Giorgio Mendoza

RBE595-S24-S04

Dr. Dadkhah Terani

Reinforcement Learning

HW3 Monte-Carlo Assignment

1- When is it suited to apply Monte-Carlo to a problem?

The Environment is Episodic: Monte Carlo methods are well-suited for episodic tasks where the agent interacts with the environment over a series of episodes, with each episode having a well-defined start and end.

The Environment is Fully Observable: Monte Carlo methods do not require knowledge of the underlying environment dynamics or transition probabilities. Therefore, they are suited for fully observable environments where the agent can directly observe the states and rewards.

The Problem is Model-Free: Monte Carlo methods are model-free, meaning they do not require a model of the environment. Instead, they learn directly from experience by sampling trajectories.

The Problem is High Variance: Monte Carlo methods can be effective in situations where there is high variance in the returns, as they rely on averaging over many samples to estimate expected returns accurately.

The Problem has Long Episodes: Monte Carlo methods can handle problems with long episodes, as they accumulate rewards over entire episodes before updating the value function.

2- When does the Monte-Carlo prediction perform the first update?

In Monte Carlo prediction, the first update typically occurs after the completion of an episode. The flow of the algorithm usually goes as follows:

Episode Completion: First, the agent interacts with the environment until the episode terminates, reaching a terminal state.

Gathering Trajectory Information: During the episode, the agent records the sequence of states, actions, and rewards experienced throughout the trajectory.

Calculation of Returns: After the episode ends, the agent calculates the return for each state visited during the episode. The return is the sum of rewards from the current time step until the end of the episode.

Value Function Update: The agent updates the value function estimates based on the returns obtained. Typically, this involves averaging the returns observed for each state over multiple episodes and using them to update the value estimates for those states.

Iterative Improvement: After the first update, the agent continues to interact with the environment, collecting more episodes and updating its value function estimates after each episode. The process iterates until the value function converges to an optimal or near-optimal solution.

So, the first update in Monte Carlo prediction occurs once the agent completes its first episode and has gathered enough information to estimate the value function for states visited during that episode.

3- What is off-policy learning and why is it useful?

Off-policy learning is an approach in RL where the agent learns from data generated by following a different policy than the one being learned. In other words, the agent learns about the optimal policy while executing a different behavior policy. The following are some of the advantages of the Off-Policy approach:

Flexibility: Off-policy learning allows the agent to learn from a broader range of data. The agent can learn from historical data collected from previous policies or even human demonstrations, which can be advantageous in various real-world applications.

Sample Efficiency: Since off-policy learning decouples the behavior policy (the policy generating actions) from the target policy (the policy being learned), it can reuse past experiences more efficiently. This leads to better sample efficiency, as the agent can learn from a wider variety of experiences.

Exploration-Exploitation Tradeoff: Off-policy learning enables the agent to separate the exploration strategy (behavior policy) from the exploitation strategy (target policy). This decoupling allows the agent to explore more diverse actions without impacting the learning process, leading to better exploration strategies.

Policy Evaluation and Control: Off-policy learning is particularly useful for policy evaluation tasks, where the goal is to estimate the value or Q-function of a given policy. It allows the agent to evaluate different policies without needing to follow them directly, which can be computationally expensive or impractical.

Stability and Robustness: Off-policy learning algorithms tend to be more stable and robust compared to on-policy methods, especially in environments with non-stationary or noisy data. This stability is because of the ability to learn from a diverse set of experiences, even those generated by suboptimal or exploratory policies.

4- (Exercise 5.5) Consider an MDP with a single nonterminal state and a single action that transitions back to the nonterminal state with probability p and transitions to the terminal state with probability 1p. Let the reward be +1 on all transitions, and let γ = 1. Suppose you observe one episode that lasts 10 steps, with a return of 10. What are the first-visit and every-visit estimators of the value of the nonterminal state?

Given that the reward is +1 for all transitions and the discount factor γ=1, the return of 10 indicates that there were 10 transitions before reaching the terminal state.

Since the episode lasts 10 steps with a return of 10, the nonterminal state must have been visited for the first time on the first step. The return following the first visit is 10, because γ=1 and there is a +1 reward for each step.

Every time the nonterminal state is visited, it contributes to the estimate. Since γ=1, we don't have to discount future rewards, and we can simply average the returns from each visit. If the nonterminal state is visited on every step before the terminal state, then it is visited 10 times, and the return after each visit is 10, 9, 8, ..., 1 respectively.

$$Every\ visit\ estimate\ = (10 + 9 + 8 + 7 + \ldots + 1)/(10)$$

The sum of the first n natural number is given by $n(n + 1)/2$, so the sum of the first 10 natural number is $10\ x\ 11/2\ =\ 55$. The average is then 55/10 = 5.5. Therefore, the first-visit estimator for the value of the nonterminal state is 10 and the every-visit estimator is 5.5.

5- (Exercise 5.7) In learning curves such as those shown in Figure 5.3 error generally decreases with training, as indeed happened for the ordinary importance-sampling method. But for the weighted importance-sampling method error first increased and then decreased. Why do you think this happened?

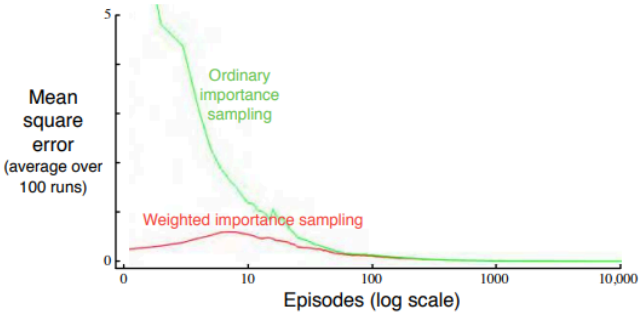

Figure 5.3: Weighted importance sampling produces lower error estimates of the value of a single blackjack state from off-policy episodes. ■

In the figure 5.3, we can see the mean square error of the value estimates plotted against the number of episodes for both ordinary importance sampling and weighted importance sampling methods. The plot shows the error decreasing over time for ordinary importance sampling. For

weighted importance sampling, the error initially increases and then decreases. These are some of the things that could've happened:

Initial Learning Phase: In the beginning episodes, the weighted importance sampling might give undue weight to certain atypical returns, especially if those returns are associated with low-probability actions under the behavior policy. This can cause large fluctuations in the estimate, resulting in an initial increase in error.

Stabilization with More Data: As more episodes are processed, the weighted importance sampling benefits from the accumulation of more data, which helps to stabilize the estimates. The weights help to correct for the discrepancy between the target and behavior policies, and as the data accumulates, this correction becomes more accurate, leading to a reduction in error.

Impact of Rare or Extreme Returns: Early on, a few extreme returns that are either too high or too low can distort the estimated value. Since these returns are weighted by the inverse of their probability under the behavior policy, they can have a significant impact. As more samples are collected, the influence of any single extreme return diminishes, and the estimate converges towards the expected value.

Normalization Effect: The weighted importance sampling uses the sum of weights as a normalizing factor, which can be quite volatile in the early stages when the sum is small. With more episodes, the normalizing sum grows larger, which reduces the impact of individual weights, leading to a smoother and more accurate estimate.

Law of Large Numbers: As the number of episodes increases, the law of large numbers starts to take effect. The variance of the weighted average decreases as more data points are included, which naturally leads to a decrease in the mean square error.

Bias-Variance Trade-Off: Initially, the estimates may be biased due to a lack of sufficient sampling, but as the number of episodes increases, the bias is reduced. However, variance is

high at the start due to the reasons mentioned above and then it decreases with more episodes, improving the overall estimate.

The initial increase in error for weighted importance sampling can therefore be attributed to the disproportionate impact of rare or extreme returns early in learning, and the subsequent decrease in error reflects the stabilization of the value estimate as more data is gathered and the normalizing effect of the weights becomes more pronounced. This behavior is usual in cases where the initial data may not be fully representative of the long-term expected values, but over time, as the data becomes more representative, the estimates converge to the true value, resulting in lower error.

6- (Exercise 5.8) The results with Example 5.5 and shown in Figure 5.4 used a first-visit MC method. Suppose that instead an every-visit MC method was used on the same problem. Would the variance of the estimator still be infinite? Why or why not?

In the given setup, the target policy always selects the left action which has a 0.9 probability of leading back to the same state and a 0.1 probability of terminating. The behavior policy selects right and left with equal probability.

Every-visit vs. First-visit MC Methods: In a first-visit MC method, only the first time a state is visited in an episode contributes to the estimate of that state's value. The every-visit MC method, however, includes every visit to the state in the estimation process.

Variance of the Estimator: For the every-visit MC method, the variance of the estimator depends on the distribution of the returns. If the returns are heavily skewed due to the policy or the structure of the MDP, this could lead to high variance. For the provided example, the ordinary importance sampling is leading to infinite variance due to the loops causing potentially infinite returns.

Impact of Every-visit MC: Switching to an every-visit MC method would mean that each loop back to the state under the target policy would contribute to the estimate. If we have a situation

where the returns do not normalize (due to, for example, the discount factor being 1 and the potential for infinite loops), then the variance could still be infinite. This is because the sample returns could be arbitrarily large due to the possibility of looping an arbitrary number of times before termination.

Behavior Policy and Exploration: If the behavior policy provides sufficient exploration, the every-visit MC method could sample enough diverse trajectories to obtain a more accurate estimate. However, if the behavior policy still has a high probability of falling into loops, as in the example given, then these loops will contribute multiple high-return samples, potentially leading to infinite variance.

In summary, switching to an every-visit MC method in the described scenario would not necessarily resolve the issue of infinite variance. The key problem is the structure of the MDP and the policies involved, which can lead to potentially infinite loops with high returns, affecting the variance of the estimator. Without additional mechanisms to handle the loops (such as using a discount factor less than 1 or changing the structure of the problem to prevent infinite loops), the variance could remain infinite even with an every-visit MC method.

7- (Exercise 5.10) Derive the weighted-average update rule (5.8) from (5.7). Follow the pattern of the derivation of the unweighted rule (2.3).

The unweighted update rule has the form of a running average where the new estimate is the old estimate plus a step-size times the error (the difference between the new sample and the old estimate).

Given the weighted-average form:

$$V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}$$

The task is to update this estimate when a new return G_n and weight W_n are obtained, then we want to derive an incremental formula that doesn't require to store all the G_k and W_k but only the current estimates and the cumulative weights.

The new estimate V_{n+1} after including G_n with weight W_n is:

$$V_{n+1} = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k}$$

Then we can express the sums up to n as the sum to n - 1 plus the nth term:

$$V_{n+1} = \frac{\sum_{k=1}^{n-1} W_k G_k + W_n G_n}{\sum_{k=1}^{n-1} W_k + W_n}$$
$$V_{n+1} = \frac{\sum_{k=1}^{n-1} W_k G_k + W_n G_n}{C_n + W_n}$$

Now, substituting V_n from eq 5.7 into the equation to get:

$$V_{n+1} = \frac{C_n \cdot V_n + W_n G_n}{C_n + W_n}$$

The next step would be to solve for V_{n+1} in terms of V_n, G_n, W_n and C_n. We would need to multiply both numerator and denominator by $1/C_n$ to isolate V_n on the right side of the equation.

$$V_{n+1} = \frac{V_n + \frac{W_n}{C_n} G_n}{1 + \frac{W_n}{C_n}}$$

So, bringing V_n to one side and the rest to the other to get:

$$V_{n+1} - V_n = \frac{\frac{W_n}{C_n} G_n - V_n \frac{W_n}{C_n}}{1 + \frac{W_n}{C_n}}$$
$$V_{n+1} - V_n = \frac{W_n}{C_n + W_n}(G_n - V_n)$$

This would give the update rule in the form of an error correction term, where $(W_n)/(C_n + W_n)$ acts as a step-size that diminishes as the cumulative weight C_n grows:

$$V_{n+1} = V_n + \frac{W_n}{C_n + W_n}(G_n - V_n)$$

This is eq. 5.8 and it shows how the new estimate V_{n+1} can be obtained incrementally from the previous estimate V_n, the new return G_n and the new weight W_n, without needing to store or recompute all the previous returns and weights.

8- (Exercise 5.11) In the boxed algorithm for off-policy MC control, you may been expecting the W update to have involved the importance-sampling ratio $\pi(A_t|S_t)/b(A_t|S_t)$, but instead it involves $1/b(A_t + S_t)$. Why is this nevertheless correct?

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)
$C(s,a) \leftarrow 0$
$\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$ (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, $t = T-1, T-2, \ldots, 0$:
$G \leftarrow \gamma G + R_{t+1}$
$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$
$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (with ties broken consistently)
If $A_t \neq \pi(S_t)$ then exit inner Loop (proceed to next episode)
$W \leftarrow W \frac{1}{b(A_t|S_t)}$

In off-policy methods, importance sampling is used to correct for the difference in the action selection probabilities between the target policy π and the behavior policy b. The importance-sampling ratio $\pi(A_t|S_t)/b(A_t|S_t)$ is the factor by which the returns are scaled to account for this difference.

However, the update rule in the algorithm shown only uses $1/b(A_t + S_t)$. This is because the control algorithm is interested in finding a greedy policy with respect to the action-value function Q. When the action A_t is not equal to the action given by the current greedy policy π, the

algorithm doesn't update the action-value function Q since these actions would not be selected by π. Therefore, when $A_t \neq \pi(S_t)$, the importance-sampling ratio would be zero because $\pi(A_t|S_t) = 0$ (assuming π is deterministic and assigns a probability of 1 to the action that maximizes Q, hence $\pi(A_t|S_t) = 1$ for the chosen ratio.

Therefore, the weight W is updated with $1/b(A_t + S_t)$ because this is the only factor that is needed to correct the estimate when the action A_t is indeed taken by the target policy. This weights the return by how unlikely it was under the behavior policy, but since it was taken by the target policy, the numerator of the importance- sampling ratio is always 1 so it's omitted.

The correction is "nevertheless correct" because it adjusts for the fact that the behavior policy may over-sample or under-sample certain actions compared to the target policy. For this case, we're only interested in evaluating the policy that always takes the greedy action, we don't need to account for actions not taken by π beyond not updating their values, which is why we don't use the full ratio $\pi(A_t|S_t)/b(A_t|S_t)$ in the weight update.