Lecture 5: Monte Carlo Methods

Tianpei Xie

Aug 4th., 2022

Contents

1 Monte Carlo Methods

Unlike the previous chapter, here we do not assume complete knowledge of the environment. Since we do not know the precise dynamic function p, we cannot use DP to find optimal value and optimal policies. Instead, we can estimate value functions and discover optimal policies via *sampling.* **Monte Carlo methods** require only *experience* – sample sequences of states, actions, and rewards from actual or simulated interaction with an environment. It either learns from *actual* experience when *no prior knowledge* of the environments dynamics is known or learns from *simulated* experiences, when we have model that generate sample transition but not complete as dynamic.

Monte Carlo Methods are ways of solving the reinforcement learning problem based on **averaging sample returns**. To ensure that well-defined returns are available, here we define Monte Carlo methods only for **episodic tasks**. Only *on the **completion** of an episode* are value estimates and policies changed. Monte Carlo methods can thus be incremental in an ***episode-by-episode sense***, but not in a step-by-step (*online*) sense.

Compare to the Bandit problem where we simply sample and average rewards, in Monte Carlo, we sample and *average returns* for each state-action pair. Also the Monte Carlo methods learn value and policy from experience in the form of *sample episodes.*

There are three **advantages** of using Monte Carlo Methods over DP:

- **No need to have complete knowledge on dynamics**: In DP, all of the probabilities $p(s', r|s, a)$ for all s', r, s, a must be computed before DP can be applied, and such computations are often complex and error-prone. In constrasts, Monte Carlo methods have ability to learn optimal behavior directly from interaction with the environment. It only need sample episodes.
- **They can be used with simulation or *sample models*, which has low expense for computation**: Generating the sample games required by Monte Carlo methods is easy. 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*, as is the case in DP. In other words, Monte Carlo methods do not ***bootstrap*** as we defined it in the previous chapter.
- **It is easy and efficient to focus Monte Carlo methods on a small subset of the states.**: In particular, note that the computational expense of estimating the value of a single state is *independent of the number of states.* This can make Monte Carlo methods particularly attractive when *one requires the value of only one or a subset of states.* One can generate many sample episodes **starting from the states of interest**, averaging returns from *only these states*, *ignoring* all others.
- **They do not bootstrap** thus it can be used when **Markov property not hold**. That is they do not update their value estimates on the basis of the value estimates of successor states. They may be less harmed by violations of the Markov property.

Maintaining ***sufficient exploration*** is an issue in Monte Carlo control methods. It is not enough just to select the actions currently estimated to be best, because then no returns will be obtained for alternative actions, and it may never be learned that they are actually better.

```
First-visit MC prediction, for estimating $V \approx v_\pi$

Input: a policy $\pi$ to be evaluated
Initialize:
    $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
    $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):
    Generate an episode following $\pi$: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless $S_t$ appears in $S_0, S_1, \ldots, S_{t-1}$:
            Append $G$ to $Returns(S_t)$
            $V(S_t) \leftarrow$ average($Returns(S_t)$)
```

Figure 1: First visit MC prediction

2 Monte Carlo Methods for Prediction

We begin by considering Monte Carlo methods for learning the *state-value function* for a given policy. Note that by definition, the state-value function v

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi \left[G_t | S_t = s \right] \\ &\approx \frac{\sum_t^{T(s)} G_t}{N(s)} \end{aligned}$$

where $N(s) = |T(s)|$ and $T(s) = \{S_t = s\}_{t=1}^{\infty}$ or the number of epsiodes for first-visit MC.

Each occurrence of state s in an episode is called a *visit* to s. Of course, s may be visited multiple times in the same episode; let us call the first time 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, whereas the ***every-visit MC method*** averages the returns following *all visits* to s. First-visit MC is shown in procedural form in the Figure 1. Since the first visit in each epsiodes are generated *i.i.d.*, the first-visit MC is easier to analyze, whereas the every-visit MC is harder since the within-episode visits are not independent. Every-visit MC would be the same except without the check for S_t having occurred earlier in the episode. Note that the estimation goes *backwards* starting from the end of episode. This way we can apply resursive equation $G_t = R_{t+1} + \gamma G_{t+1}$.

By the law of large numbers the sequence of averages of these estimates converges to their expected value. Each average is itself an unbiased estimate, and the standard deviation of its error falls as $1/\sqrt{n}$, where n is the number of returns averaged.

3 Monte Carlo Methods for Control

If a model is not available, then it is particularly useful to estimate *action values* (the values of state-action pairs) rather than *state values*. Without a model, the state-value is not sufficient since we cannot one simply looks ahead one step and chooses whichever action leads to the best combination of reward and next state. One must explicitly estimate the value of each action in

Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

```
Initialize:
    $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
    $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
    $Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

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 step of 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 $Returns(S_t, A_t)$
            $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
            $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
```

Figure 2: MC control with exploring starts

order for the values to be useful in suggesting a policy. Thus, one of our primary goals for Monte Carlo methods is to estimate q_*. Note that

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \\ &\approx \frac{1}{N(s,a)} \sum_{t}^{T(s,a)} G_t \end{aligned}$$

where $N(s,a) = |T(s,a)|$, $T(s,a) := \{t : S_t = s, A_t = a, t = 0, .., T-1, ...\}$ for every-visit MC or the timestamps for the first visit for all episodes.

Conveniently, the same process as in previous section works for action values. The Monte Carlo methods compute average returns starting in state s, taking actions a, and thereafter following policy π. A state-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. 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. These methods converge quadratically, as before, to the true expected values as the number of visits to each stateaction pair approaches infinity.

The challenge for estimating action value via MC is that many state-action pairs may ***never be visited***, esp. for a deterministic policy π, when one will only observe returns only for one of actions from each state. With no returns to average, the Monte Carlo estimates of the other actions will not improve with experience. To compare alternatives we need to estimate the value of ***all*** *the actions from each state*, not just the one we currently favor. This is the **exploration-exploiation tradeoff**, as discussed in the multi-armed bandit.

For *policy evaluation* to work for action values, we must assure *continual exploration.* One way to do this is by specifying that the 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. We call this the assumption of ***exploring starts***. Figure 2 shows the algorithm for **Monte Carlo with exploring starts**.

Note that at the initial step of the algorithm, a random state-action pair (S_0, A_0) is generated where no policy used for generating action. Then the policy π is used to generate the episodes afterwards.

The exploring start method is not very reliable esp. when learning directly from actual interaction with an environment. In that case the starting conditions are unlikely to be so helpful. The most common alternative approach to assuring that all stateaction pairs are encountered is to consider only policies that are stochastic with a nonzero probability of selecting all actions in each state.

3.1 Monte Carlo for Policy Iteration

Given the estimation of action-value function q, we can choose the greedy policy $\pi(s) = \text{argmax}_a \, q(s, a)$. *Policy improvement* then can be done by constructing each π_{k+1} as the greedy policy with respect to q_k. This greey policy π_{k+1} is known to improve over existing policy π_k by the *policy improvement theorem.* From last section, we can use Monte Carlo Methods for policy evaluation. Then we have generalized policy iteration again

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

where $\xrightarrow{E}$ denotes a policy evaluation and $\xrightarrow{I}$ denotes a policy improvement. Unlike DP in previous chapter, Monte Carlo methods can be used to find optimal policies given only sample episodes and no other knowledge of the environments dynamics.

For Monte Carlo 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***. In Monte Carlo ES, all the returns for *each state-action pair* are accumulated and averaged, $\underline{\text{irrespective of what policy}}$ was in force when they were observed. **Stability** is achieved only when both the policy and the value function are **optimal**. 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.

The convergence guarantee for Monte Carlo Policy Iteration is based on two unlikely assumptions above.

1. the episodes have **exploring starts**
2. policy evaluation could be done with an infinite number of episodes.

To obtain a practical algorithm we will have to remove both assumptions, which will be considered later.

Removing the second assumption is relatively easy. In both DP and Monte Carlo cases there are two ways to solve the problem. One is to hold firm to the idea of approximating q_{π_k} in each policy evaluation. Measurements and assumptions are made to obtain ***bounds*** on the magnitude and *probability of error* in the estimates, and then $\underline{\textit{sufficient steps}}$ are taken during each policy evaluation to assure that these bounds are sufficiently small. This approach can probably be made completely satisfactory in the sense of guaranteeing correct convergence *up to some level of approximation.* However, it is also likely to require far **too many episodes** to be useful in practice on any but the smallest problems.

```
On-policy first-visit MC control (for $\varepsilon$-soft policies), estimates $\pi \approx \pi_*$

Algorithm parameter: small $\varepsilon > 0$
Initialize:
    $\pi \leftarrow$ an arbitrary $\varepsilon$-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 $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):
    Generate an episode following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    Loop for each step of 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 $Returns(S_t, A_t)$
            $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$
            $A^* \leftarrow \arg\max_a Q(S_t, a)$                    (with ties broken arbitrarily)
            For all $a \in \mathcal{A}(S_t)$:
                $\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$
```

Figure 3: On-policy Monte Carlo with epsilon-soft policies

The second approach is to give up trying to complete policy evaluation before returning to policy improvement. On each evaluation step we move the value function toward q_{π_k}, but we do not expect to actually get close except over many steps. One extreme form of the idea is value iteration, in which only one iteration of iterative policy evaluation is performed between each step of policy improvement.

4 On-policy and Off-policy learning without exploring starts

Without exploring starts, the only 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 **continual exploration**:

- ***On-policy* methods**. On-policy methods attempt to evaluate or improve the *policy* that is used to make decisions.
- ***Off-policy* methods**. Off-policy methods evaluate or improve a policy **different** from that used to generate the data. In this case we say that learning is from data "off" the target policy. There are
    - **target policy**, i.e. the policy being learned about, denoted as $\pi(a|s)$;
    - **behavior policy**, i.e. the policy used to generate behavior, denoted as $b(a|s)$

4.1 On-policy method with ϵ-soft policies

In on-policy control methods the policy is generally ***soft***, meaning that $\pi(a|s) > 0$ for all $s \in \mathcal{S}$ and all $a \in \mathcal{A}(s)$, but gradually shifted closer and closer to a deterministic optimal policy. For example, the ϵ-*greedy* algorithms discussed in Multi-armed Bandit algorithm are examples of ϵ-***soft*** policies, defined as policies for which $\pi(a|s) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$ for all states and actions, for some $\epsilon \geq 0$. This ensures that all states are visited infinitely in the long run.

Note that the overall idea of on-policy Monte Carlo control is still that of *GPI*. In this case, using the learned greedy policy via policy improvement would prevent further exploration of nongreedy actions. On the other hand, as long as the new policy moves *towards greedy policy*, we would see GPI continue towards optimal solution. For any ϵ-soft policy, π, any ϵ-greedy policy with respect to q_π is guaranteed to be **better** than or equal to π. Figure 3 describes the on-policy first-visit Monte Carlo control for ϵ-soft policy. Here the policy select optimal action $\text{argmax}_a\, q(s,a)$ with probability $1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}$ and other uniformly randomly with probability $\frac{\epsilon}{|\mathcal{A}(s)|}$.

Note that by nature, this algorithm will $\underline{\text{not converge to optimal policy}}$ π_*, which is a deterministic greedy policy. Instead it will **converge to an ϵ-greedy policy** that is close enough to the optimal policy with random action selection. This is because any ϵ-greedy policy π' with respect to q_π is an improvement over any ϵ-soft policy π by the policy improvement theorem.

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

Thus $v_{\pi'} \geq v_\pi$ for all s, $\pi' \geq \pi$. This *suboptimal* solution is a **compromise** for on-policy learning for the exploration-exploitation tradeoff.

ϵ-soft policy describes an environment where with probability $1 - \epsilon$ it behaves like the old environment and with probability ϵ it choose a new, uniformly random action. 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 ϵ-soft polices. The optimal state-value functions for ϵ-soft polices are like

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

This makes sure that the Generalized Policy Iterations works for ϵ-soft polices.

4.2 Off-policy prediction with importance sampling

The exploration-exploitation tradeoff states that all algorithms need to learn action values conditional on subsequent $\underline{\textit{optimal}}$ behavior, but they need to behave non-optimally in order to $\underline{\textit{explore}}$ all actions (to *find* the optimal actions), i.e. learning optimal behavior while maintaining *continual exploration.*

- On-policy methods are generally simpler and are considered first. Off-policy methods require additional concepts and notation, and because the data is due to a different policy, off-policy methods are often of greater variance and are slower to converge.
- Off-policy methods also have a variety of additional uses in applications. For example, they can often be applied to learn from data generated by a conventional non-learning controller, or from a human expert. Off-policy learning is also seen by some as key to learning multi-step predictive models of the worlds dynamics.
- On-policy methods have to make a sacrifice to the performance by choosing sub-optimal policy in order to keep exploring. Off-policy methods do not have this issue.

In this section, assume both target and behavior policy are fixed and given. We are considering the off-policy prediction problem, i.e. estimating v_π and q_π, where episodes follow policy $b \neq \pi$. Also we made the *assumption of* ***coverage***: i.e. $\text{supp}(\pi(\cdot|s)) \subseteq \text{supp}(b(\cdot|s))$. That means that every action taken under π is also taken, at least occasionally, under b; or, for action a, $\pi(a|s) > 0$ implies $b(a|s) > 0$. The behavior policy b has to be *stochastic* to provide coverage while the target policy π could be *deterministic.*

Almost all off-policy methods utilize **importance sampling**, a general technique for estimating expected values under one distribution given samples from another. We apply importance sampling to off-policy learning by weighting returns according to the relative probability of their trajectories occurring under the target and behavior policies, called ***the importance-sampling ratio***.

$$P(A_t, S_{t+1}, A_{t+1}, \dots, S_T \big| S_t, A_{t:T-1} \sim \pi) = \prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)$$

where p is the state-transition probability. The importance-sampling ratio is defined as

$$\begin{aligned} \rho_{t:T-1} &= \frac{P(A_t, S_{t+1}, A_{t+1}, \dots, S_T \big| S_t, A_{t:T-1} \sim \pi)}{P(A_t, S_{t+1}, A_{t+1}, \dots, S_T \big| S_t, A_{t:T-1} \sim b)} \\ &= \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k) p(S_{k+1}|S_k, A_k)} \\ &= \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} \end{aligned} \tag{1}$$

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

With importance sampling, the estimate of state-value function $v_\pi(s)$ can be obtained by average

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

Input: an arbitrary target policy $\pi$
Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
    $Q(s,a) \in \mathbb{R}$ (arbitrarily)
    $C(s,a) \leftarrow 0$

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

Figure 4: Off-policy every-visit Monte Carlo prediction with weighted importance sampling

returns based on policy b and ratio ρ

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_b\left[\rho_{t:T-1} G_t | S_t = s\right] \\ &\approx \frac{1}{|N(s)|} \sum_{t \in N(s)} \rho_{t:T(t)-1} G_t \end{aligned} \tag{2}$$

where $N(s) := \{t : S_t = s\}$ for an every-visit method; for a first-visit method, $N(s)$ would only include time steps that were first visits to s within their episodes. And let $T(t)$ be the first time of termination following time t (i.e. end of current episode), and G_t denote the return after t up through $T(t)$. The above method is called ***ordinary importance sampling***.

An important alternative is ***weighted importance sampling***, which uses a weighted average, defined as

$$V(s) := \frac{\sum_{t \in N(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in N(s)} \rho_{t:T(t)-1}}, \tag{3}$$

or zero if the denominator is zero.

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*** whereas *weighted importance sampling* is *biased* (though the bias converges asymptotically to zero). On the other hand, the variance of ordinary importance sampling is in general ***unbounded*** because the variance of the ratios can be unbounded, whereas in the weighted estimator the largest weight on any single return is ***one***. In fact, assuming bounded returns, the variance of the weighted importance-sampling estimator converges to zero even if the variance of the ratios themselves is infinite. In practice, the weighted estimator usually has dramatically lower variance and is strongly preferred. 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.

```
**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)
    $C(s,a) \leftarrow 0$
    $\pi(s) \leftarrow \arg\max_a Q(s,a)$    (with ties broken consistently)

Loop forever (for each episode):
    $b \leftarrow$ any soft policy
    Generate an episode using $b$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    $W \leftarrow 1$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
        $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t,A_t)} [G - Q(S_t, A_t)]$
        $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$    (with ties broken consistently)
        If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
        $W \leftarrow W \frac{1}{b(A_t|S_t)}$
```

Figure 5: Off-policy Monte Carlo Control

4.3 incremental update of off-policy prediction

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

In weighted importance sampling, suppose we have a sequence of returns $G_1, G_2, ..., G_{n-1}$, all starting in the same stateand each with a corresponding random weight W_i (e.g., $W_i = \rho_{t:T(t_i)-1}$). We wish to form the estimate

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

We can derive an incremental update as below:

$$V_{n+1} = V_n + \frac{W_n}{C_n}\left[G_n - V_n\right], \quad n \geq 1 \tag{4}$$

$$C_{n+1} = C_n + W_{n+1} \tag{5}$$

where $C_0 = 0$. The Figure 4 contains a complete episode-by-episode incremental algorithm for Monte Carlo policy evaluation. The algorithm is nominally for the off-policy case, using weighted importance sampling, but applies as well to the on-policy case just by choosing the target and behavior policies as the same.

5 Off-policy Monte Carlo Control

Combine two sections above, we obtain the **off-policy control algorithm**. Recall that the distinguishing feature of on-policy methods is that they *estimate the value* of a policy while using it for control. In off-policy methods these two functions are separated. An advantage of this separation is that the target policy may be *deterministic* (e.g., greedy), while the behavior policy can *continue to sample* all possible actions. They follow the behavior policy while learning about and improving the target policy. To maintain continual exploration, the behavior policy is required to be *soft*.

Figure 5 shows the off-policy Monte Carlo Control using weighted importance sampling. This follows the GPI to estimate π_* and q_*. The target policy $\pi \approx \pi_*$ is the greedy policy with respect to Q, which is an estimate of q_π. The behavior policy need to provide coverage for target π. Thus choosing the ϵ-soft policy.

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. Potentially, this could greatly slow learning. There has been insufficient experience with off-policy Monte Carlo methods to assess how serious this problem is.