Monte Carlo Methods

Whereas Dynamic Programming learns the task in a iterative approach where you learn for every state or state-action values, Monte Carlo method learns for only the state-actions taken throughout the episode. This requires keeping track of state-action Q values just like Policy Iteration, so Monte Carlo is useful when you have to learn $v_*(s)$ or $q_*(s,a)$ via sampling/experience because you have no model $p(s^{'}, r^{'} | s, a)$. To generate the trajectory/experience, you use policy π to select your actions throughout the episode:

$$S_0, A_0, R_1, S_1, A_1, ..., S_{T-1}, A_{T-1}, R_T. \tag{1}$$

At the end of episode, you compute the return for every state visited:

$$v_\pi(s) = \mathrm{E}_\pi[G_t | S_t = s], \tag{2}$$

or

$$q_\pi(s,a) = \mathrm{E}[G_t | S_t = s, A_t = a]. \tag{3}$$

where $G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$.

Therefore, the estimated value for a state s or state action (s,a) is the average of all the returns that the agent has collected in that state or action of the state:

$$V_\pi(s) = \frac{1}{N} \sum_{k=1}^{N} G_{s_k}, \tag{4}$$

and

$$Q_\pi = \frac{1}{N} \sum_{k=1}^{N} G_{s,a_k}. \tag{5}$$

Calculating Values

For a generated trajectory $(S_0, A_0, R_1, S_1, A_1, ..., S_{T-1}, A_{T-1}, R_T)$, we calculate the returns for each moment t:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... + \gamma^{T-t-1} R_T. \tag{6}$$

We can't use V anymore because value iteration does not update policy π or remember state-value Q, and solving the task requires the formulation of policy by evaluating **each individual action** in a state. So, we can only learn for Q which already implicity learn the model:

$$q_\pi(s,a) = \sum_{s^{'},r} p(s^{'}, r | s, a)[r + \gamma v(s^{'})]. \tag{7}$$

where as if you want to choose and action from V, you still need to pick the action a with the highest value, and to do that, you need know the next state for every action a leads to and you need to know model p to get probability distributions for the next state $s^{'}$ from (s, a).

Importance of Exploration

Since $Q(s, a)$ is an estimate that is improved by the agent collecting experience following unoptimal policies. Therefore, the estimates may not be accurate especially in the beginning of the learning and there is a chance that the a bad estimate prevents the agent from ever choosing (s, a) that might become optimal in the future. To prevent this, we make sure that all actions are chosen from time to time using:

1. Exploring starts with random state $S_0 \sim S$ and random action $A_0 \sim A(S_0)$ and,
2. Stochastic policies: $\pi(a|s) > 0, \forall a \in A(s)$.

Stochastic Policies

You can either generate the experience with the same policy you're trying to optimize (On-Policy), or generate experience with an exploratory policy b different from the one we're going to optimize.

ϵ-greedy Policy

You select a random action at probability ϵ, and select highest $Q(s, a)$ at probability $1 - \epsilon$:

$$\pi(a|s) = \{\begin{matrix} 1 - \epsilon + \epsilon_r & \text{for } a = a* \\ \epsilon_r & \text{for } a = a* \end{matrix} \tag{8}$$

where $\epsilon_r = \frac{\epsilon}{|A|}$.

Off-Policy strategy

We can use 2 seperate policies for exploration $b(a|s)$ to collect the experience/trajectory, and optimization (Target policy $\pi(a|s)$ that uses the experience from b) to improve towards optimal policy:

$$\pi(s) \leftarrow \arg\max_a Q(s, a). \tag{9}$$

This means that Exploratory policy has to cover all the actions that the Target policy can take:

$$\pi(a|s) > 0 \Rightarrow b(a|s) > 0. \tag{10}$$

Both b and π will still be using the same action values, but b will have a bit more randomness in choosing the action, so the average return is not approximated under π but under b which handles the exploration:

$$\mathrm{E}_b[G_t | S_t = s, A_t = a] = q_b(s, a). \tag{11}$$

We need to make sure that the Exploratory policies are exploring properly. We can use the **Importance Sampling**, statistical technique to estimate the expected values of a distribution by working with samples from another distribution:

$$W_t = \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)} \tag{12}$$

which gives the ratio of probability of following the trajectory by target policy and probability of following the trajectory by exploratory policy.

By correcting the returns using importance sampling, we will approximate the value under π:

$$\mathrm{E}[W_t G_t \mid S_t = s] = v_\pi(s). \tag{13}$$

We then update the Q values iteratively using the *alpha* approach:

$$Q(s, a) \leftarrow Q(s, a) + \frac{W_t}{C(s, a)}[G - Q(s, a)] \tag{14}$$

where $C(s, a) = \sum_{k=1}^{N} W_k$ to normalize the updates to smooth out the learning process.