Chapter 5 Monte Carlo Method

Created	@November 6, 2023 4:19 PM
Tags	

Written By YEE

5.1 Monte Carlo Prediction

我們先使 Monte Carlo methods 為利用policy來學習state-value function。我們以前有學過value of the state是**期望回報值**，或是也可以說是從那刻開始算起的**期望累積獎勵**。很明顯這個情況下我們可以用既有經驗來去**預測它在該state觀測到的平均回報值**，**這些逐漸累積的回報值的平均最終會收束到一個期望值**。這個就是 Monte Carlo methods的精隨。

假設我們要預測 $v_\pi(s)$，我們有獲得一連串的episode是從 π 帶入 s 所獲得的資訊。在一個episode中， s 是可以被到訪(visit)多次的。每個 episode的對該 state的第一次到訪我們稱之為：第一次對 s的到訪 (first visit to s)。The first-visit MC method 是計算第一次到達state s 的回報值，而every-visit MC method是預測每次訪問 state s 的回傳值的平均，這兩個MC method 非常相似。如今first-visit MC method比較受廣泛討論，歷史追述至1940年代，這張主要討論的是first-visit MC method。

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

Input: a policy π to be evaluated

Initialize (初始化)：

$V(s) \in \mathbb{R}$, arbitrary, for all $s \in \mathcal{S}$

$Return(s) \leftarrow$ 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, $t = T-1, T-2, \ldots, 1$ (由後往前)

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

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$ (first-visit only,如果出現過則忽略)

Append G to $Returns(S_t)$

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

當拜訪 s 的次數接近無限大時 $v(s)$ 就會收束。每次的抽樣平均若是 unbiased 的話這些抽樣平均的標準差會是 $1/\sqrt{(n)}$， n 是抽樣 n 次平均回傳值。

Example 5-1 Black Jack(21點賭場版本)

雖然我們已經有 Black Jack 的 environment 資訊，但還是很難用 DP method 去計算。DP method需要下一個state的分佈($p(s', a|s, r)$)，在 Black Jack 中不好定義。假如說玩家現在有14點然後玩家選擇要stick，莊家翻開卡片後，玩家最後 reward 是+1的機率是多少？所有機率都要先被計算過後才能開始DP的過程，這些計算過程既繁瑣又容易出錯。然而在MC下這個問題就很簡單了。

DP列出來的是所有的可能，而MC只有使用一個episode中的樣本，然而DP只需要一步的轉換，MC需要跑完一個episode。這些差異很準確的反應在這兩個演算法的特性。

再者，MC的計算成本跟有幾個state無關，讓MC非常適合用來計算部分的 state或是單一個state。

5.2 Monte Carlo Estimation of Action Value

在沒有模型(model)的情況下，預測 action value 會比預測 state value 來的容易。有模型的話 state value 就已經足夠。在沒有模型的情況下，state value是不夠的，我們還必須要明確地預測每個action 的 value 才能對policy有幫助。

policy evaluation 是預測 $q_\pi(s, a)$， state-action pair (s, a)在到達state s 並採取 action a 的情況才可以說已經到訪(visit)這個 (s, a) state-action pair。

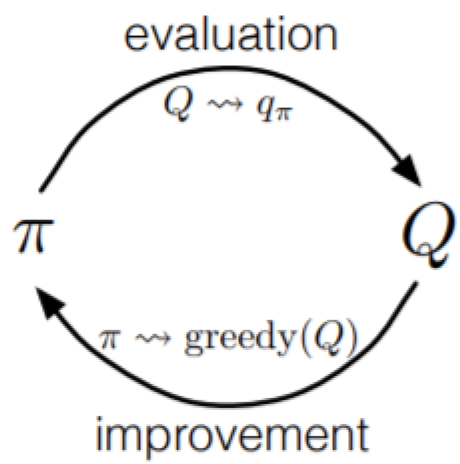
唯一的問題是這種方式可能會造成某些 (s, a) 沒辦法到訪，沒走過的state因為沒有回傳值平均所以不會從經驗中進步，這是個很嚴重的問題因為我們需要比較所有可行的action才能做選擇。

我們需要確保每個 (s, a)都有到訪過的方法是加入exploration的方式。Episode 從 (s, a) pair 開始，然後每個pair都有大於零的機率被選擇到，這保證了在每個 (s, a) 在無限個episode之前都會到訪。

這個Exploration 有時候是有用的，不過他不能被過度的依賴，特別是直接在實際的action上做學習，如果是剛剛的情況下起始的幫助就不大。最常見確保每個 (s,a)pair的方法就是在policy上選擇任意非零的機率選擇任意一個 (s,a)。上述以後會談到。

5.3 Monte Carlo Control

我們現在可以說MC Estimation可以在受控制的情況下使用。根據先前談過的GPI



Action-value function 反覆修正讓其更接近policy 的 action-value，policy 從現在的action-value function不斷的改進。

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} q_{\pi_2} \xrightarrow{I} \pi_3 \ldots \xrightarrow{I} \pi_* \xrightarrow{E} q_*$$

$\xrightarrow{E}$是完整的policy evaluation

$\xrightarrow{I}$是完整的policy improvement

我們可以利用讓policy greedy的方式(對應當前的value function)實現Policy Improvement。

隨著多個Episode的經驗，action value 的近似值會逐漸接近真實的function。

$$\pi(s) = \arg\max_a q(s,a) \tag{5.1}$$

然後我們可以利用每一個 π_{k+1}對應於 q_{π_k} 構建Policy Improvement

Policy Improvement Theorem可以應用在 π_k 跟 π_{k+1}因為 For all $s \in \mathcal{S}$,

$$
\begin{aligned}
q_{\pi_k}(s, \pi_{k+1}(s)) =& q_{\pi_k}(s, \arg\max_a q_{\pi_k}(s,a)) \\
=& \max_a q_{\pi_k}(s,a) \\
\geq& q_{\pi_k}(s, \pi_k(s)) \\
\geq& v_{\pi_k}(s)
\end{aligned}
$$

我們假設 policy evaluation 經歷了無限個 episode之後 policy就可以被推算出來。證明這個假設的方法其實十分簡單。這個在classical DP method也有這個 iterative policy evaluation，只會漸進式的接近true value function。

在DP跟MC我們皆有兩種方法去了解問題，第一個是建立在每次 policy evaluation 對 q_{π_k} 的預測，足夠的迭代可以保證區間收束的足夠小，不過這可能花太多時間，在小問題迭代太多個episode。

第二種方法是建立在避免迭代無限個 episodes，也就是說我們放棄在policy improvement 之前完成 policy evaluation。我們在每次policy evaluation step將其帶入 q_{π_k}，我們在重複做足夠多次之前不會期望我們會有趨近true value function的想法。我們過去在GPI就有討論過這個方法了，不過這次我們更加極端，我們在每個step完成就交替做 evaluation 跟 improvement。

而在MC policy iteration我們在每個 episode 交換做 evaluation 跟 improvement。在每個 episode 中我們觀察回傳值並應用到 policy evaluation，然後policy 在所有有經過的地方做 improvement，這可以被定義為 **Monte Carlo ES, for Monte Carlo Exploring Start**

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

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrary), for all $s \in \mathcal{S}$

$Q(s,a) \in \mathbb{R}$ (arbitrary), for all $s \in \mathcal{S},\ a \in \mathcal{A}(s)$

$Return(s,a) \leftarrow$ 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 pair have probability > 0

Generate an episode from $S_0,\ A_0$ 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, $t = T-1, T-2, \ldots, 1, 0$:

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

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 $Return(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}\left(Return(S_t, A_t)\right)$

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

EX 5.4 這個 pseudocode 效率並不好，他需要儲存所有回傳值並不斷重複計算平均值，如果能將 Incremental Implementation 運用於此將可大幅增加其效率。

Initialize:

$\pi(s) \in \mathcal{A}(s)$ (arbitrary), for all $s \in \mathcal{S}$

$Q(s, a) \in \mathbb{R}$ (arbitrary), for all $s \in \mathcal{S},\ a \in \mathcal{A}(s)$

$Return(s, a) \leftarrow$ NO LONGER NEEDED

$n \leftarrow 0$ (episode count)

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

$n \leftarrow n + 1$

$G \leftarrow \gamma G + R_{t+1}$ (This is G_n)

Unless the pair $S_t, A_t,$ appears in $S_0, A_0, S_1, A_1, \ldots S_{t-1}, A_{t-1}$

$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{n}[G - Q(S_t, A_t)]$
(Incremental Implementation, right, left Q is Q_{n-1}, Q_n)

$\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$

過程：

$$
\begin{aligned}
Q_n(S_t, A_t) =& \frac{1}{n}\sum_{i=1}^{n} G_i(S_t, A_t) \\
=& \frac{1}{n}(G_n(S_t, A_t) + \sum_{i}^{n-1} G_i(S_t, A_t)) \\
=& \frac{1}{n}(G_n(S_t, A_t) + (n-1)\frac{1}{n-1}\sum_{i}^{n-1} G_i(S_t, A_t)) \\
=& \frac{1}{n}(G_n(S_t, A_t) + (n-1)Q_{n-1}(S_t, A_t)) \\
=& \frac{1}{n}(G_n(S_t, A_t) + nQ_{n-1}(S_t, A_t) - Q_{n-1}(S_t, A_t)) \\
=& Q_{n-1}(S_t, A_t) + \frac{1}{n}(G_n(S_t, A_t) - Q_{n-1}(S_t, A_t))
\end{aligned}
$$

5.4 Monte Carlo Control without Exploring Starts

上述5.3的方法只會在最一開始時選擇任意作為Exploration start，因此並沒有確保所有state都可以被拜訪到的問題。有兩種方法可以確保我們可以拜訪到所有state，on-policy method以及 off-policy method。On-policy會嘗試著去evaluate 或 improve 拿來做抉擇的policy；Off-policy則是用不同 policy 產出的 data 來更新現在的 policy。 MC ES是 on-policy method。在這小節我們會示範如何設計不使用 Exploration start 的 on-policy MC control method 。

在 On-policy control method ， policy 會很有彈性 (**soft**)，也就是說 對於所有 $s \in \mathcal{S}, a \in \mathcal{A}(s)$policy會 $\pi(a|s) > 0$，不過過程中這個"彈性"會在更多的episode迭代之下慢慢去做調整，越變越嚴格(strict)，在這裡我們用 ϵ-greedy policies，也就是說大部分時候我們依舊是用所預測的action value 的最大值，但是在 ϵ 的機率會做隨機選擇。也就是在各個非-greedy action 被 policy 選到的機率為 $\frac{\epsilon}{|\mathcal{A}(s)|}$，給予greedy-action則是 $1-\epsilon+\frac{\epsilon}{\mathcal{A}(s)}$。現在的 ϵ-greedy policy是 ϵ-soft 的一個例子，其定義為 $\pi(a|s) \geq \frac{\epsilon}{|\mathcal{A}(s)|}$ 在所有的state且 $\epsilon > 0$

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

Algorithm: parameter small $\epsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ϵ-soft policy

$Q(s, a) \in \mathbb{R}$ (arbitrary), for all $s \in \mathcal{S},\ a \in \mathcal{A}(s)$

$Return(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S},\ a \in \mathcal{A}(s)$

Loop forever(For each episode):

Generate an episode from $S_0,\ A_0$ 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, $t = T-1, T-2, \ldots, 1, 0$:

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

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 $Return(S_t, A_t)$

$Q(S_t, A_t) \leftarrow$ average $(Return(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

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