强化学习基础篇（三十一）策略梯度(3)Actor-Critic算法

1.引入Baseline

在使用策略梯度方法更新过程中，降低方差的另一种方法是使用baseline。

在REINFORCE算法得到的更新方式为：

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R]=\mathbb{E}_\tau\left[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

其中的$G_t=\sum_{t'=t}^{T-1} r_t$是由轨迹产生的回报，具有很高的方差，如果考虑其上减去一个baseline $b(s)$：

$$\nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R]=\mathbb{E}_\tau\left[\sum_{t=0}^{T-1}\left(G_t-b\left(s_t\right)\right) \cdot \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

一般而言，baseline的选择可以是回报的期望：

$$b\left(s_t\right)=\mathbb{E}\left[r_t+r_{t+1}+\ldots+r_{T-1}\right]$$

Baseline的引入可以降低方差，但是有baseline不含有参数θ，所以不会改变更新过程的梯度：

$$\mathbb{E}_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) b\left(s_t\right)\right]=0$$

$$E_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\left(G_t-b\left(s_t\right)\right)\right]=E_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) G_t\right]$$

$$\operatorname{Var}_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\left(G_t-b\left(s_t\right)\right)\right]<\operatorname{Var}_\tau\left[\nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) G_t\right]$$

这里的baseline的选择还可以是一个另一个被w参数化的函数。

$$\nabla_\theta J(\theta)=\mathbb{E}_\tau\left[\sum_{t=0}^{T-1}\left(G_t-b_w\left(s_t\right)\right) \cdot \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

2、Vanilla Policy Gradient算法

通过加入baseline，我们可以得到Vanilla Policy Gradient算法:

procedure Policy Gradient(α)

Initialize policy parameters θ and baseline values $b(s)$ for all s, e.g. to 0

for iteration $= 1, 2, \dots$ **do**

Collect a set of m trajectories by executing the current policy π_θ

for each time step t of each trajectory $\tau^{(i)}$ **do**

Compute the *return* $G_t^{(i)} = \sum_{t'=t}^{T-1} r_{t'}$

Compute the *advantage estimate* $\hat{A}_t^{(i)} = G_t^{(i)} - b(s_t)$

Re-fit the baseline to the empirical returns by updating $\mathbf{w}$ to minimize

$$\sum_{i=1}^{m}\sum_{t=0}^{T-1} \|b(s_t) - G_t^{(i)}\|^2$$

Update policy parameters θ using the policy gradient estimate $\hat{g}$

$$\hat{g} = \sum_{i=1}^{m}\sum_{t=0}^{T-1} \hat{A}_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)})$$

with an optimizer like SGD ($\theta \leftarrow \theta + \alpha \cdot \hat{g}$) or Adam

return θ and baseline values $b(s)$

3、使用Critic降低方差

在实际中$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$更新过程的$G_t$可以使用动作值函数代替$Q^{\pi_\theta}\left(s_t, a_t\right)$，动作值函数作为Critic可以由参数化的函数近似：

$$Q_w(s, a) \approx Q^{\pi_\theta}(s, a)$$

所以策略梯度更新可以修改为：

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} Q_w\left(s_t, a_t\right) \cdot \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

这样就可以形成Actor-Critic算法，其中：

- Actor是策略函数，用于产生动作，其更新过程会根据Critic提供的方向进行策略参数θ的更新。
- Critic是价值函数，用于评估Actor产生动作的奖励，其更新过程会基于参数w更新。Critic相当于会评价通过Actor产生的动作。

如果使用线性函数进行Q函数的近似$Q_w(s, a) = \psi(s, a)^T \mathbf{w}$，然后使用$TD(0)$的方法更新Critic的参数$w$，使用PG更新Actor的参数$\theta$，这样就有简单的QAC算法：

Algorithm 2 Simple QAC

for each step **do**

generate sample s, a, r, s', a' following π_θ

$\delta = r + \gamma Q_{\mathbf{w}}(s', a') - Q_{\mathbf{w}}(s, a)$ #TD error

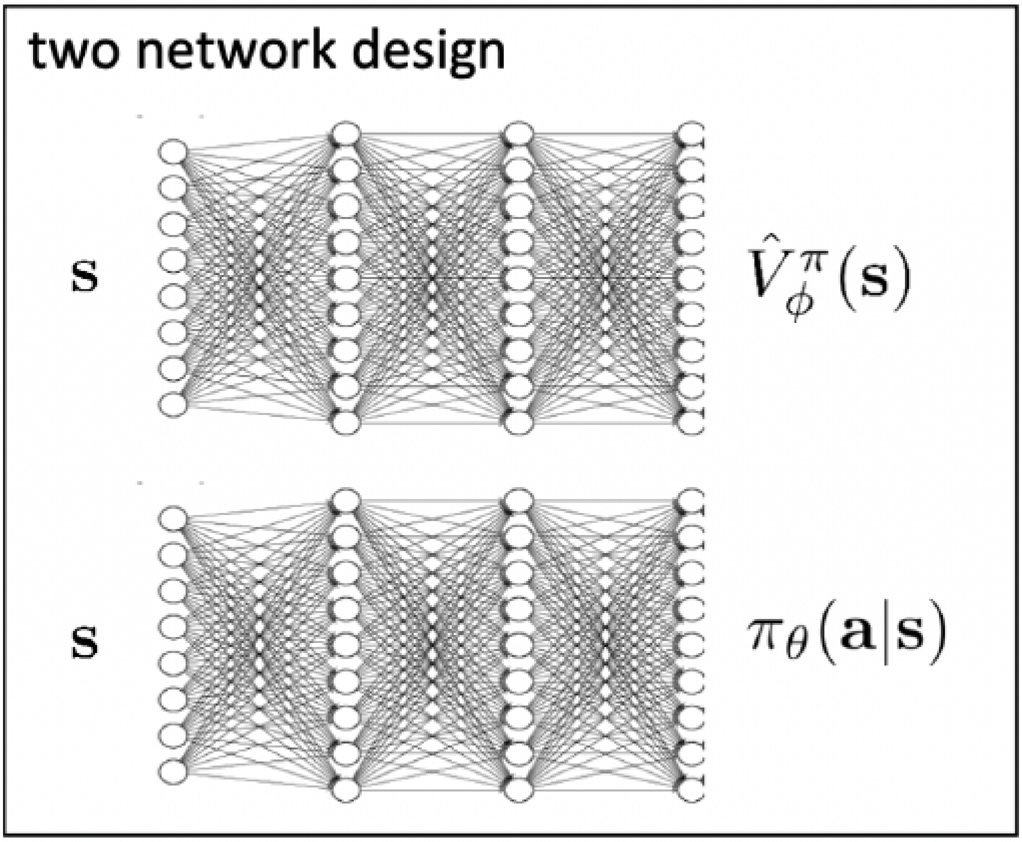
$\mathbf{w} \leftarrow \mathbf{w} + \beta \delta \psi(s, a)$

$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_{\mathbf{w}}(s, a)$

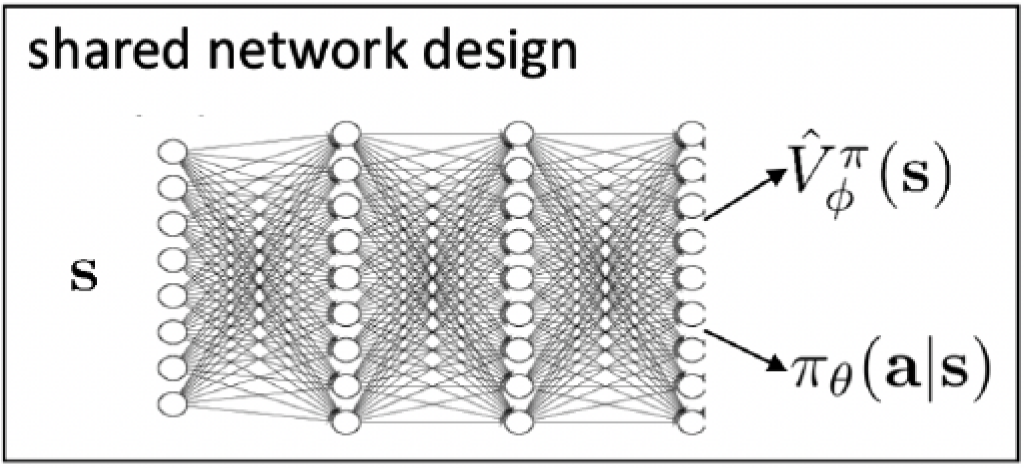
end for

4、Actor-Critc函数近似

在AC算法中，我们需要维护两组参数，在实现过程中可以由两种网络的设计，一种是分别使用神经网络拟合两组参数，第一组输出价值函数，第二组输出策略。



另一种方法是让两个输出共享同一个网络：



5、使用Baseline降低AC的方差

我们到Q函数的形式为：

$$Q^{\pi,\gamma}(s,a) = \mathbb{E}_\pi\left[r_1 + \gamma r_2 + \ldots \mid s_1 = s, a_1 = a\right]$$

价值函数为：

$$\begin{aligned} V^{\pi,\gamma}(s) &= \mathbb{E}_\pi\left[r_1 + \gamma r_2 + \ldots \mid s_1 = s\right] \\ &= \mathbb{E}_{a\sim\pi}\left[Q^{\pi,\gamma}(s,a)\right] \end{aligned}$$

如果将价值函数作为一个baseline，可以定义优势函数如下：

$$A^{\pi,\gamma}(s,a)=Q^{\pi,\gamma}(s,a)-V^{\pi,\gamma}(s)$$

这样使用Advantage funtion的策略梯度就为：

$$\nabla_\theta J(\theta)=\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A^{\pi,\gamma}(s,a)\right]$$

使用N-step 近似

我们之前使用的是MC的回报G_t,但也可以使用TD的方法进行更新，或者n-step方法进行更新：

比如：

$$\begin{aligned} n=1(TD) \quad & G_t^{(1)}=r_{t+1}+\gamma v\left(s_{t+1}\right) \\ n=2 \quad & G_t^{(2)}=r_{t+1}+\gamma r_{t+2}+\gamma^2 v\left(s_{t+2}\right) \\ n=\infty(MC) \quad & G_t^{(\infty)}=r_{t+1}+\gamma r_{t+2}+\ldots+\gamma^{T-t-1} r_T \end{aligned}$$

使用了n-step方法的优势函数可以为：

$$\begin{aligned} \hat{A}_t^{(1)} & =r_{t+1}+\gamma v\left(s_{t+1}\right)-v\left(s_t\right) \\ \hat{A}_t^{(2)} & =r_{t+1}+\gamma r_{t+2}+\gamma^2 v\left(s_{t+2}\right)-v\left(s_t\right) \\ \hat{A}_t^{(\infty)} & =r_{t+1}+\gamma r_{t+2}+\ldots+\gamma^{T-t-1} r_T-v\left(s_t\right) \end{aligned}$$

这里$\hat{A}^{(1)}$具有低variance，但是高的bias，相反$\hat{A}_t^{(\infty)}$具有高variance，但是低的bias。