \in \mathcal{A}$
- $Returns(s,a) \leftarrow$ an empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}$

while true (loop forever)(loop for each episode) **do**
  Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have prob > 0
  Generate an episode from S_0, A_0, following π:
  $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$
  Let $G \leftarrow 0$
  for $t = T-1, T-2, ..., 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t does not appear in $S_0, S_1, \ldots, S_{t-1}$ **then**
      Append 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 while

Remark. Observation: π is improved by Q, and π drives Q to approximate q_*.

For each state and action pair, it maintains a list of all returns and repeatedly calculates their mean. The algorithm can be more efficient if we apply incremental implementation. For every $Q(s,a)$, we add index so that we initialize Q with $Q_0(s,a)$ and update in the order $Q_1(s,a), Q_2(s,a)$, and so on.

We walk through the algorithm so that the readers could understand more easily. For each while loop, we have an episode: $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$. Fix $(S_t, A_t) = (s,a)$. $G_n(S_t, A_t) = G_n(s,a)$ is the n-th updated reward with $G = G_n(S_t, A_t) = G_0(s,a) = 0$. We approximate $Q(S_t, A_t) = Q(s,a)$ by incremental implementation, as stated in Chapter 2. Last, we update $\pi(S_t) = \pi(s)$ with greedy algorithm, taking the action that has the maximal action value under state s to be the policy.

Algorithm 4 Monte Carlo ES (incremental implementation) for estimating $\pi \approx \pi^*$

Initialize:

- $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$
- $Q_0(s,a) \in \mathbb{R}$ for all $s \in \mathcal{S}, a \in \mathcal{A}$
- $Returns(s,a) \leftarrow$ an empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}$

while true (loop forever)(loop for each episode) **do**
  Choose $S_0 \in \mathcal{S}, A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have prob > 0
  Generate an episode from S_0, A_0, following π:
  $S_0, A_0, R_1, S_1, A_1, R_2, ..., S_{T-1}, A_{T-1}, R_T$
  Let $G \leftarrow 0$
  for $t = T-1, T-2, ..., 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t does not appear in $S_0, S_1, \ldots, S_{t-1}$ **then**
      Append G to $Returns(S_t, A_t)$
      $n \leftarrow \text{len}(Returns(s,a))$
      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{n}(G(S_t, A_t) - Q(S_t, A_t))$
      $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
    end if
  end for
end while

6 Monte Carlo Control without Exploring Starts

How can we avoid the unlikely assumption of exploring starts? The general way to ensure that all actions are selected infinitely often is for the agent to continue to select them. There are two approaches to ensuring this: *on-policy* methods and *off-policy* methods.

On-policy methods attempt to evaluate or improve a policy that is used to make decisions, whereas off-policy methods evaluate or improve a policy different from that used to generate the data. The Monte Carlo ES method is an example of an on-policy method. In this section we show how an **on-policy** Monte Carlo control method can be designed that does not use the unrealistic assumption of exploring starts.

In on-policy control methods, the policy is generally *soft*, meaning that $\pi(s \mid a) > 0$ for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}(s)$, but generally shifted closer to a deterministic optimal policy. The on-policy method we present in this section uses ε-greedy policies with subtle modifications. All nongreedy actions are given the minimal probability of selection, $\frac{\varepsilon}{|\mathcal{A}(s)|}$, and the greedy action is given $1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|}$. The ε-greedy policies are examples of ε-*soft* policies with $\pi(a|s) \geq \frac{\varepsilon}{|\mathcal{A}(s)|}$ for all states and actions, for some $\varepsilon > 0$. Among ε-*soft* policies, ε-greedy policies are in some sense those that are closest to greedy.

The overall idea of on-policy Monte Carlo control is still that of GPI. We use

first-visit MC methods to estimate the action-value function for the current policy. However, we cannot simply improve the policy by making it greedy (w.r.t. the current value function), because that would prevent further exploration of nongreedy actions. In our on-policy method we will move it only to an ε-greedy policy. For any ε-*soft* policy π, any ε-greedy policy w.r.t. q_π is guaranteed to be better than or equal to π.

Algorithm 5 On-policy first-visit MC control for ε-*soft* policies, estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$
Initialize:

- $\pi \leftarrow$ an arbitrary ε-*soft* policy
- $Q(s, a) \in \mathbb{R}$ (arbitrarily) for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
- $Returns(s, a) \leftarrow$ empty list for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$

while true (repeat forever)(for each episode) **do**
  Generate an episode following π: $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for each step of episode, $t = T-1, T-2, ..., 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if the pair S_t, A_t does not appear in $S_0, A_0, S_1, A_1, ..., S_{t-1}, A_{t-1}$ **then**
      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 $a \in \mathcal{A}(s)$ **do**

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

      end for
    end if
  end for
end while

That any ε-greedy policy with respect to q_π is an improvement over any ε-*soft* policy π is assured by the policy improvement theorem. Let π' be the ε-greedy policy. The conditions of the policy improvement theorem apply because

for any $s \in \mathcal{S}$:

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

We explain the inequality from the second line to the third line. Note that

$$\begin{aligned}
\sum_a \pi(a \mid s) &= 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| + [|\mathcal{A}(s)| \cdot \varepsilon/|\mathcal{A}(s)| - \varepsilon/|\mathcal{A}(s)|] \\
&= 1 - \varepsilon + \varepsilon/|\mathcal{A}(s)| + \varepsilon - \varepsilon/|\mathcal{A}(s)| \\
&= 1.
\end{aligned}$$

Then,

$$\begin{aligned}
\sum_a \pi(a \mid s) - \varepsilon/|\mathcal{A}(s)| &= 1 - \varepsilon + 0 + [0 * (|\mathcal{A}(s)| - 1)] \\
&= 1 - \varepsilon.
\end{aligned}$$

Thus,

$$\sum_a \frac{\pi(a \mid s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1 - \varepsilon} = 1.$$

The sum is a weighted average with nonnegative weights summing to 1, and as such it must be less than or equal to the largest number averaged. Thus, $\pi' \geq \pi$ (i.e., $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$).

The equality can hold when both π' and π are optimal among the ε-soft policies, that is, when they are better than or equal to all other ε-soft policies. The proof is trivial and the readers can check the textbook.

7 Off-policy Prediction via Importance Sampling

How can control methods learn about the optimal policy while behaving according to an exploratory policy? The on-policy approach in the preceding section in actually a compromise: it learns action values not for the optimal policy but for a near-optimal policy that still explores. A more straightforward approach is to **use two policies**, one that is learned to become optimal policy, and one

that is more exploratory and is used to generate behavior. The policy being learned is called **target policy**, and the policy used to generated behavior is called the **behavior policy**. In this case, we say that learning is from data **"off" the target policy** and the overall process is termed *off-policy learning*.

Because the data is due to a different policy, off-policy methods are often of greater **variance** and are slower to converge. On the other hand, off-policy methods are more powerful and general. They include on-policy methods as the special case in which the target policy and behavior policy are the same.

Suppose we wish to estimate v_π or q_π, but all we have are episodes following another policy b, where $b \neq \pi$. In this case, π is the target policy and b is the behavior policy, and **both policies are considered fixed and given**.

In order to use episodes from b to estimate values for π, we require that every action taken under π is also taken under b. This is called the assumption of *coverage*, that is, we require that $\pi(a \mid s)$ implies $b(a \mid s) > 0$.

Almost all off-policy methods utilize *importance sampling*, a general technique for estimating expected values under one distribution given sample from another. The *importance-sampling ratio* is necessary here, which is the relative probability of their trajectories occurring under the target and behavior policies. Informally, it is the ratio converting the probability from behavior policy to that of the target policy.

Given a starting state S_t, the probability of the subsequent state-action trajectory, $A_t, S_{t+1}, A_{t+1}, ..., S_T$ under any policy π is

$$\begin{aligned}\mathbb{P}(A_t, S_{t+1}, A_{t+1}, ..., S_T \mid S_t, A_{t:T-1} \sim \pi) &= \pi(A_t \mid S_t)p(S_{t+1} \mid S_t, A_t) \cdot p(S_T \mid S_{T-1}, A_{T-1}) \\ &= \Pi_{k=t}^{T-1}\pi(A_k \mid S_k)p(S_{k+1} \mid S_k, A_k)\end{aligned}$$

where p is the state-transition probability. Thus the relative probability of the trajectory under the target and behavior policies is

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

The importance sampling ratio ends up depending only on the two policies and the sequence, not on the MDP.

Recall that we wish to estimate the expected returns under the target policy, but all returns G_t we have is due to b. These returns have wrong expectation $\mathbb{E}[G_t \mid S_t = s] = v_b(s)$, so we need the ration $\rho_{t:T-1}$ to transform the returns to have the right expectation:

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

Now we introduce two ways of sampling: ordinary importance sampling and weighted importance sampling. Before this, we introduce some notations:

- We number the time steps in a way that increase across episodes boundaries.
- $\mathcal{J}(s)$: the set of all time steps in which s is visited

- $T(t)$: the first time of termination following time t
- G_t: the return after t up through $T(t)$

For first-visit methods, $\mathcal{J}(s)$ contains only one time step which records the first visit to state s. For every-visit methods, the set $\{G_t\}_{t\in\mathcal{J}(s)}$ are all the returns of state s, and $\{\rho_{t:T(t)-1}\}_{t\in\mathcal{J}(s)}$ are the corresponding importance-sampling ratios.

Then, to estimate $v_\pi(s)$, we simply scale the returns by the ratios and average the results:

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

The weighted importance sampling is defined as:

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

or zero if the denominator is zero. Similarly, the weighted importance sampling for action values is:

$$Q(s,a) = \frac{\sum_{t\in\mathcal{J}(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\mathcal{J}(s)} \rho_{t:T(t)-1}}.$$

To understand these two varieties, let's consider the estimates of their **first-visit** methods **after observing a single return** from state s. In weighted-average estimate, $\rho_{t:T(t)-1}$ is cancelled, so that the estimate is equal to the observed return independent of the ratio (assuming the ratio is nonzero). This is a reasonable estimate, but its expectation is $v_b(s)$ not $v_\pi(s)$, and in the statistical sense it is biased. In contrast, the first-visit version of the ordinary importance-sampling estimator is always $v_\pi(s)$ in expectation, but it can be extreme.

Formally, the difference between the first-visit methods of the two kinds of importance sampling is expressed in their **biases** and **variances**. Ordinary importance sampling is unbiased but its variance can be unbounded. Weighted importance sampling is biased whereas the largest weight on any single return is 1. In fact, assuming bounded returns, the variance of the weighted importance-sampling estimator converges to zero even if the variance of the ratios is infinite (Precup, Sutton, Dasgupta 2001). In practice, the **weighted estimator** usually has dramatically lower variance and **is strongly recommended**.

The **every-visit methods** for ordinary and weighed importance sampling are both **biased**, though, again, the bias **falls asymptotically to zero** as the number of samples increases. In practice, every-visit methods are often preferred because they remove the need to keep track of which states have been visited and because they are much easier to extend to approximations. A complete every-visit MC algorithm for on-policy policy evaluation using weighted importance sampling is given in the next section on page 110.

Two examples are presented in the book which shows the comparison of ordinary importance sampling and weighted importance sampling. The readers can check for it.

8 Incremental Implementation

In ordinary importance sampling, the returns are scaled by the importance sampling ratio $\rho_{t:T(t)-1}$. For these methods we can again use the incremental methods of Chapter 2, but using the scaled returns in place of the rewards of that chapter:

$$V_{m+1}(s) = V_m(s) + \frac{1}{m}[\rho_{t:T(t)-1}G_m(s) - V_m(s)].$$

where m is the number of updates of $V(s)$, $G_m(s)$ is the return at time m (not cumulative return!!).

Suppose we have a sequence of returns $G_1, G-2, ..., G_{n-1}$ (not cumulative return!!), all starting in the same state and each with a corresponding random weight W_i (e.g., $W_i = \rho_{t:T(t)-1}$). For weighted importance sampling, we wish to form the estimate

$$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}, n \geq 2.$$

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

$$V_{n+1} = V_n + \frac{W_n}{C_n}[G_n - V - n], n \geq 1,$$

and

$$C_{n+1} = C_n + W_{n+1}$$

where $C_0 = 0$ and V_1 is arbitrary. The algorithm is presented here:

Algorithm 6 Off-policy MC prediction (policy evaluation) for estimating $Q \approx q_\pi$

Input: an arbitrary target policy
Initialize:

- $Q(s,a) \leftarrow \mathbb{R}$ (arbitrary)
- $C(s,a) \leftarrow 0$

while true (loop forever)(loop for each episode) **do**
  $b \leftarrow$ any policy with coverage of π
  Generate an episode following π: $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  for each step of episode, $t = T-1, T-2, ..., 0$, while $W \neq 0$: **do**
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $W \leftarrow W\frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
  end for
end while

(every visit, weighted importance sampling)

9 Off-policy Monte Carlo Control

On-policy Monte Carlo 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.

The box on the next page shows an on-policy Monte Carlo control method, based on GPI and weighted importance sampling, for estimating π_* and q_*. This is similar to value iteration since we approximate π_* and q_* at the same time.

Algorithm 7 Off-policy MC control, for estimating $\pi \approx \pi_*$

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

- $Q(s,a) \in \mathbb{R}$ arbitrarily
- $C(s,a) \leftarrow 0$
- $\pi(s) \leftarrow \arg\max_a Q(s,a)$

while true (loop forever)(loop for each episode) **do**
  $b \leftarrow$ any soft policy
  Generate an episode using b: $S_0, A_9, R_1, ..., S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  for each step of episode, $t = T-1, T-2, ..., 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $\pi(S_t) \arg\max_a Q(S_t, a)$
    If $A_t \neq \pi(S_t)$ then exit inner loop (proceed to next episode)
    $W \leftarrow W \frac{1}{b(A_t|S_t)}$
  end for
end while

The importance-sampling ratio involves $\frac{\pi(A_t|S_t)}{b(A_t|S_t)}$, but the algorithm instead involves $\frac{1}{b(A_t|S_t)}$ which is nevertheless true. If $A_t \neq \pi(S_t)$ then we exit the inner loop. If $A_t = \pi(S_t)$, then the probability of choosing A_t is 1 as the episode has already contains A_t. Thus, we are safe to say $\pi(A_t \mid S_t) = 1$ during the update of W.

10 Discounting-aware Importance Sampling

Consider the case where episodes are long and γ is significantly less than 1, **say that episodes last 100 steps and that $\gamma = 0$.** The return from time 0 will then be just $G_0 = R_1$, but its importance sampling ratio will be a product of 100 factors, $\frac{\pi(A_0|S_0)}{b(A_0|S_0)} \frac{\pi(A_1|S_1)}{b(A_1|S_1)} \cdot \frac{\pi(A_9 9|S_9 9)}{b(A_9 9|S_9 9)}$. In ordinary importance sampling, the return will be scaled by the entire product, but it is really only necessary to scale by the first factor, by $\frac{\pi(A_0|S_0)}{b(A_0|S_0)}$. The other 99 factors are irrelevant because after the first reward the return has already been determined. These later factors are all independent of the return and of expected value 1; they do not change the expected update, but they add enormously to its variance.

Let us now consider an idea for avoiding this large extraneous variance. The essence of the idea is to think of discounting as determining a probability of termination or, equivalently, a degree of partial termination. For any $\gamma \in [0, 1)$, we can think of the return G_0 as partly terminating in one step, to the degree $1-\gamma$, producing a return of just the first reward, R_1, and as partly terminating after two steps, to the degree $(1-\gamma)\gamma$, producing a return of $R_1 + R_2$, and so on. The latter degree corresponds to terminating on the second step, $1-\gamma$, and not having already terminated on the first step, γ. The partial returns here are called *flat partial returns*:

$$\bar{G}_{t:h} = R_{t+1} + R_{t+2} + ... + R_h, \text{ with } 0 \leq t \leq h \leq T,$$

where "flat" denotes the absence of discounting, and "partial" denotes that these returns do not extend all the way to termination but instead stop at h, called the *horizon*. We can rewrite conventional G_t as:

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

Now we need to scale the flat partial returns by an importance sampling ratio that is similarly truncated. As $\bar{G}_{t:h}$ only involves rewards up to a horizon h, we only need the ratio of the probabilities up to $h-1$. We define an ordinary

importance-sampling estimator as

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

and a weighted importance-sampling estimator as

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

We call these two estimators *discounting-aware* importance sampling estimators. They take into account the discount rate but have no effect if $\gamma = 1$.

Exercise 5.14 Provide the off-policy Monte Carlo control method with weighted discounting-aware importance estimator. Note that we have to convert the equation to action values first.

The problem is too hard and left as an exercise for readers.

11 Per-decision Importance Sampling

There is one more way that may be able to reduce variance even in the absence of discounting (that is, even if $\gamma = 1$). In the on-policy estimators, each term of the sum in the numerator is itself a sum:

$$\begin{aligned}\rho_{t:T-1}G_t &= \rho_{t:T-1}(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-t-1}R_T)\\ &= \rho_{t:T-1}R_{t+1} + \rho_{t:T-1}\gamma R_{t+2} + \cdot + \rho_{t:T-1}\gamma^{T-t-1}R_T\end{aligned}$$

The off-policy estimators rely on the expected values of these terms, which can be written in a simpler way. Note that each sub-term above is a product of a random reward and a random importance-sampling ratio. For example,

$$\rho_{t:T-1}R_{t+1} = \frac{\pi(A_t \mid S_t)\pi(A_{t+1} \mid S_{t+1})\pi(A_{t+2} \mid S_{t+2})\dots\pi(A_{T-1} \mid S_{T-1})}{b(A_t \mid S_t)b(A_{t+1} \mid S_{t+1})b(A_{t+2} \mid S_{t+2})\dots b(A_{T-1} \mid S_{T-1})}R_{t+1}.$$

Moreover, the expected value of all these other factors is one:

$$\mathbb{E}[\frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}] \doteq \sum_a b(A_k \mid S_k)\frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)} = \sum_a \pi(a \mid s) = 1.$$

Write $\rho_{t:t} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$. Note that the importance sampling ratio is independent of R_{t+1}. By $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$, we have

$$\begin{aligned}\mathbb{E}[\rho_{t:T-1}R_{t+1}] &= \mathbb{E}[\rho_{t:t}R_{t+1} \cdot \frac{\pi(A_{t+1} \mid S_{t+1})\pi(A_{t+2} \mid S_{t+2})\dots\pi(A_{T-1} \mid S_{T-1})}{b(A_{t+1} \mid S_{t+1})b(A_{t+2} \mid S_{t+2})\dots b(A_{T-1} \mid S_{T-1})}]\\ &= \mathbb{E}[\rho_{t:t}R_{t+1}] \cdot 1\\ &= \mathbb{E}[\rho_{t:t}R_{t+1}].\end{aligned}$$

Similarly, we can write

$$\mathbb{E}[\rho_{t:T-1}R_{t+k}] = \mathbb{E}[\rho_{t:t+k-1}R_{t+k}].$$

It follows then that the expectation of our original term can be written

$$\mathbb{E}[\rho_{t:T-1}G_t] = \mathbb{E}[\tilde{G}_t]$$

where

$$\tilde{G}_t = \rho_{t:t}R_{t+1} + \rho_{t:t+1}R_{t+2} + \cdots + \rho_{t:T-1}R_T.$$

We call this idea per-decision importance sampling.

It follows immediately that the ordinary-importance-sampling estimator using $\tilde{G}_t$:

$$V(s) = \frac{\sum_{s\in\mathcal{J}(s)} \tilde{G}_t}{|\mathcal{J}(s)|}.$$

Is there a per-decision version of weighted importance sampling? This is less clear. So far, all the estimators that have been proposed for this that we know of are not consistent (that is, they do not converge to the true value with infinite data).

12 Summary

Question: is it necessary to loop over an entire episode, i.e., "wait" until the end of the episode (since we have to calculate G, the cumulative discounted future reward)? Is it possible to update the estimate for each step in the episode? The answer is yes, it is possible. More surprisingly, it is a more powerful method and it still converges to the optimal value function. We will see this method in the next chapter.