蒙特卡洛方法

之前介绍的通过迭代法解决bellman方程的过程中，都假设模型已知，也就是环境的动态属性已知（$p(s', r|s, a)$ 是随环境变化，之前都假设知道每个时刻的 $p(s', r|s, a)$ ，并且状态数量少且是离散的）。然而在实际问题中，我们可能对环境的动态特性并不是那么清楚，但是我们可以得到足够多的数据，那么我们同样可以用强化学习来建模解决这个问题，这类不利用模型的算法被称为Model-free的方法。Monte Carlo方法便是一种Model-free的方法。

Monte Carlo的核心思想是用样本均值来近似期望。依据大数定理，对于随机变量X，给定N个iid采样的样本 $\{x_j\}_{j=1}^N$, 则 $\bar{x} = \frac{1}{N}\sum_{j=1}^{N} x_j$ 可以视为 $E(X)$ 的无偏估计。

Monte Carlo Method

回忆策略评估和策略优化的过程中，需要计算 $p_\pi(s'|s)$ 和 r_π ，但是在Model-free的算法中不知道精确的p和r，可以采用Monte Carlo Method直接估计给定策略下的v(s) 和q(s, a)，我们先讨论v(s)，q(s, a)与之类似。

$$\text{策略评估：}\ v_\pi = r_\pi + \gamma P_\pi v_\pi$$

$$\text{策略优化：}\ \pi = \arg\max_\pi (r_\pi + \gamma P_\pi v_\pi)$$

回顾v(s)的定义：每个状态下的累计回报的期望

$$v_\pi(s) = E[G_t|S_t = s]$$

由大数定理可知，可以用样本均值估计期望，假设 $g_1, ..., g_n$ 是多个episode得到的n个s状态的累计回报，则 $v(s) = \frac{g_1 + ... + g_n}{n}$ 是 G_t 的无偏估计。

假设某个episode为 $S_1, A_1, R_2, A_2, S_2, R_3, S_1, A_1, R_4$ ，针对出现了两次的 S_1 ,有两种方法：

First-visit： 每个episode的数据只记录每个S第一次出现的值，即只记录
$g(S_1) = R_4 + \gamma R_3 + \gamma^2 R_2$

Every-visit: 每个episode的数据记录每次S出现的值，还记录 $g(S_1) = R_4$

下面的伪代码是Every-visit的，采取倒序遍历所有episode，计算得到每一个时刻的奖励值，分别放到对应状态的集合中，计算每一个状态的累计回报来估计真实的v(s)。被访问次数多的状态计算出的v(s)越接近真实。q(s, a)的计算与之类似。

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 forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

Monte Carlo的策略优化与迭代法一致。策略评估和策略优化依次进行，即针对每个episode，先进行策略评估，再进行策略优化。但是策略优化面临一个问题，因为采取贪心的策略，要确保所有(s,a)都被计算到，意味着我们需要从每个(s,a)对开始生成足够多的episode。

为了解决以上问题，采用soft policy ε-greedy，足够长的episode中可能就会包含每个(s,a)对足够多次，不需要从每个(s, a)对开始采集大量episodes，具体公式如下：

$$\pi(a \mid s) = \begin{cases} 1 - \frac{\varepsilon}{|\mathcal{A}(s)|}(|\mathcal{A}(s)| - 1), & \text{for the greedy action,} \\ \frac{\varepsilon}{|\mathcal{A}(s)|}, & \text{for the other } |\mathcal{A}(s)| - 1 \text{ actions.} \end{cases}$$

其中 $\epsilon \in [0, 1], |\mathcal{A}(s)|$ 是状态总数。该式保证了采取贪婪行动的概率总是大于其他行动，通过调整 ϵ 的大小，可以实现exploitation & exploration的平衡。

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 $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

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 $a \in \mathcal{A}(S_t)$:

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

当状态的数量很大时，Monte Carlo的计算量要显著低于迭代法，因为可以只对重要的(也就是在episode中经常被访问到的)状态进行采样和更新，而求解Bellman方程需要更新所有的状态。

存在的问题：

- 每次更新需要等到一个episode结束，只适合Episodic Task。
- 当episode很长或者数量很多时，效率低

Temporal Difference 时序差分

可以在一边生成episode的过程中，一边用生成好的样本进行估算，不需要完整的episode。

按照Robbins-Monro算法（一种随机近似算法），$v(S_t)$ 作为要更新迭代的值，G_t 是采样到的样本值，就有

$$v(S_t) = v(S_t) + \alpha[G_t - v(S_t)]$$

其中 $G_t = R_{t+1} + \gamma R_{t+2} + ... = R_{t+1} + \gamma G_{t+1}$ ，又已知：

$$v_\pi(s) = E[G_t|S_t = s] = E[R_{t+1} + \gamma G_{t+1}|S_t = s] = R_{t+1} + \gamma v_\pi(S_{t+1})$$

用 $E(G_t)$ 近似代替 G_t ,可以得到：

$$v(S_t) = v(S_t) + \alpha[R_{t+1} + \gamma v(S_{t+1}) - v(S_t)]$$

这样只需要知道下一个状态就可以对当前状态进行更新.

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
    $S \leftarrow S'$
  until S is terminal

与之类似的，在策略优化时需要用到的q(s, a)也可以使用TD进行更新。在这里S'是下一个状态，A'是根据策略在S'下采取的动作，区别是如果S'是终止状态，没有A'，额外定义q(S', A') = 0。当更新了q(s, a)后，可以再对策略进行更新（ε-greedy），

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal