Chapter 13 Policy Gradient Method

Created	@January 11, 2024 8:19 AM
Tags	

這章節我們考慮一種學習的method 裡面有 policy parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ where $\pi(a|s,\boldsymbol{\theta}) = \Pr\{A_t = a|S_t = s, \boldsymbol{\theta}_t = \boldsymbol{\theta}\}$我們基於一些 表現(performance)衡量(measure)的 scalar(純量) 的gradient $J(\boldsymbol{\theta})$，這些 method希望可以有最佳(最大化)表現，所以她的更新是基於 gradient ascent in J

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_T + \alpha\widehat{\nabla J(\boldsymbol{\theta})}$$

where $\widehat{\nabla J(\boldsymbol{\theta})} \in \mathbb{R}^{d'}$ 是隨機性的預測(stochastic estimate)，它會嘗試著去近似(approximate) gradient 的 performance measure並遵從其 變數 $\boldsymbol{\theta}$。

13.1 Policy Approximation and its Advantages

在 Policy Gradient Method，一個policy 可以被參數化，只要 $\pi(a|s,\boldsymbol{\theta})$可以被相對應的參數進行微分，也就是說只要 $\nabla\pi(a|s,\theta)$ (column vector 是 $\pi(a|s,\boldsymbol{\theta})$ 對 θ 的偏微分) 存在且有限 在所有的 $s \in \mathcal{S}, a \in \mathcal{A}$ 及 $\boldsymbol{\theta} \in \mathbb{R}^{d'}$。一般來說，為了要有 exploration 我們需要確保 policy 並非 deterministic (i.e., $\pi(a|s,\boldsymbol{\theta}) \in (0,1)\text{ for all } s,a,\boldsymbol{\theta}$)。在這節我們提供了一個最常見的 parameterization(參數化) 的方法給離散的action space並會指出其給action value method 的優勢。Policy-based method也有方法可以搞定 continuous action space。

如過action space是離散且不要太大，那最常見能參數化的方法是去製作參數的數值的偏好 (form parameterized numerical preference) $h(s,a,\boldsymbol{\theta}) \in \mathbb{R}$ for each state-action pair，這個擁有著在這個state中最高偏好會給予最高的機率根據 指數型的 softmax 分布

$$\pi(a|s,\boldsymbol{\theta}) = \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}} \tag{13.2}$$

我們稱這個演算法式 soft-max in action preference

這個 action的偏好可以被任意的參數化，聚個例子它可以被D(A)NN所計算，這個 $\boldsymbol{\theta}$ 就是所有weight vector之間的所有連結

$$h(s,a,\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}(s,a) \tag{13.3}$$

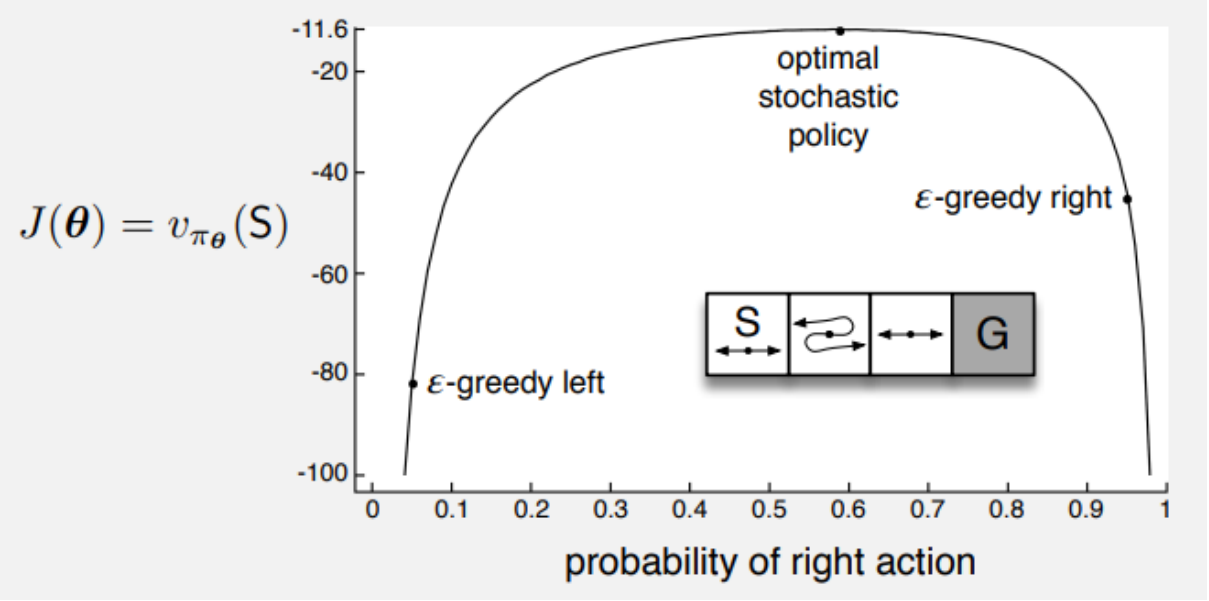
$\mathbf{x}(s,a)$ 就是我們在 9.5小節所建構的feature vector。

使用softmax的action偏好去參數化policies的其中一個優勢就是他的預測(approximate) policy 可以向 deterministic policy靠攏，也就是用 ϵ-greedy action 選擇 ，也就是說一直都會有 ϵ 的機率去選擇一個隨機的action，當然你也可以直接用 softmax 基於 action value 做的分布。不過這樣是無法達成 deterministic policy的。而這個 action value的預測值會收斂至真實值，當然會有點差異，會是其中一個特定的機率，而不是 0 或是 1。如果 softmax 的 參數包含一個叫做 temperature 的參數，這個 temperature 會持續減少去達到 determisim(決定性)，不過事實上你很難在沒有任何資訊的情況下選擇一個合理的消退時間表，甚至設定一個初始的 temperature。Action的偏好是不一樣的因為它不會像特定值去靠攏，而他是被驅使去製造一個最佳化的具隨機性policy，如果這個policy是deterministic的話，這個偏好最佳 action會被無限的放大甩掉所有 次級 optimal policy。

第二個優勢是其根據 action 偏好的 softmax 可以讓選擇action的時候有著任意的機率。再有著重量級的 function approximation的時候，能夠最好的去預測這個policy的方法就是靠著隨機性(stochastic) ，舉個例子，在牌局中，非完整的資訊的最佳解常常會是做兩個不同的事有著特定的機率像是在牌局中虛張聲勢(bluffing)。 action value並沒有一個自然的方式可以去找到 stochastic optimal policies，因此 policy approximation可以。 像是下列的 Example 13.1

Example. 13.1 Short corridor with switch actions

假設有一個小型的gridworld如下圖，reward一樣為 -1 per step，首先 第一格跟第三格是一樣的，往左就是左，往右就是右(第一個state向左會導致下個state依舊在原地)，但是第二格的Action選擇右會導致你往左移、選擇向左會導致你向右移動。這是個很困難的問題是因為所有的state在 function approximation看起來都是一樣的。尤其是我們定義所有的 s ： $\mathbf{x}(s,\text{right}) = [1,0]^\top$和 $\mathbf{x}(s,\text{left}) = [0,1]^\top$ 。一個 action-value 的 method有著 ϵ-greedy action的選擇會被迫選擇兩個policy之一，要不是選擇右會有高機率 $1-\epsilon/2$或是選擇左也有相同高的機率，如果 $\epsilon = 0.1$，這兩個policy在起始點會有一個value -44 及 -82 對應向右policy 及向左 policy如下圖，不過method的表現會大幅增加如果它可以學習一個特定的機率會讓這個例子達到-11.6的值。



或許policy parameterization相較於action value parameterization所具有的最簡單的優勢就是，policy可以用更簡單的函數來做近似(approximate)。兩者之間的問題在於複雜度不同。對某些人來說，action-value function是比較簡單的，因為它們更容易近似。但對某些人來說，policy才是比較簡單的。通常policy-based method會學的比較快，並產生更好的漸近策略(asymptotic policy)(as in Tetris; see S¸im¸sek, Alg´orta, and Kothiyal, 2016)。

最後，我們注意到，policy parameterization的選擇有時候是將關於你所期望的策略形式的先驗知識注入強化學習系統的好方法。這通常是使用policy-based learning method的最重要的原因。

13.2 The Policy Gradient Theorem

除了對 ϵ-greedy policy上的優勢，還有在理論的基礎上，在連續的 policy parameterization下學習參數時 action機率改變也是平滑的，然而只要在action value做小小的變動，而如果這個變動使得其他地方state有了最大值， ϵ-greedy的 action 機率就可能會有較大的改變。尤其是說因為這個連續性才能使 policy-gradient method去近似 gradient ascent (13.1)。

這個 episodic 跟 continuous 對於performance measure $J(\boldsymbol{\theta})$的解釋不相同所以我們應該把它分開處理，不過我們還是嘗試結合兩者，所以我們將這重大理論的結果以一個系列的等式表示。

這個部份我們先將視角投向 episodic case，我們定義這個performance measure在這個episode的起始點，我們可以利用這個非隨機的 state s_0 去簡化表示法讓其不會去所失任何有意義的generality，那我們可以定義這個 performance of episodic case

$$J(\boldsymbol{\theta}) = v_{\pi_{\boldsymbol{\theta}}}(s_0) \tag{13.4}$$

where $v_{\pi_{\boldsymbol{\theta}}}$ 是 true value function of $\pi_{\boldsymbol{\theta}}$，以 $\boldsymbol{\theta}$ 定義的policy。在這裡我們會將其定義為沒有 discounting，但是我們仍將discounting 變數加入在psudeocode裡。

有了這個 function approximation，也許要讓這個policy保證每步都在進步會很困難，問題出在這個performance會因為 action 的選擇及 state 的分配機率，然後去影響 policy parameter。給定一個 state，policy parameter對action的效用以及reward可以被直接地從parameterization的知識去做計算。但是policy對state distribution的影響則是一個環境(environment)的函數，而這通常是未知的。當梯度是取決於策略變化對於狀態分佈的未知影響的時候，我們能夠如何估測關於策略參數的效能梯度(performance gradient)？

幸運的是，對於這個挑戰已經有一個很不錯的理論答案，也就是策略梯度理論，這個理論提供了一個關於策略參數的效能梯度的解析度(analytic expression)，這也是我們在近似梯度上升的時候所需要的 (13.1)，而這並不涉及狀態分佈的導數(derivative)。用於episodic case的policy gradient theorem如下：

Proof of the Policy Gradient Theorem (episodic case)

With just elementary calculus and re-arranging of terms, we can prove the policy gradient theorem from first principles. To keep the notation simple, we leave it implicit in all cases that π is a function of $\boldsymbol{\theta}$, and all gradients are also implicitly with respect to $\boldsymbol{\theta}$. First note that the gradient of the state-value function can be written in terms of the action-value function as

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r\,|\,s,a) \big(r + v_\pi(s')\big) \Big] \\
& && \text{(Exercise 3.19 and Equation 3.2)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'\,|\,s,a) \nabla v_\pi(s') \Big] && \text{(Eq. 3.4)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'\,|\,s,a) && \text{(unrolling)} \\
&\qquad \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''\,|\,s',a') \nabla v_\pi(s'') \big] \Big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}
$$

after repeated unrolling, where $\Pr(s \to x, k, \pi)$ is the probability of transitioning from state s to state x in k steps under policy π. It is then immediate that

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{(box page 199)} \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{(Eq. 9.3)} \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) && \text{(Q.E.D.)}
\end{aligned}
$$

💡 $p(s', r|s, a) = \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$
(3.2)

💡 $p(s'|s, a) = \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r\in\mathcal{R}} p(s', r|s, a)$
(3.4)

💡 $\eta(s) = h(s) + \sum_{\bar{s}} \eta(\bar{s}) \sum_a \pi(a|\bar{s})p(s|\bar{s}, a), \text{ for all } s \in \mathcal{S}$
(9.2)

💡 $\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')}, \text{ for all } s \in \mathcal{S}$
(9.3)

$$\nabla J(\boldsymbol{\theta}) \propto \sum_a \mu(s) \sum_a q_\pi(s, a) \nabla \pi(a|s, \boldsymbol{\theta}) \tag{13.5}$$

其中 gradients 是 column vector 以 $\boldsymbol{\theta}$ 的部件做部分微分， π 是 policy 對應 parameter vector $\boldsymbol{\theta}$，這個符號 $\propto$ 是與成正比，在 episodic case，這個會是episode 的長度，在continuing test這個會是1，因此，這關係實際上是相等的。這個機率分布 μ (第9及第10章) 是 on-policy 在 π 的分布。

13.3 REINFORCE: Monte Carlo Policy Gradient

現在你可以開始進入第一個 policy-gradient algorithm，我們先回想以前的stochastic gradient ascent (13.1)，他需要一個方法可以獲得樣本所以這個樣本的梯度的期望值跟實際上參數的 performance measure 的梯度成正比 ，這個樣本梯度只要跟policy的梯度成正比叫好因為它可以被step size所吸收，否則 step size就會是任意值，這個 policy gradient theorem 也有跟 gradient 成正比的部分，所以我們只需要一個方法去同樣一個期望值去approximate 這個 ，policy gradient theorem右半邊是state weight 的總和，這state weight則是由他在policy π 多常被訪問的比例去做分配

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \\ &= \mathbb{E}_\pi \Big[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \Big]\end{aligned} \tag{13.6}$$

我們可以在這裡將 stochastic gradient-ascent 具體化

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta}) \tag{13.7}$$

其中 $\hat{q}$ 是對 q_π 的 approximation，這個演算法被稱做 all-action method 因為她的update包含了所有有關的action，不過目前我們只包含了 A_t，其中一個 action 在時間 t 的時候被採取。

我們繼續推導我們的REINFORCE，在此我們提出了 A_t 跟以前我們提出了 S_t 一樣 (13.6)，以隨機的變數去做替代以 policy π 下的期望值，再從這個期望值去採樣。(13.6)包含了一個合適的 sum over actions 但是各項並不是以 $\pi(a|s,\boldsymbol{\theta})$ 作為比重，所以我們乘上了這個weight 但是不改變方程式的等於，做法就是前面將其乘上後面再除以 $\pi(a|s,\boldsymbol{\theta})$ 我們就有了下面式子

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) \propto & \mathbb{E}_\pi \Big[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})} \Big] \\ = & \mathbb{E}_\pi \Big[q_\pi(S_t,A_t) \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \Big] \ (\text{replace } a \text{ by the sample } A_t \sim \pi) \\ = & \mathbb{E}_\pi \Big[G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \Big] \ (\text{because } \mathbb{E}_\pi[G_t|S_t,A_t] = q_\pi(S_t,A_t))\end{aligned}$$

where G_t 也是一般的return，這個就是我們想要的，一個可以在每個時間點被採樣的”量”(quantity) 其期望值跟梯度成正比，用這個我們就可以推到出我們的generic stochastic gradient ascent 演算法 (13.1) 得到我們的 REINFORCE update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha G_t \frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

這個更新非常的直觀(intuitive appeal)，每個遞增是跟 G_t 與 向量 成正比，也就是機率使用這個 action的梯度除以機率使用這個action，這個向量是在參數的領域裡的方向，也就是說未來重複在 S_t 時會採取 A_t 的增加量，**這個更新會增加 參數向量上的參數向量會與return成正比，然後會跟action probability成反比**。前面那句是有道理的，因為這會導致參數往那個能夠生成最高return的actions的方向移動。後面那句也是有道理的，不然一直被選到的那些actions是處於優勢的(更新會頻繁地往它們的方向移動)，即使它們沒有產生最高的return也有可能會勝出的。

注意到，REINFORCE使用的是從時間 t 開始的完整的return，這包含episode結束之前的所有的未來的報酬(future rewards)。從這個意義上來說，REINFORCE是一種 MC 演算法，而且僅適用於episodic case(就像Chapter 5一樣，在episode結束之後再回頭去做更新)。下面給出完整的演算法的pseudocode：

REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for π_*

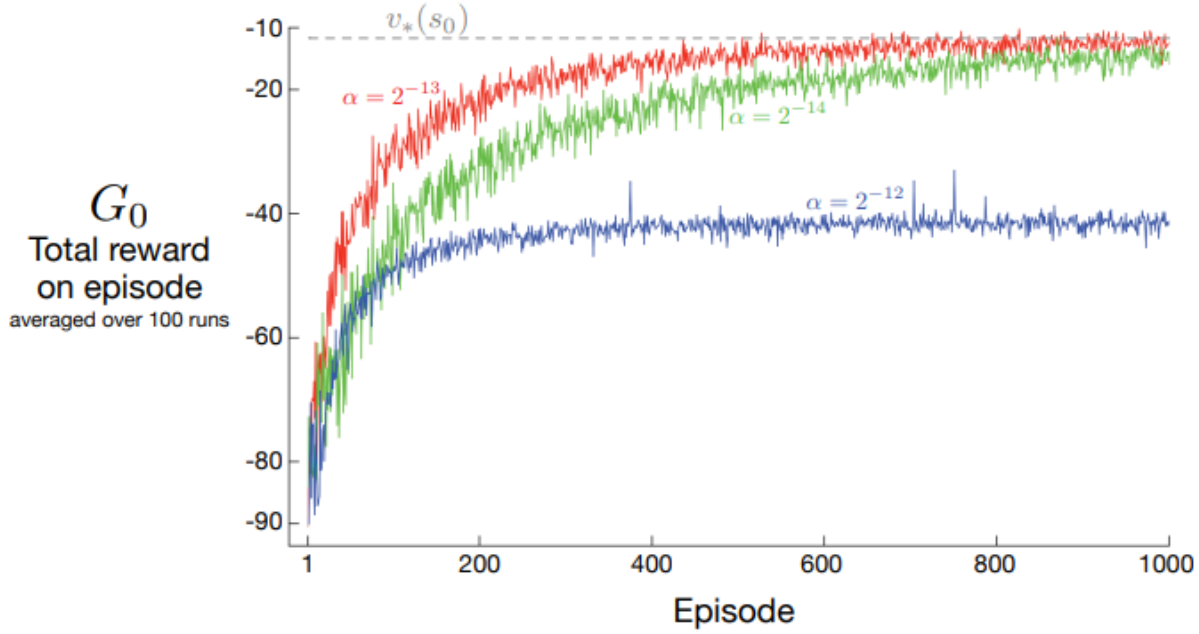
Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$

我們注意到最後一個更新跟 (13.8) 有點部大一樣

第一個不同的地方是pseudocode使用的是 $\nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$替代 $\frac{\nabla \pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$因為 $\nabla \ln x = \frac{\nabla x}{x}$，這個向量已經在論文中有很多不同的名字跟標註了，不過我們在這裡稱他為 eligibility vector，注意一下這個policy parameter只會在這個演算法出現。

第二個不同的地方是它包含了 γ^t ，因為較早的時候我們有提到說我們是用 $\gamma = 1$ 的情況下在處理這些演算法。



做為一個隨機梯度方法，REINFORCE有著很好的理論收斂性。根據其結構，episode的expected update會跟performance gradient(效能梯度)的方向相同。這確保在足夠小的 α的情況下，其預期效能的改進，並且在減少 α 的標準隨機近似條件下收斂到局部最佳。REINFORCE做為一種Monte Carlo method，它可能會有高方差，這會導致它學習上會慢一點點。

EX 13.3證明

$$\nabla \ln \pi(a|s,\boldsymbol{\theta}) = \mathbf{x}(s,a) - \sum_b \pi(b|s,\boldsymbol{\theta})\mathbf{x}(s,b) \tag{13.9}$$

13.4 REINFORCE with Baseline

policy gradient theorem 可以被 generalized ，跟一個任意的 baseline $b(s)$ 去做比較。

$$\nabla J(\boldsymbol{\theta}) \propto \sum_a \mu(s) \sum_a (q_\pi(s,a) - b(s)) \nabla \pi(a|s,\boldsymbol{\theta}) \tag{13.10}$$

baseline 可以是任意一個function，只要他不會因為 a 而改變值。等式仍然有效既使其相減為0

$$\sum_a b(s)\nabla\pi(a|s,\boldsymbol{\theta}) = b(s)\nabla\sum_a \pi(a|s,\boldsymbol{\theta}) = b(s)\nabla 1 = 0$$

policy gradient theorem with baseline (13.10) 可以被用來推導出一個更新的規則，這個更新法則就可以產生新的REINFORCE 包含一個 general base

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\Big(G_t - b(S_t)\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)}$$

因為這個 baseline 可能都是為 0，這個update是嚴格的generalization of REINFORCE，一般來說，baseline不會讓期望值更新改變，但是她可能會較大的影響標準差。MDP的baseline必須要依照state去做調整，在一些state 任意 action都會回饋較高的reward，此時就需要一個高的baseline去做generalization，同理低return 就要一個低的baseline。一個常見的挑選方法就是以現在的 estimated state value $\hat{v}(S_t,\mathbf{w})$。

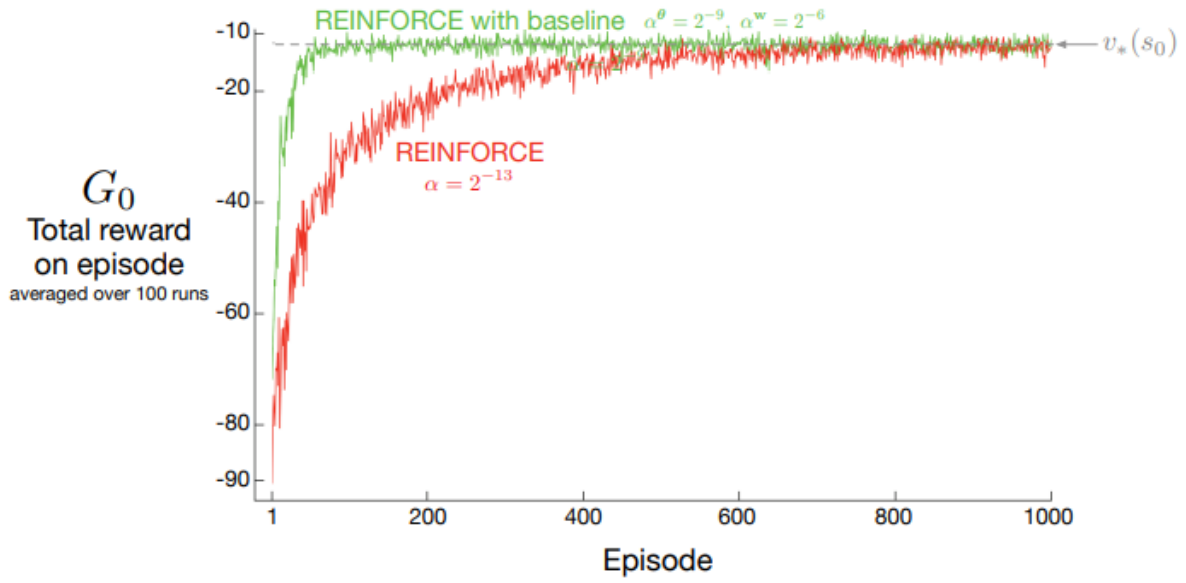
因為 REINFORCE 是 MC method學習policy參數 $\boldsymbol{\theta}$，不過感覺我們會很自然地用 MC 去學習 state-value weight。

REINFORCE with Baseline (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
  Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$ $\qquad (G_t)$
    $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S_t,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\gamma^t\delta\nabla\ln\pi(A_t|S_t,\boldsymbol{\theta})$

這個演算法有兩個 step-size $\alpha^{\mathbf{w}}$ 及 $\alpha^{\boldsymbol{\theta}}$ ($\alpha^{\boldsymbol{\theta}}$ 是 (13.11)的 α)，而 $\alpha^{\mathbf{w}}$的選擇則是相對較簡單，在線性的情況我們可以照著先人的經驗設定為 $\alpha^{\mathbf{w}} = 0.1/\mathbb{E}[||\nabla\hat{v}_t(S_t,\mathbf{w})||^2_\mu]$ (見 Section 9.6)，然而 $\alpha^{\boldsymbol{\theta}}$要怎麼設定會比較模糊，他會依賴於reward的標準差跟on-policy 的參數化。



上圖描述加上baseline可以使REINFORCE學習速度較快。

13.5 Actor-Critic Method

在 REINFORCE with baseline 這個用於學習的 state value function只會預估 state 轉換(transition)的下一個 state，這個預測為了後面的序列(sequence)設立了 baseline(基準線)，不過這是在action之前所估測的，因此沒有辦法用來評估action。另一方面在 actor-critic method state-value function會應用在第二個 transition state，這個第二個State的預測值做過discount並加上reward構成了 one-step return $G_{t:t+1}$，可以去較好的去預測真實 return 不管是從標準差的角度還是從計算上的角度來說都較有優勢，既使這樣做會使其偏差。我們真到可以去利用 n-step return 跟 eligibility trace 去調整 bias(Chapter 7 & 12)。剛 state-value function 被使用去評估 action我們就把他稱之為 critic，然後整體上來說policy gradient method就是actor-critic method，值得提醒的是這個梯度預測的偏差並不是 bootstrapping所造成的；actor 仍然會有 bias既使critic是用MC method中學習的。

第一我們考慮 one-step actor-critic method，我們在第六章談到了 TD(0), Sarsa(0), Q-learning，這個最方便的地方是他們都可以完全的 online 跟漸進式，避免了 eligibility trace 的複雜度，他們都是 eligibility trace 的特殊例子，不過這些例子較容易使人理解。One-step actor-critic methods 替換掉 full return of REINFORCE (13.11) with one step return(然後使用他學習的 state value function 建立 baseline) 如下

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\Big(G_{t:t+1} - \hat{v}(S_t, \mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \tag{13.12}$$

$$= \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \tag{13.13}$$

$$= \boldsymbol{\theta}_t + \alpha\delta_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})} \tag{13.14}$$

我們自然可以把 state-value-function learning method和 semi-gradient TD(0) 做結合，現在她可以完全online，漸進演算法

One-step Actor–Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\delta\nabla\ln\pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

這個 generalization 到 forward-view(12章) n-step method然後推展至 λ-return演算法是很直接的，僅僅只有從 (13.12)將return替換為 $G_{t:t+n}$或是 G_t^λ，這個backward view of the λ-return 演算法也是很直覺，就是將 actor 跟 critic 分開 eligibility trace如章節12一樣。

Actor–Critic with Eligibility Traces (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla \ln\pi(A|S,\boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

13.6 Policy Gradient for Continuing Problem

如 10.3 所說的，continuous problem需要去定義performance才能得到每個timestep的平均reward

$$
\begin{aligned}
J(\boldsymbol{\theta}) = r(\pi) &= \lim_{h\to\infty}\frac{1}{h}\sum_{t=1}^{h}\mathbb{E}[R_t|S_0, A_{0:t-1}\sim\pi] \\
&= \lim_{t\to\infty}\mathbb{E}[R_t|S_0, A_{0:t+1}\sim\pi] \\
&= \sum_s \mu(s)\sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)r
\end{aligned}
\tag{13.15}
$$

μ 就是 π 的state分布

$\mu(s) = \lim_{t\to\infty}\Pr\{S_t = s|A_{0:t}\sim\pi\}$，其跟 S_0 具有依賴性(假設其具有遍歷性ergodicity) 還我們要記得，這是一個特殊的分佈，如果你根據 π 去做選擇，你都會落在同一個分布中。

$$\sum_s \mu(s) \sum_a \pi(a|s,\boldsymbol{\theta}) p(s'|s,a) = \mu(s'), \text{ for all } s' \in \mathcal{S} \tag{13.16}$$

Actor–Critic with Eligibility Traces (continuing), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: $\lambda^{\mathbf{w}} \in [0,1]$, $\lambda^{\boldsymbol{\theta}} \in [0,1]$, $\alpha^{\mathbf{w}} > 0$, $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\bar{R}} > 0$
Initialize $\bar{R} \in \mathbb{R}$ (e.g., to 0)
Initialize state-value weights $\mathbf{w} \in \mathbb{R}^d$ and policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)
Initialize $S \in \mathcal{S}$ (e.g., to s_0)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d-component eligibility trace vector)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
Loop forever (for each time step):
  $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
  Take action A, observe S', R
  $\delta \leftarrow R - \bar{R} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}} \delta$
  $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}} \mathbf{z}^{\mathbf{w}} + \nabla \hat{v}(S,\mathbf{w})$
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}} \mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \mathbf{z}^{\mathbf{w}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \delta \mathbf{z}^{\boldsymbol{\theta}}$
  $S \leftarrow S'$

在 continuing case 我們定義 values $v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$ 及 $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$ 的相對差分回報(differential return)為：

$$G_t = R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \dots . \tag{13.17}$$

有了這個替代的定義，這個policy gradient theorem 可以同時給 episodic 及 continuing case證明如下。

Proof of the Policy Gradient Theorem (continuing case)

The proof of the policy gradient theorem for the continuing case begins similarly to the episodic case. Again we leave it implicit in all cases that π is a function of $\boldsymbol{\theta}$ and that the gradients are with respect to $\boldsymbol{\theta}$. Recall that in the continuing case $J(\boldsymbol{\theta}) = r(\pi)$ (13.15) and that v_π and q_π denote values with respect to the differential return (13.17). The gradient of the state-value function can be written, for any $s \in \mathcal{S}$, as

$$\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{for all } s \in \mathcal{S} && \text{(Exercise 3.18)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \Big] && \text{(product rule of calculus)} \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a) \big(r - r(\boldsymbol{\theta}) + v_\pi(s') \big) \Big] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \big[-\nabla r(\boldsymbol{\theta}) + \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \big] \Big].
\end{aligned}$$

After re-arranging terms, we obtain

$$\nabla r(\boldsymbol{\theta}) = \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] - \nabla v_\pi(s).$$

Notice that the left-hand side can be written $\nabla J(\boldsymbol{\theta})$, and that it does not depend on s. Thus the right-hand side does not depend on s either, and we can safely sum it over all $s \in \mathcal{S}$, weighted by $\mu(s)$, without changing it (because $\sum_s \mu(s) = 1$):

$$\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \sum_s \mu(s) \left(\sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] - \nabla v_\pi(s) \right) \\
&= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&\quad + \sum_s \mu(s) \sum_a \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s) \\
&= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&\quad + \sum_{s'} \underbrace{\sum_s \mu(s) \sum_a \pi(a|s) p(s'|s,a)}_{\mu(s') \text{ (13.16)}} \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s) \\
&= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) + \sum_{s'} \mu(s') \nabla v_\pi(s') - \sum_s \mu(s) \nabla v_\pi(s) \\
&= \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a). \qquad \text{Q.E.D.}
\end{aligned}$$

13.7 Policy Parameterization for Continuing Actions

policy-based method 提供了實際的方法解決大 action space的問題，甚至是連續的state並有著無限種action，電腦會去學習這些action機率分布，舉個例子，action set可能會是實數的情況，action的選擇為高斯分布

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{13.18}$$

μ 跟 σ 是常態分佈的標準差， $p(x)$ 為機率"密度" 而非機率本身，所以他可以大於1。通常我們對 $p(x)$ 任意範圍進行微分可以取得 x 落在範圍內的機率。

為了能夠生成policy parameterization，我們可以將policy定義為real-valued scalar action(實數純量動作?)上的正態分佈機率密度，其均值與標準差則由取決於state的parametric function approximation (參數函數近似)所給出。如下：

$$\pi(a|s,\boldsymbol{\theta}) = \frac{1}{\sigma(s,\boldsymbol{\theta}\sqrt{2\pi})} \exp\left(-\frac{(a-\mu(s,\boldsymbol{\theta}))^2}{2\sigma(s,\boldsymbol{\theta})^2}\right) \tag{13.19}$$

where $\mu : \mathcal{S} \times \mathbb{R}^{d'} \to \mathbb{R}$ and $\sigma : \mathcal{S} \times \mathbb{R}^{d'} \to \mathbb{R}^+$是兩種 parameterized function approximation，為了完成這個範例，我們會需要給這些approximatior 一個形式。為此，我們把policy的參數向量分成兩部份，也就是 $\boldsymbol{\theta} = [\boldsymbol{\theta}_\mu, \boldsymbol{\theta}_\sigma]^\top$ ，一部分是用於approximate 平均，一部分則是用 approximate 標準差，這個平均可以被視為線性函數去做approximation，而標準差則必為正數，較適合用線性函數的 exponential 去做approximation。

$$\mu(s,\boldsymbol{\theta}) = \boldsymbol{\theta}_\mu^\top \mathbf{w}_\mu(s) \quad \text{and} \quad \sigma(s,\boldsymbol{\theta}) = \exp(\boldsymbol{\theta}_\sigma^\top \mathbf{x}_\sigma(s)) \tag{13.20}$$

where $\mathbf{x}_\mu(s)$ 和 $\mathbf{x}_\sigma(s)$ 是 feature vector 以 Section 9.5的方法去建構。有了這些定義，所有的演算法就可以被應用在學習、應用在 real-valued actions。