策略梯度算法

本文主要介绍策略梯度（Policy Gradient）算法。

核心思想

强化学习的目标是 使得 代理学会一种策略，根据状态采取行为令未来获得的期望奖励 更高。而代理未来获得的期望奖励为：

$$\mathbb{E}_\tau[R(\tau)]$$

其中，τ 为轨迹。代理执行一个行为后，我们将后面所有的可能的轨迹完全记录下来，并根据每条轨迹的 奖励 计算 奖励信号 $R(\tau)$ 的期望。很明显，这就是代理执行该行为后所能获得的期望奖励，同一个状态下，不同行为可能导致不同的 期望奖励，期望奖励越大，意味着这个行为越‘**适用于**’当前状态。

而我们又可以将 $\mathbb{E}_\tau[R(\tau)]$ 写作 $\sum_\tau[p(\tau|\theta)*R(\tau)]$ ，其中 $p(\tau|\theta)$ 表示 在神经网络参数为 θ 的情况下，轨迹 τ 出现的概率。

我们可以发现，我们希望最大化的值 $\mathbb{E}_\tau[R(\tau)]$ 是神经网络参数 θ 的函数！ 这意味着我们可以利用梯度上升的方法，逐渐更新 参数 θ 使得奖励信号大的轨迹 τ 更容易出现，也就完成了强化学习的目标。

问题在于，这个值 $\mathbb{E}_\tau[R(\tau)]$ 虽然是 θ 的函数，但是他的梯度究竟是什么？

梯度求解

由于$\mathbb{E}_\tau[R(\tau)]=\sum_\tau[p(\tau|\theta)*R(\tau)]$，求和算子和求导算子可以交换顺序，所以原式关于 θ 的导数为 $\sum_\tau[\nabla_\theta p(\tau|\theta)*R(\tau)]$ 。
我们只需要找到所有的轨迹 τ ，并求出 $\nabla_\theta p(\tau|\theta)*R(\tau)$ ，再求和即可获得所需的梯度。

很多时候轨迹 τ 的数量很大，甚至是无穷的，此时上述方法显然不再适用。

也许我们可以想到大数定律，采样 足够多 的轨迹 τ ，对梯度做一个估计。然而**请注意**，大数定律描述的是**采样数目足够多时，样本均值趋于总体期望。**这里，我们要求的值为 $\sum_\tau[\nabla_\theta p(\tau|\theta)*R(\tau)]$，类似于一个积分，我们采样足够多的轨迹 τ 相当于在足够长的区间内求积分值，除非我们可以证明剩余区间的积分值约等于 0 。不然，我们仅仅采样足够多的 τ 也许并不能很好地估计梯度。

所以我们尝试继续对上面的式子进行数学变换。

$\nabla_\theta p(\tau|\theta)*R(\tau)=p(\tau|\theta)\nabla_\theta log(p(\tau|\theta))*R(\tau)$（这就是**策略梯度定理**）
故梯度为
$\sum_\tau p(\tau|\theta)\nabla_\theta log(p(\tau|\theta))*R(\tau)=\mathbb{E}_\tau[\nabla_\theta log(p(\tau|\theta))*R(\tau)]$

这里我们运用的思想是——将求和转换为期望。很显然，这样处理后，梯度可以运用大数定律估计求得。

算法的缺陷

a) 每一次求梯度都需要大量采样，可能导致算法效率不足。
b) 每次采样的结果都只能运用一次，导致样本利用率不足。
c) 算法十分不稳定，对于动作空间比较大的情况效果较差。

改进算法——近端策略优化

近端策略优化（Proximal Policy Optimization）着重改善样本利用率不足的问题。限于篇幅，本文中不再介绍，我将在后面的文章中对比 **DQN** 和 **PPO**。