1. 背景

- actor 指的是 policy update，因为策略是用来采取某些动作的，所以其被称为 actor
- critic 指的是 policy evaluation， 它通过计算 action value 或 state value 来评估一个策略好还是不好

让我们来回顾一下 policy gradient 算法：

（1） 首先我们有一个度量的指标$J(\theta)$，它可以是$\bar{v}_\pi$或$\bar{r}_\pi$

（2） 采用梯度上升算法来使$J(\theta)$最大化：

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta J(\theta_t) = \theta_t + \alpha E_{S\sim\eta, A\sim\pi}[\nabla_\theta \ln \pi(A|S,\theta_t)\, q_\pi(S,A)]$$

（3） 使用 stochastic gradient 替换 true gradient：

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \ln \pi(a_t|s_t,\theta_t)\, q_t(s_t,a_t)$$

- 这个算法对应的就是 actor
- 算法中近似估计$q_t(s_t,a_t)$对应的就是 critic

2. 最简单的 Actor-Critic 算法（QAC）

有多种不同的方法来近似计算q_π：

- 蒙特卡洛方法进行近似，上一章中已经介绍
- TD 算法进行近似，其伪代码为：

Aim: Search for an optimal policy by maximizing $J(\theta)$.

At time step t in each episode, do

Generate a_t following $\pi(a|s_t,\theta_t)$, observe r_{t+1}, s_{t+1}, and then generate a_{t+1} following $\pi(a|s_{t+1},\theta_t)$.

Critic (value update):

$$w_{t+1} = w_t + \alpha_w\big[r_{t+1} + \gamma q(s_{t+1},a_{t+1},w_t) - q(s_t,a_t,w_t)\big]\nabla_w q(s_t,a_t,w_t)$$

Actor (policy update):

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_\theta \ln \pi(a_t|s_t,\theta_t) q(s_t,a_t,w_{t+1})$$

- 该算法中的 critic 对应的就是 Sarsa 算法+值函数近似
- 该算法中的 actor 对应的就是策略梯度方法+值函数近似
- 该算法是 on-policy 的，即与环境交互产生经验的策略和进行更新的策略是相同的
- 由于 softmax 的存在，策略是 stochastic，所以不需要$\varepsilon - greedy$

3. Advantage Actor-Critic 算法（A2C）

它的基本思想是：在 QAC 的基础上引入偏置量来减小估计的方差

首先介绍一个性质：policy gradient 对于引入一个新的偏置值是不会发生变化的：

$$\nabla_\theta J(\theta) = E_{S\sim\eta,\,A\sim\pi}[\nabla_\theta\, ln\, \pi(A|S,\theta)\, q_\pi(S,A)]$$

$$= E_{S\sim\eta,\,A\sim\pi}[\nabla_\theta\, ln\, \pi(A|S,\theta)\, (q_\pi(S,A) - b(S))]$$

$b(S)$是 S 的一个标量函数，为什么引入一个新的偏置值不会发生变化？

要使该等式成立，则要有：

$$E_{S\sim\eta,\,A\sim\pi}[\nabla_\theta\, ln\, \pi(A|S,\theta)\, b(S)] = 0$$

计算过程为：

$$E_{S\sim\eta,\,A\sim\pi}[\nabla_\theta\, ln\, \pi(A|S,\theta)\, b(S)] = \sum_{s\epsilon S} \eta(s) \sum_{a\in A} \pi(a|s,\theta) \nabla_\theta\, ln\, \pi(a|s,\theta)\, b(s)$$

$$= \sum_{s\epsilon S} \eta(s) \sum_{a\in A} \nabla_\theta \pi(a|s,\theta) b(s)$$

$$= \sum_{s\epsilon S} \eta(s) b(s) \sum_{a\in A} \nabla_\theta \pi(a|s,\theta)$$

$$= \sum_{s\epsilon S} \eta(s) b(s) \nabla_\theta \sum_{a\in A} \pi(a|s,\theta)$$

$$= \sum_{s\epsilon S} \eta(s) b(s) \nabla_\theta 1 = 0$$

引入一个偏置值有什么作用?

令$\nabla_\theta J(\theta)=E[X]$，则$X(S,A)=\nabla_\theta \ln\pi(A|S,\theta)\,[q_\pi(S,A)-b(S)]$

- 由上述证明可知，$E[X]$与$b(S)$无关
- 但是，$D[X]$并不是与$b(S)$无关（证明省略）

我们的目标就是找到最优的偏置值b，使得方差$D[X]$达到最小

最优的偏置值b的定义为（证明省略）:

$$b(s)=\frac{E_{A\sim\pi}[\|\nabla_\theta \ln\pi(A|s,\theta)\|^2 q(s,A)]}{E_{A\sim\pi}[\|\nabla_\theta \ln\pi(A|s,\theta)\|^2]}\ for\ \forall s\in S$$

但是它过于复杂，所以我们去除权重$\|\nabla_\theta \ln\pi(A|s,\theta)\|^2$，得到

$$b(s)=E_{A\sim\pi}[q(s,A)]=v_\pi(s)$$

将该 baseline 引入到 AC 算法中，得到:

$$\begin{aligned}\theta_{t+1}&=\theta_t+\alpha E\big[\nabla_\theta \ln\pi(A|S,\theta_t)\,[q_\pi(S,A)-v_\pi(S)]\big]\\&=\theta_t+\alpha E[\nabla_\theta \ln\pi(A|S,\theta_t)\,\delta_\pi(S,A)]\end{aligned}$$

其中$\delta_\pi(S,A)=q_\pi(S,A)-v_\pi(S)$被称为 advantage function（优势函数）

该算法的 stochastic 版本为:

$$\begin{aligned}\theta_{t+1}&=\theta_t+\alpha\nabla_\theta \ln\pi(a_t|s_t,\theta_t)\,[q_t(s_t,a_t)-v_t(s_t)]\\&=\theta_t+\alpha\nabla_\theta \ln\pi(a_t|s_t,\theta_t)\,\delta_t(s_t,a_t)\end{aligned}$$

将该算法进行重新组织，表示为:

$$\begin{aligned}\theta_{t+1}&=\theta_t+\alpha\nabla_\theta \ln\pi(a_t|s_t,\theta_t)\,\delta_t(s_t,a_t)\\&=\theta_t+\alpha\frac{\nabla_\theta\pi(a_t|s_t,\theta_t)}{\pi(a_t|s_t,\theta_t)}\delta_t(s_t,a_t)\\&=\theta_t+\alpha\frac{\delta_t(s_t,a_t)}{\pi(a_t|s_t,\theta_t)}\nabla_\theta\pi(a_t|s_t,\theta_t)\end{aligned}$$

将$\frac{\delta_t(s_t,a_t)}{\pi(a_t|s_t,\theta_t)}$称为 step size，它能够很好地平衡 exploration 和 exploitation，理由与上一章相同

δ_t比q_t更好，因为$\delta_\pi(S,A)=q_\pi(S,A)-v_\pi(S)$，而$v_\pi(s)=E[q_\pi(s,A)|S=s]$，它实际上是$q_\pi(s,A)$的平均值，因此$\delta_t$实际上表示了 action value 的相对值

δ_t可以近似表示 TD error：

$$\delta_t = q_t(s_t, a_t) - v_t(s_t) \rightarrow r_{t+1} + \gamma v_t(s_{t+1}) - v_t(s_t)$$

原因如下：

$$E[q_\pi(S,A) - v_\pi(S)|S = s_t, A = a_t] = E[R + \gamma v_\pi(S') - v_\pi(S)|S = s_t, A = a_t]$$

这样做的好处是：如果采用原计算方法，需要训练两个神经网络，一个用来近似计算v_t，另一个用来近似计算q_t；而这样替换后，只需要训练一个神经网络，用来近似计算v_t

这样，我们就得到 A2C 算法的伪代码为：

Aim: Search for an optimal policy by maximizing $J(\theta)$.

At time step t in each episode, do

Generate a_t following $\pi(a|s_t, \theta_t)$ and then observe r_{t+1}, s_{t+1}

TD error (advantage function):

$$\delta_t = r_{t+1} + \gamma v(s_{t+1}, w_t) - v(s_t, w_t)$$

Critic (value update):

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w v(s_t, w_t)$$

Actor (policy update):

$$\theta_{t+1} = \theta_t + \alpha_\theta \delta_t \nabla_\theta \ln \pi(a_t|s_t, \theta_t)$$

4. Importance sampling（重要性采样）

我们前面介绍的算法实际上都是 on-policy 的，那么有没有办法将算法转换为 off-policy 呢？

考虑这样一个例子：

有随机变量$X \in \{-1, +1\}$，它的概率分布p_0为：

$$p_0(X = +1) = 0.5, p_0(X = -1) = 0.5$$

则其期望值为：

$$E_{X \sim p_0} = (+1) \times 0.5 + (-1) \times 0.5 = 0$$

如何通过采样$\{x_i\}$来近似计算期望值?

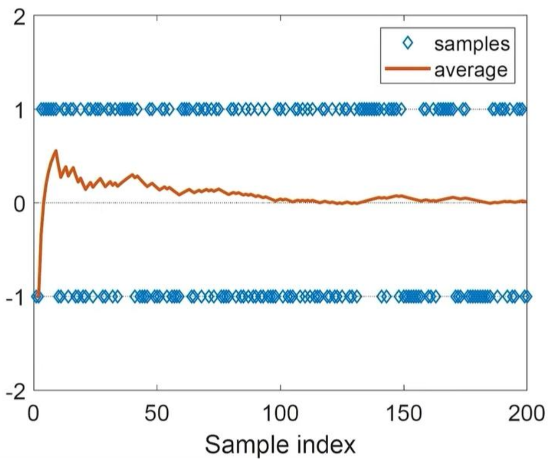
（1） 通过大数定理来进行近似计算

$\{x_i\}$是根据p_0生成的采样序列，则有:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \to E[X] \ as \ n \to \infty$$

$$E[\bar{x}] = E[X], D[\bar{x}] = \frac{1}{n}D[X]$$

结果如图所示:



（2） 通过重要性采样技术来进行计算

假设采样序列$\{x_i\}$是根据另一种分布p_1来进行生成的，p_1为:

$$p_1(X=+1) = 0.8, p_1(X=-1) = 0.2$$

则其期望值为:

$$E_{X\sim p_1} = (+1)\times 0.8 + (-1)\times 0.2 = 0.6$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \to E_{X\sim p_1}[X] = 0.6 \neq E_{X\sim p_0}[X]$$

我们能否用$\{x_i\}\sim p_1$来近似估计$E_{X\sim p_0}[X]$呢?

为什么要这样做：我们想使用 behavior policy 产生的经验来估计$E_{A\sim\pi}[*]$，其中π为 target policy

$$E_{X\sim p_0}[X]=\sum_x p_0(x)x=\sum_x p_1(x)\frac{p_0(x)}{p_1(x)}x=\sum_x p_1(x)f(x)=E_{X\sim p_1}[f(X)]$$

其中，

$$f(x)=\frac{p_0(x)}{p_1(x)}x$$

这样我们可以通过近似估计$E_{X\sim p_1}[f(X)]$来近似估计$E_{X\sim p_0}[X]$

如何近似估计$E_{X\sim p_1}[f(X)]$？令

$$\bar{f}=\frac{1}{n}\sum_{i=1}^{n}f(x_i)\,,where\ x_i\sim p_1$$

则有：

$$E_{X\sim p_1}[\bar{f}]=E_{X\sim p_1}[f(X)],D_{X\sim p_1}[\bar{f}]=\frac{1}{n}D_{X\sim p_1}[f(X)]$$

因此可以使用$\bar{f}$来近似估计$E_{X\sim p_1}[f(X)]=E_{X\sim p_0}[X]$，

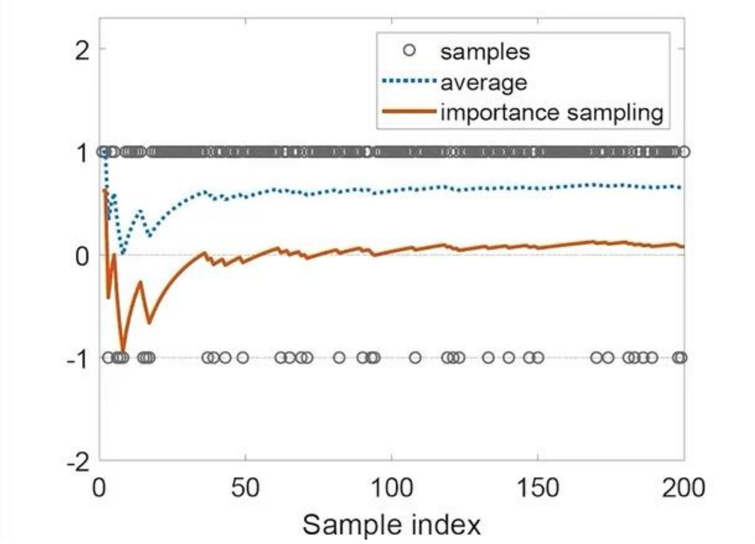
$$E_{X\sim p_0}[X]\approx\bar{f}=\frac{1}{n}\sum_{i=1}^{n}f(x_i)=\frac{1}{n}\sum_{i=1}^{n}f(x_i)=\frac{1}{n}\sum_{i=1}^{n}\frac{p_0(x_i)}{p_1(x_i)}x_i$$

其中，$\frac{p_0(x_i)}{p_1(x_i)}$被称为 importance weight

我们可能会产生这样一个疑问：$\bar{f}=\frac{1}{n}\sum_{i=1}^{n}\frac{p_0(x_i)}{p_1(x_i)}x_i$需要$p_0(x)$，如果我们知道$p_0(x)$，为什么不直接计算$E_{X\sim p_0}[X]$？

因为它适用于以下情形：给定x能够计算出$p_0(x)$，但是难以计算期望。比如连续情况下，需要计算积分；p_0的表达式十分复杂，难以计算；p_0没有表达式，例如神经网络

通过重要性采样近似计算另一种分布的例子为:



5. off-policy actor-critic 算法

将重要性采样技术运用到 actor-critic 算法中:

假设β是用来与环境交互产生经验的 behavior policy，则我们的目标是去更新 target policy π，使得以下 metric 达到最大值

$$J(\theta)=\sum_{s\in S} d_\beta(s) v_\pi(s)=E_{S\sim d_\beta}[v_\pi(S)]$$

其中d_β是策略β下的 stationary distribution

在 discounted case（即$\gamma\in(0,1)$）下，对该目标函数进行求导得到:

$$\nabla_\theta J(\theta)=E_{S\sim\rho,\,A\sim\beta}\left[\frac{\pi(A|S,\theta)}{\beta(A|S)}\nabla_\theta \ln\pi(A|S,\theta)\, q_\pi(S,A)\right]$$

其中β为 behavior policy，ρ为状态分布，$\frac{\pi(A|S,\theta)}{\beta(A|S)}$实际上就是 importance weight，$\pi(A|S,\theta)$对应$p_0$，$\beta(A|S)$对应$p_1$

我们仍然可以给它加上一个 baseline 来减小方差:

$$\nabla_\theta J(\theta)=E_{S\sim\rho,\,A\sim\beta}\left[\frac{\pi(A|S,\theta)}{\beta(A|S)}\nabla_\theta \ln\pi(A|S,\theta)\,[q_\pi(S,A)-b(S)]\right]$$

取$b(S) = v_\pi(S)$，则有：

$$\nabla_\theta J(\theta) = E_{S\sim\rho,\, A\sim\beta}\left[\frac{\pi(A|S,\theta)}{\beta(A|S)}\nabla_\theta \ln \pi(A|S,\theta)\,[q_\pi(S,A) - v_\pi(S)]\right]$$

用 stochastic gradient 替换 true gradient，结果为：

$$\theta_{t+1} = \theta_t + \alpha_\theta \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)}\nabla_\theta \ln \pi(a_t|s_t,\theta_t)\,[q_t(s_t,a_t) - v_t(s_t)]$$

将后面部分近似估计为 TD error，则有：

$$q_t(s_t,a_t) - v_t(s_t) \approx r_{t+1} + \gamma v_t(s_{t+1}) - v_t(s_t) = \delta_t(s_t,a_t)$$

则算法表示为：

$$\theta_{t+1} = \theta_t + \alpha_\theta \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)}\nabla_\theta \ln \pi(a_t|s_t,\theta_t)\,\delta_t(s_t,a_t)$$

而由于：

$$\pi(a_t|s_t,\theta_t)\nabla_\theta \ln \pi(a_t|s_t,\theta_t) = \nabla_\theta \pi(a_t|s_t,\theta_t)$$

算法可以表示为：

$$\theta_{t+1} = \theta_t + \alpha_\theta \frac{\delta_t(s_t,a_t)}{\beta(a_t|s_t)}\nabla_\theta \pi(a_t|s_t,\theta_t)$$

其中$\frac{\delta_t(s_t,a_t)}{\beta(a_t|s_t)}$为 step size，在这里它并没有平衡 exploration 和 exploitation，因为分母由$\pi(a_t|s_t)$变为了$\beta(a_t|s_t)$

总算法的伪代码为：

Initialization: A given behavior policy $\beta(a|s)$. A target policy $\pi(a|s,\theta_0)$ where θ_0 is the initial parameter vector. A value function $v(s,w_0)$ where w_0 is the initial parameter vector.
Aim: Search for an optimal policy by maximizing $J(\theta)$.

At time step t in each episode, do
  Generate a_t following $\beta(s_t)$ and then observe r_{t+1}, s_{t+1}.
  TD error (advantage function):
    $\delta_t = r_{t+1} + \gamma v(s_{t+1}, w_t) - v(s_t, w_t)$
  Critic (value update):
    $w_{t+1} = w_t + \alpha_w \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)}\delta_t \nabla_w v(s_t, w_t)$
  Actor (policy update):
    $\theta_{t+1} = \theta_t + \alpha_\theta \frac{\pi(a_t|s_t,\theta_t)}{\beta(a_t|s_t)}\delta_t \nabla_\theta \ln \pi(a_t|s_t,\theta_t)$