Reinforcement Learning

Zexuan Meng

November 8, 2023

1 Monte Carlo Methods

Monte Carlo methods in RL are a class of algorithms that rely on *repeated random sampling* and average the sample returns. They are used to estimate the value functions and discover optimal policies. Monte Carlo methods are used when the agent's environment is modeled as a Markov Decision Process (MDP), but the agent does not have complete knowledge of the environment. In particular, it does not know the transition probabilities and rewards for each state. Instead, it learns from experience by sampling episodes of experience from the environment. The *experience*, sequences of states, actions, and rewards, are the only information required to apply Monte Carlo methods. We define Monte Carlo methods only for episodic tasks. That is, the experience is divided into episodes and all episodes will eventually terminate.

To conclude, the key characteristics of Monte Carlo methods are:

- **Model-free**: Monte Carlo methods do not require a model of the environment's dynamics.
- **Episodic-based**: Monte Carlo methods learn from complete episodes. They do not bootstrap.
- **Sampled returns**: Monte Carlo methods use the *empirical mean return* rather than the *expected return* to estimate the value functions.

1.1 Basic concept for Monte Carlo methods

1.1.1 Repeated random sampling

Repeated random sampling is a statistical technique where subsets, or samples, are taken from a larger population or dataset while each sample is selected randomly. The idea behind repeated random sampling is to obtain a representative subset of the larger population, which can be analyzed to infer the properties of the entire population. Each sample should be selected using a process that gives every possible sample an *equal* chance of being chosen.

Here are some sampling techniques that may be employed:

- **Simple random sampling**: Every member of the population has an equal chance of being included in the sample.
- **Stratified sampling**: The population is divided into mutually exclusive groups, called strata, and random samples are taken from each stratum.
- **Cluster sampling**: The population is divided into mutually exclusive groups, called clusters, and then all members of the clusters are included in the sample.

1.1.2 Bootstrapping in Monte Carlo methods

Bootstrapping refers to the process of updating estimates based on other current estimates rather than waiting until the final outcome is known.

Monte Carlo methods *do not* bootstrap. Because they update the value function only at the end of an episode, based on the entire sequence of observed rewards from the episode. They use the total accumulated reward from the current state until the end of the episode to update the value function. In this case, the value function for a certain state is updated based on the *actual rewards* rather than the *expected rewards*.

1.2 Monte Carlo Prediction

First, we would like to learn the state-value function for a given policy with Monte Carlo methods. The value of a certain state, the expected return, is the expected cumulative future discounted reward starting from that state. To estimate from experience, we can simply average the returns observed after visits to that state. *As more returns are observed, by the law of large numbers, the average should converge to the expected value.* This idea is the basis of Monte Carlo prediction.

Each occurrence of state s in an episode is called a *visit* to s.

- **First-visit**: The first time-step t that state s is visited in an episode.
- **Every-visit**: Every time-step t that state s is visited in an episode.

First-visit Monte Carlo methods estimate the value of a state as the average of the returns following first visits to that state. Every-visit Monte Carlo methods estimate the value of a state as the average of the returns following all visits to that state. The two methods converge to the true value function as the number of visits to each state goes to infinity. Notice that there is no guarantee to pass by all states. Here we only focus on first-visit Monte Carlo methods.

Detailed explanation of Algorithm 1:

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

Input: a policy π to be evaluated
Initialize:
  $V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$
  $Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$
loop for each episode
  Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for $t = T-1, T-2, \ldots, 0$ **do**
    $G \leftarrow \gamma G + R_{t+1}$
    if S_t not 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 loop

Initialize:

- $V(s)$: value function in state s.
- $Returns(s)$: total rewards in state s.

Loop for each episode:

- Generate an episode: how we use the policy.
- G: the total return from the current state to the end of the episode.
- if S_t not in $S_0, S_1, \ldots, S_{t-1}$: check if it is the first visit to state S_t. This clause will ensure that we only count the first visit to state S_t.
- $V(S_t) \leftarrow$ average$(Returns(S_t))$: the average here is the key idea of MC methods, the utilization of law of large numbers.

Now let's talk about why we did a backward loop in each episode to update the total return. The reason is very intuitive since the return G_t from a state S_t is the total discounted reward received from time t *onwards* until the end of the episode:

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-t-1} R_T \\ &= R_{t+1} + \gamma G_{t+1} \end{aligned}$$

Since R_T is the immediate reward provided by the terminal state, G_T is always 0. In this case, $G_{T-1} = R_T + \gamma G_T = R_T$ and it's really easy to start at this point.

$$
\begin{aligned}
G_{T-1} &= R_T + \gamma G_T = R_T \\
G_{T-2} &= R_{T-1} + \gamma G_{T-1} = R_{T-1} + \gamma R_T \\
G_{T-3} &= R_{T-2} + \gamma G_{T-2} = R_{T-2} + \gamma R_{T-1} + \gamma^2 R_T \\
&\vdots \\
G_1 &= R_2 + \gamma G_2 = R_2 + \gamma R_3 + \gamma^2 R_4 + ... + \gamma^{T-2} R_T \\
G_0 &= R_1 + \gamma G_1 = R_1 + \gamma R_2 + \gamma^2 R_3 + ... + \gamma^{T-1} R_T
\end{aligned}
$$

Why can we do the calculation like this? Because the Monte Carlo diagram goes all the way to the end of the episode and there is no diversion. Another important fact about Monte Carlo methods is that the estimates for each state are independent. Above all, I think that's why we chose to start at the end of the episode to update the total return.

1.3 Compare with Dynamic Programming

Let's recall the Bellman equation for the optimal state-value function:

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right]$$

In Dynamic Programming, we need to know the *transition matrix* as well as the *reward system*, but it is not always the realistic condition. In reality, we may not know $p(s', r|s, a)$. In this case, we cannot compute $V(s)$. By adopting Monte Carlo methods, we play enough number of episodes of the game and extract the information needed.

In DP, we didn't play the game because we already knew the dynamics of the game, which is at each state we knew what are the probabilities of going to another state when we took certain actions, and we knew what the reward is going to be. In Monte Carlo methods, we don't know those data unless we play the game. That's the key difference between DP and MC methods.