强化学习基础篇（三十）策略梯度(二)MC策略梯度算法

1、Score Function

假设策略π_θ是可微分的，并且在任何时候都不为0，我们可以使用下面的小技巧去转换为从$\nabla_\theta \pi_\theta(s,a)$到$\nabla_\theta \log \pi_\theta(s,a)$的求解。

$$\begin{aligned}\nabla_\theta \pi_\theta(s,a) &= \pi_\theta(s,a)\frac{\nabla_\theta \pi_\theta(s,a)}{\pi_\theta(s,a)}\\ &= \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\end{aligned}$$

其中$\nabla_\theta \log \pi_\theta(s,a)$也叫作score function。

2、Score Function方式的优势

这种转换为对score function的优势可以通过下面两种策略的形式体现出来：

一是、当策略π_θ是softmax函数，其形式是，$\pi_\theta(s,a) \propto e^{\phi(s,a)^\top \theta}$，则score function则可以有着非常简洁的如下形式：

$$\nabla_\theta \log \pi_\theta(s,a) = \phi(s,a) - \mathbb{E}_{\pi_\theta}[\phi(s,\cdot)]$$

二是、当策略π_θ是高斯函数，$a \sim \mathcal{N}\left(\mu(s), \sigma^2\right)$，其中均值$\mu(s)=\phi(s)^\top\theta$，方差假设为固定值$\sigma^2$(也可以对$\sigma$进行参数化)，其score function为：

$$\nabla_\theta \log \pi_\theta(s,a) = \frac{(a-\mu(s))\phi(s)}{\sigma^2}$$

这两个例子可以看到Score Function在某些时候是比原来的形式更加容易进行计算。

3、一步MDP的策略梯度

如果我们考虑一个简单的MDP场景，只需要一个动作步就会结束，初始状态会从一个分布中产生$s \sim d(s)$，从初始状态开始只需要经过一个时间步就结束，并得到奖励$r=R_{s,a}$。这个场景里直接使用最大似然方法计算策略梯度。

$$\begin{aligned}J(\theta) &= \mathbb{E}_{\pi_\theta}[r]\\ &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\mathcal{R}_{s,a}\end{aligned}$$

然后对目标函数进行梯度计算，计算过程中使用了score function的技巧。

$$\begin{aligned}\nabla_\theta J(\theta) &= \sum_{s\in\mathcal{S}} d(s) \sum_{a\in\mathcal{A}} \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\mathcal{R}_{s,a}\\ &= \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) r]\end{aligned}$$

4、多步MDP的策略梯度

将前面简单的one-step的MDP，扩展到多步的MDP的情况：

- (a) 定义在一个episode中，我们采集的轨迹记为τ：

$$\tau = (s_0, a_0, r_1, \ldots s_{T-1}, a_{T-1}, r_T, s_T) \sim (\pi_\theta, P(s_{t+1} \mid s_t, a_t))$$

- (b) 定义轨迹τ的奖励之和为：$R(\tau)=\sum_{t=0}^{T-1} R\left(s_t, a_t\right)$
- (c) 策略的目标函数定义为：

  $$J(\theta)=\mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T-1} R\left(s_t, a_t\right)\right]=\sum_\tau P(\tau ; \theta) R(\tau)$$

  这里的$P(\tau ; \theta)=\mu\left(s_0\right) \prod_{t=0}^{T-1} \pi_\theta\left(a_t \mid s_t\right) p\left(s_{t+1} \mid s_t, a_t\right)$表示在执行策略$\pi_\theta$的的时候轨迹$\tau$的概率。
- (d) 然后我们目标就是找到最优的参数θ可以极大化目标函数$J(\theta)$。

  $$\theta^*=\arg \max _\theta J(\theta)=\arg \max _\theta \sum_\tau P(\tau ; \theta) R(\tau)$$

- (d) 其SGD的优化方法是要计算目标函数的梯度$\nabla_\theta J(\theta)$。

  $$\begin{aligned} \nabla_\theta J(\theta) & =\nabla_\theta \sum_\tau P(\tau ; \theta) R(\tau) \\ & =\sum_\tau \nabla_\theta P(\tau ; \theta) R(\tau) \\ & =\sum_\tau \frac{P(\tau ; \theta)}{P(\tau ; \theta)} \nabla_\theta P(\tau ; \theta) R(\tau) \\ & =\sum_\tau P(\tau ; \theta) R(\tau) \frac{\nabla_\theta P(\tau ; \theta)}{P(\tau ; \theta)} \\ & =\sum_\tau P(\tau ; \theta) R(\tau) \nabla_\theta \log P(\tau ; \theta)\end{aligned}$$

  在这一步中对$P(\tau ; \theta)$的估计可以使用m次采样的方法进行，即：

  $$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{i=1}^m R\left(\tau_i\right) \nabla_\theta \log P\left(\tau_i ; \theta\right)$$

  这里继续对$\nabla_\theta \log P(\tau ; \theta)$进行分解，分解过程中将消去所有与$\theta$无关的部分：

  $$\begin{aligned} \nabla_\theta \log P(\tau ; \theta) & =\nabla_\theta \log \left[\mu\left(s_0\right) \prod_{t=0}^{T-1} \pi_\theta\left(a_t \mid s_t\right) p\left(s_{t+1} \mid s_t, a_t\right)\right] \\ & =\nabla_\theta\left[\log \mu\left(s_0\right)+\sum_{t=0}^{T-1} \log \pi_\theta\left(a_t \mid s_t\right)+\log p\left(s_{t+1} \mid s_t, a_t\right)\right] \\ & =\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\end{aligned}$$

  这里的结果很感人，将轨迹的概率梯度转换为了求策略的梯度：

  $$\nabla_\theta \log P(\tau ; \theta)=\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)$$

  最终目标函数的梯度$\nabla_\theta J(\theta)$可以转换为：

  $$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{i=1}^m R\left(\tau_i\right) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta\left(a_t^i \mid s_t^i\right)$$

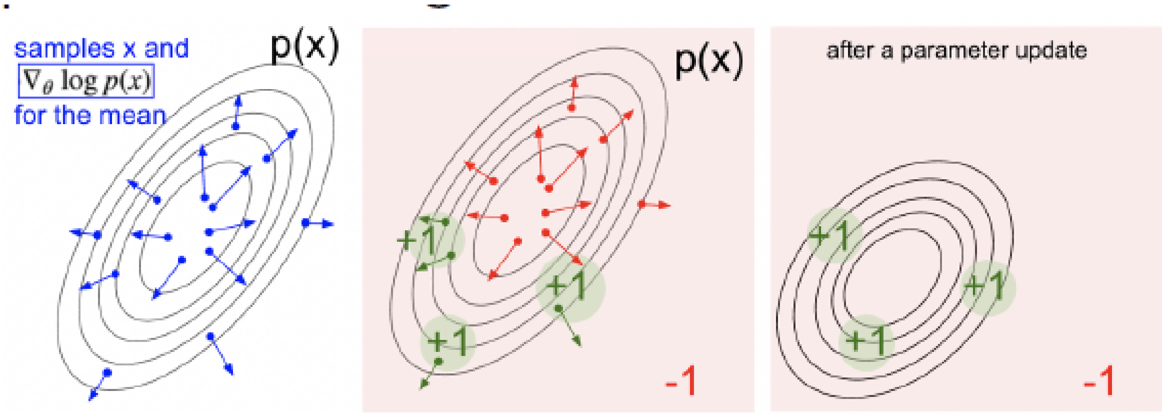
  最终我们优化目标函数的时候就完全不需要了解模型的动态参数。

5、理解Score Function Gradient估计

我们的策略函数优化的目标函数是$E_{\tau \sim \pi_\theta}[R(\tau)]$，其中$\theta$是参数，轨迹$\tau$是由采样产生。如果我们将其重新为更加广义的形式，通过函数$f(x)$替换奖励$R(\tau)$：

$$\begin{aligned} \nabla_\theta \mathbb{E}_{p(x ; \theta)}[f(x)] & =\mathbb{E}_{p(x ; \theta)}\left[f(x) \nabla_\theta \log p(x ; \theta)\right] \\ & \approx \frac{1}{S} \sum_{s=1}^S f\left(x_s\right) \nabla_\theta \log p\left(x_s ; \theta\right), \text { where } x_s \sim p(x ; \theta)\end{aligned}$$

其中函数$f(x)$的x由参数化的分布$p(x;\theta)$采样产生，需要优化参数θ使得$f(x)$的期望尽可能大。其中与策略函数优化的方式一样使用了score function的技巧，然后对$p(x;\theta)$进行采样。这个过程可以通过下面这个图理解下：



图里面每个样本是从$p(x)$采样产生，我希望采样产生出来的x能够使得$f(x)$尽量得大，$\nabla_\theta \log p(x)$的优化过程即在优化函数的形状，第一张图的每个蓝色箭头表示 $\nabla_\theta \log p(x)$，中间的各种圈是函数$f(x)$，将$p(x)$分布采样出的值给与一定的权重就得到中间图，绿色的点是权重为正的值，红色为权重为负的值。

然后我们需要使得$p(x)$的分布尽量往绿色的区域移动拟合，其拟合后的$f(x)$如最右侧的图，将使得$f(x)$能够在后续采样出的值尽可能有更大的可能性。

Policy Gradient估计 vs 最大似然估计

如果我们将Policy Gradient估计与最大似然估计进行对比：

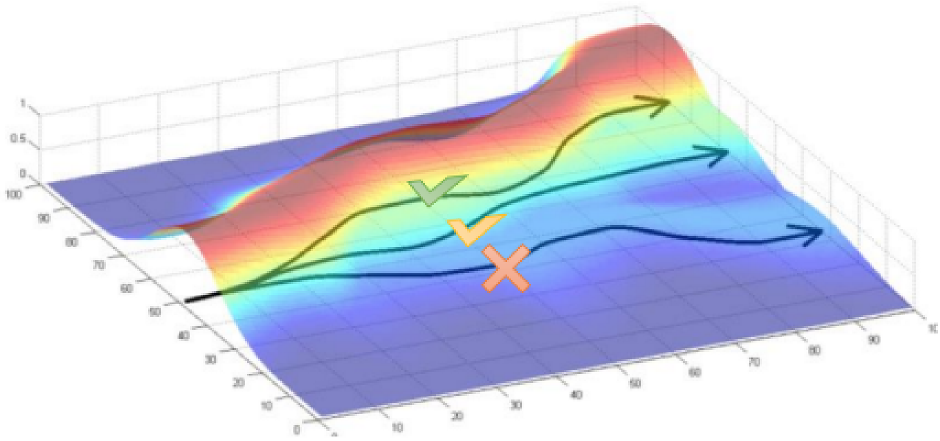
Policy Gradient估计的形式为：

$$\nabla_\theta J(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_{t,m} \mid s_{t,m}\right) \right) \left(\sum_{t=1}^{T} r\left(s_{t,m}, a_{t,m}\right) \right)$$

最大似然估计形式为：

$$\nabla_\theta J_{ML}(\theta) \approx \frac{1}{M} \sum_{m=1}^{M} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_{t,m} \mid s_{t,m}\right) \right)$$

这两者是比较类似的，除了Policy Gradient估计的形式是包含了reward函数，reward函数对likelihood的结果进行加权。即Policy Gradient估计可以看做加权后的极大似然估计。这里目标是在策略优化过程中，鼓励策略能够进入到能够产出尽可能多奖励的区域中。比如下图中，优化过程中希望轨迹的分布向着红色更高奖励的区域移动。



6、策略梯度的问题

我们策略梯度更新为：

$$\nabla_\theta J(\theta) \approx \frac{1}{m} \sum_{i=1}^{m} R\left(\tau_i\right) \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta\left(a_t^i \mid s_t^i\right)$$

其中存在的问题在于其轨迹为相当于MC方法产生出，虽然是unbiased，但是其方差（variance）很大。比如在采样不同的轨迹τ可能差异特别大，有些时候无奖励，有些时候奖励特别大。

降低方差将是更加高级的强化学习算法的核心，降低方差即可保障学习过程更加稳定。

一般有两种改进办法，一是引入时序的因果关系，二是使用baseline。

7、利用时序的因果关系(Causality)降低方差

我们已经推导得出的Policy gradient形式为：

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

其由两部分的sum构成，一个是对reward的累加，另一个是对score function的累加。

我们可以引入时序的因果关系(Causality)使得第一个加和的reward更小，我们在t'时间之前做的似然估计并不会对后续得到奖励造成影响，

对于一个单个奖励的梯度形式是：

$$\nabla_\theta \mathbb{E}_\tau\left[r_{t'}\right] = \mathbb{E}_\tau\left[r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]$$

若将所有时间步进行累加，经过推导可以得到简化后的Policy gradient形式：

$$\begin{aligned}
\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim \pi_\theta}[R] &= \mathbb{E}_\tau\left[\sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right] \\
&= \mathbb{E}_\tau\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right) \sum_{t'=t}^{T-1} r_{t'}\right] \\
&= \mathbb{E}_\tau\left[\sum_{t=0}^{T-1} G_t \cdot \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\right]
\end{aligned}$$

其中$G_t = \sum_{t'=t}^{T-1} r_{t'}$。这里是通过因果性去掉没有必要的奖励部分，在时间$t'$的策略不会影响$t < t'$部分的奖励。最后得到的采样形式是：

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

通过这种方式可以得到经典的REINFORCE算法：

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$
Repeat forever:
  Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  For each step of the episode $t = 0, \ldots, T-1$:
    $G \leftarrow$ return from step t
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi(A_t|S_t, \boldsymbol{\theta})$