强化学习

Reinforcement learning

第7节 策略梯度

policy gradient

张世周

Outlines

- 1.1 介绍
- 1.2 有限差分策略梯度
- 1.3 蒙特卡洛策略梯度
- 1.4 “Actor-Critic”策略梯度

基于策略的强化学习

在上一节课中我们使用参数θ近似了价值函数和动作价值函数：

$$V_\theta(s) \approx V^\pi(s)$$
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

在这节课之前策略的改善都是直接利用价值函数，比如ε-greedy策略

在本节课中，我们将直接参数化策略

$$\pi_\theta(s,a) = \mathbb{P}[a \mid s,\theta]$$

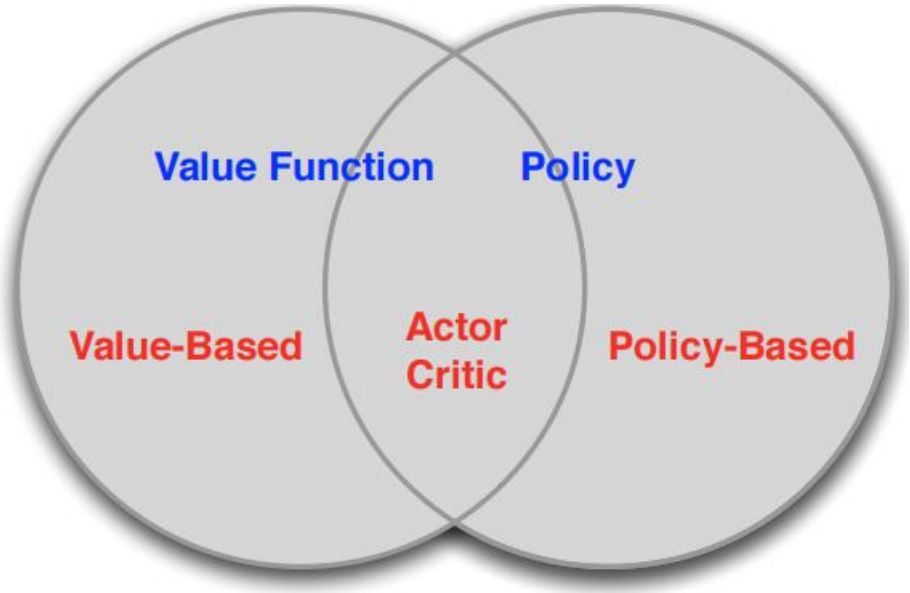
我们依旧聚焦无模型的强化学习

基于价值和基于策略的强化学习的区别

Value-Based的强化学习是通过学习价值函数指导策略制定

Policy-Based的强化学习是直接学习策略没有价值函数

还有一种Actor-Critic的强化学习方法是既学习价值函数也学习策略



基于策略的强化学习的优劣势

优势：

1、具有更好的收敛性

2、在对于那些拥有高维度或连续状态空间来说更有效

3、能够学习到一些随机策略

劣势：

1、通常收敛到局部最优而不是全局最优（通常已经够好了）

2、评估策略通常效率低下且方差很大

例子：剪刀石头布

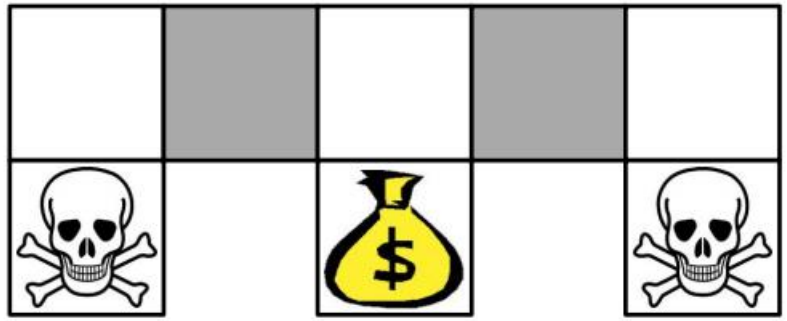
对于剪刀石头布的游戏，只要一方有一个确定性的策略，就会被对手抓住进而整体上输掉。这个时候最好的策略就是随机选择每次出法，以得到最大可能的总体奖励。

例子：迷宫寻宝（1）

智能体无法区分灰色格子：

考虑以下形式的特征（对于所有N,E,S,W）

$$\phi(s,a) = \mathbf{1}(\text{wall to N}, a = \text{move E})$$



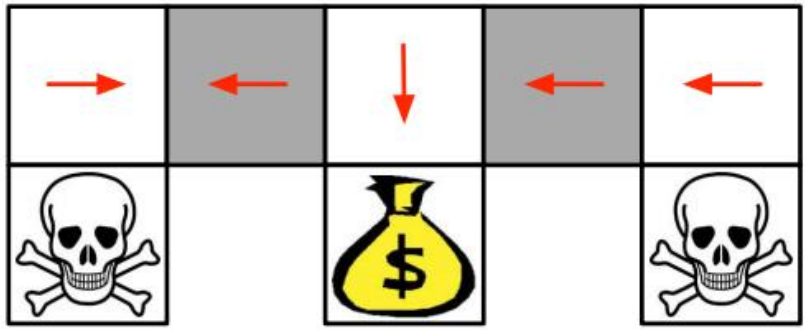
对于基于价值的RL，使用近似值函数比较

$$Q_\theta(s,a) = f(\phi(s,a),\theta)$$

到基于策略的RL，使用参数化策略

$$\pi_\theta(s,a) = g(\phi(s,a),\theta)$$

例子：迷宫寻宝（2）



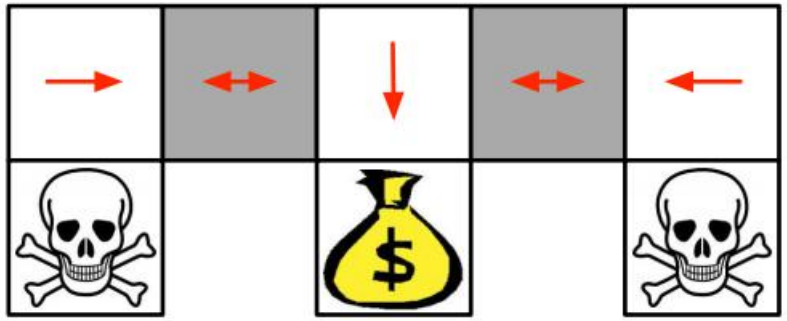
最优确定性策略将

- 在两种灰色状态下向W移动（用红色箭头显示）
- 或在两种灰色状态下向E移动

无论哪种方式，它都可能陷入困境，永远无法拿到钱

基于价值的RL学习一个近似确定性的策略，比如ε-greedy策略或greedy策略。因此，它将在很长一段时间内在走廊中徘徊

例子：迷宫寻宝（3）



最优随机策略将在灰色方格下随机向E移动或向W移动

$$\pi_\theta(\text{wall to N and S, move E}) = 0.5$$
$$\pi_\theta(\text{wall to N and S, move W}) = 0.5$$

它将以高概率在几个步骤中达到目标状态

基于策略的RL可以学习最优随机策略

策略目标函数

目标：给出带参数θ的策略 $\pi_\theta(s,a)$ ，寻找最优参数θ。

但是我们怎么才能确定策略的质量好坏呢？

- 在能够产生完整Episode的环境下我们可以使用start value：

$$J_1(\theta) = V^{\pi_\theta}(s_1) = \mathbb{E}_{\pi_\theta}[v_1]$$

- 在连续环境中我们可以使用average value：

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s)$$

- 或者是使用average reward per time-step。其中 $d^{\pi_\theta}(s)$ 是在当前策略下马尔科夫链的关于状态的一个静态分布。

$$J_{avR}(\theta) = \sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(s,a) \mathcal{R}_s^a$$

策略优化

基于策略的强化学习实际上是一个优化问题，找到参数θ来最大化目标函数 $J(\theta)$

一些方法不使用梯度，比如：爬山算法，单纯形，遗传算法

使用梯度通常可以提高效率，比如：梯度下降，共轭梯度和拟牛顿算法。

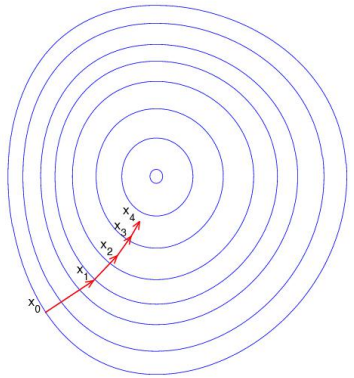
本讲内容将主要聚焦于使用梯度的策略优化。

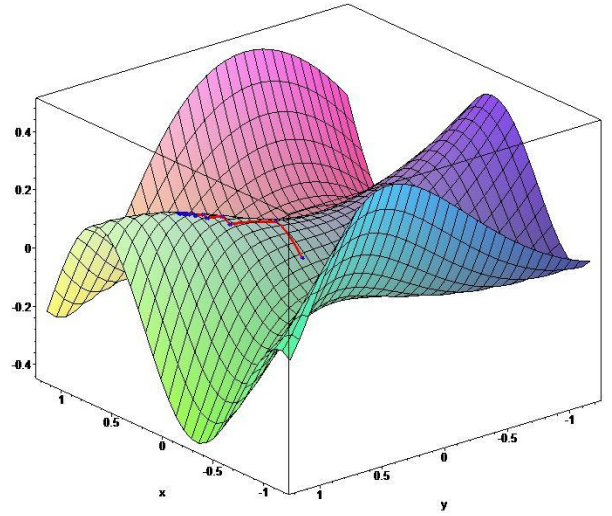
策略梯度

使 $J(\theta)$ 表示所有的策略目标函数。策略梯度算法是在 $J(\theta)$ 中通过提升策略的梯度来寻找局部最大值 $\Delta\theta = \alpha \nabla_\theta J(\theta)$

其中 $\nabla_\theta J(\theta)$ 就是策略梯度

$$\nabla_\theta J(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_n} \end{pmatrix}$$





α是一个步长参数

通过有限差分计算梯度

这是非常常用的数值计算方法，特别是当梯度函数本身很难得到的时候。具体做法是，针对参数θ的每一个分量θ_k，使用如下的公式粗略计算梯度：

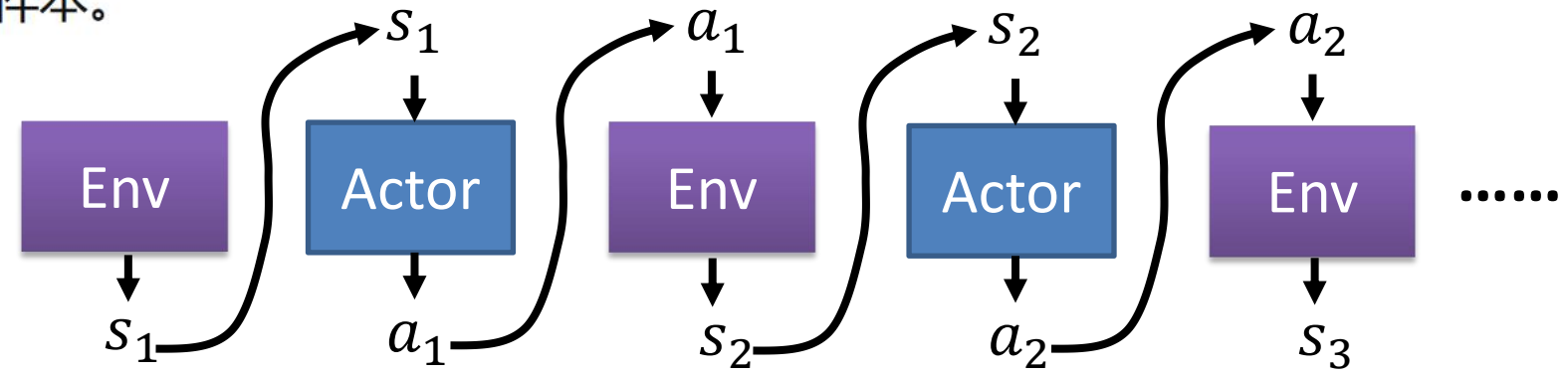
$$\frac{\partial J(\theta)}{\partial \theta_k} \approx \frac{J(\theta + \epsilon u_k) - J(\theta)}{\epsilon}$$

u_k 是一个单位向量，仅在第k个维度上值为1，其余维度为0。

这种方法使用n个求值计算n维中的策略梯度，简单、低效且有噪声——但有时有效适用于任意策略，即使策略不可微

策略梯度的推导

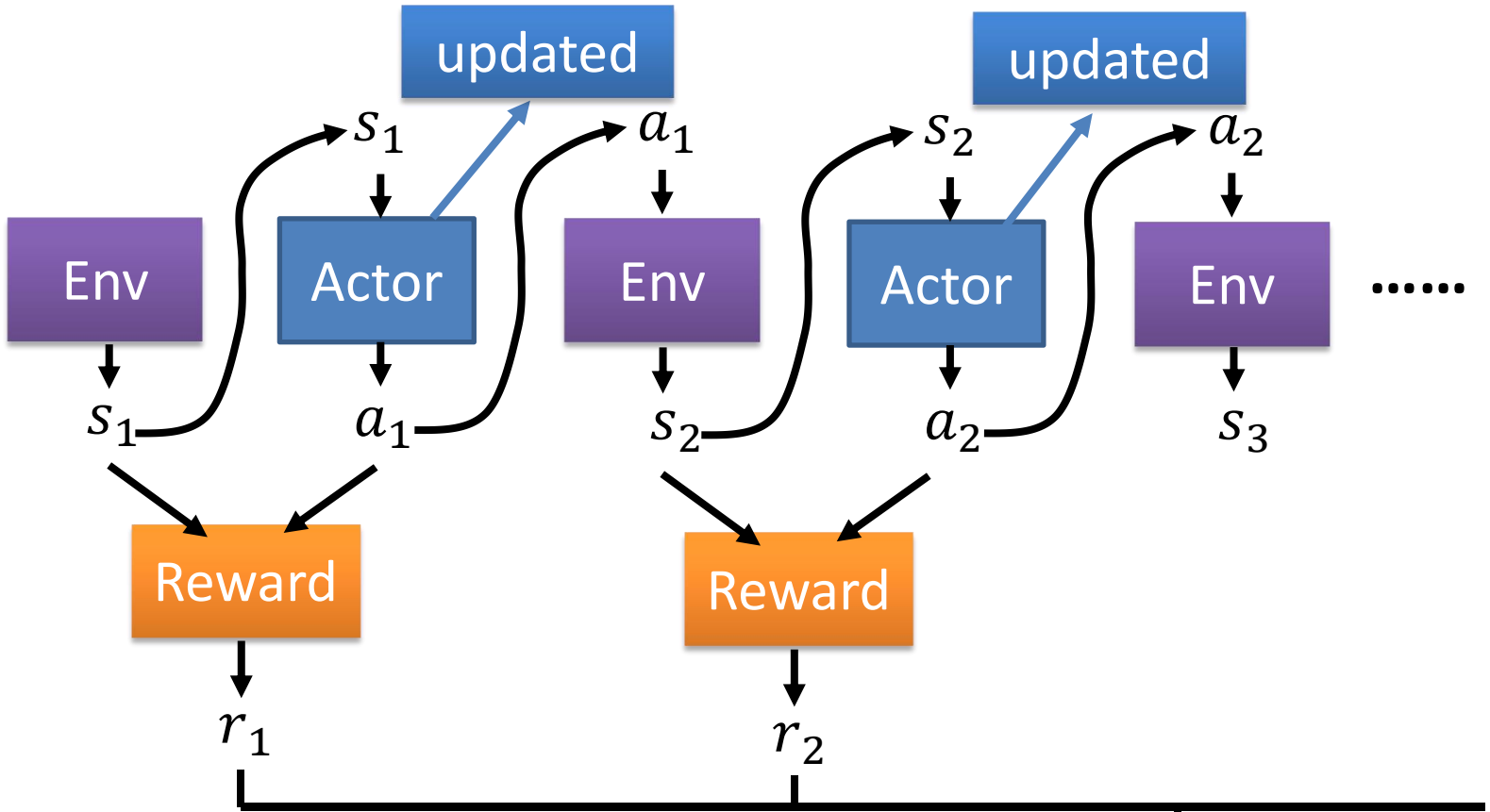
用 τ 表示每次仿真的状态-行为序列 $s_1, a_1, s_2, a_2, \cdots, s_\tau, a_\tau$ ，每一个轨迹代表了强化学习的一个样本。



Trajectory $\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$

轨迹的回报： $R(\tau) = \sum_{t=1}^{T} \gamma^t R(s_t, a_t)$ 。

策略梯度的推导



Expected Reward

$$\overline{R}_\theta = \sum R(\tau)p_\theta(\tau) = E_{\tau\sim p_\theta(\tau)}[R(\tau)] \qquad R(\tau) = \sum_T r_t$$

用 $p_\theta(\tau)$ 表示（在参数 θ 下）轨迹 τ 出现（完美复现 $s_1, a_1, s_2, a_2, \cdots, s_\tau, a_\tau$ ）的概率。

策略梯度的推导

目的：调整参数 θ 使得 $R(\tau)$ 越大越好，所以强化学习的目标函数可表示为：

$$U(\theta)=\sum_{\tau}R(\tau)p_{\theta}(\tau)$$

> **（注：轨迹回报的期望值=该轨迹出现的概率 × 该轨迹的回报值——>加权求和）**

强化学习的目标是：

$$\max_{\theta}U(\theta)=\max\sum_{\tau}R(\tau)p_{\theta}(\tau)$$

> 不同的策略 π_{θ} 影响了不同轨迹出现的概率，换句话说， $p_{\theta}(\tau)$ 其实是一个分布，我们感兴趣的是移动这个分布（通过改变参数 θ ）来提高回报值； 在一个固定的环境中，轨迹的 $R(\tau)$ 是稳定的。

策略梯度的推导

从似然率角度：

$$\nabla_\theta U(\theta) = \nabla_\theta \sum_\tau p_\theta(\tau) R(\tau)$$

求和符号内的式子收敛时，求和符号与梯度符号可以交换

$$= \sum_\tau \nabla_\theta p_\theta(\tau) R(\tau)$$

$$= \sum_\tau \frac{p_\theta(\tau)}{p_\theta(\tau)} \nabla_\theta p_\theta(\tau) R(\tau)$$

$$= \sum_\tau p_\theta(\tau) \frac{\nabla_\theta p_\theta(\tau)}{p_\theta(\tau)} R(\tau)$$

使用了 $\nabla_\theta \log(z) = \frac{1}{z} \nabla_\theta z$

$$= \sum_\tau p_\theta(\tau) \nabla_\theta \log p_\theta(\tau) R(\tau)$$

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\nabla_\theta \log p_\theta(\tau) R(\tau) \right]$$

利用当前策略 π_θ 采样 m 条轨迹，使用经验平均来估计梯度：

$$\nabla_\theta U(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log p_\theta(\tau) R(\tau)$$

策略梯度的推导

从重要性采样角度：

对于参数的更新 $\theta_{old} \rightarrow \theta$ ，我们使用参数 θ_{old} 产生的数据去评估参数 θ 的回报期望，由重要性采样得到：

$$
\begin{aligned}
U(\theta) &= \sum_{\tau} p_{\theta}(\tau) R(\tau) \\
&= \sum_{\tau} p_{\theta_{old}}(\tau) \frac{p_{\theta}(\tau)}{p_{\theta_{old}}(\tau)} R(\tau) \\
&= \mathbb{E}_{\tau \sim p_{\theta_{\text{old}}}} \left[\frac{p_{\theta}(\tau)}{p_{\theta_{old}}(\tau)} R(\tau) \right]
\end{aligned}
$$

此时导数变成了 $\nabla_{\theta} U(\theta) = \mathbb{E}_{\tau \sim p_{\theta_{\text{old}}}} \left[\frac{\nabla_{\theta} p_{\theta}(\tau)}{p_{\theta_{old}}(\tau)} R(\tau) \right]$

（注： θ_{old} 是一个过去的值，或者说是已知的值，所以它不带任何参数。

当 $\theta = \theta_{old}$ 时，我们得到当前策略的导数：

$$
\begin{aligned}
\nabla_{\theta} U(\theta)\Big|_{\theta=\theta_{old}} &= \mathbb{E}_{\tau \sim p_{\theta_{\text{old}}}} \left[\frac{\nabla_{\theta} p_{\theta}(\tau)\Big|_{\theta_{old}}}{p_{\theta_{old}}(\tau)} R(\tau) \right] \\
&= \mathbb{E}_{\tau \sim p_{\theta_{\text{old}}}} \left[\nabla_{\theta} \log p_{\theta}(\tau)\Big|_{\theta_{old}} R(\tau) \right]
\end{aligned}
$$

策略梯度的推导

策略梯度的理解：

$$\nabla_\theta U(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log p_\theta(\tau) R(\tau)$$

$\nabla_\theta \log p_\theta(\tau)$ 是轨迹 τ 的概率随参数 θ 变化最陡的方向

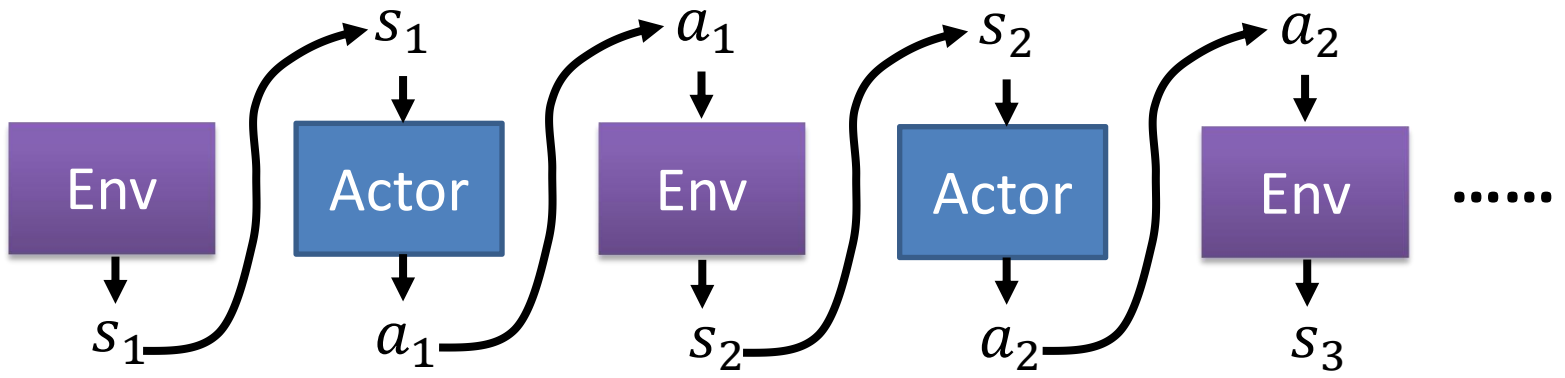
（注：$\log$ 是单调递增的函数，对 $\log p_\theta(\tau)$ 求梯度相当于对 $p_\theta(\tau)$ 求梯度）

沿正方向，轨迹出现的概率会变大

沿负方向，轨迹出现的概率会变小

$R(\tau)$ 控制了参数更新的方向和步长，正负决定了方向，大小决定了增大（减小）的幅度

策略梯度的推导



$p_\theta(\tau)$ **Trajectory** $\tau = \{s_1, a_1, s_2, a_2, \cdots, s_T, a_T\}$

$$= p(s_1)p_\theta(a_1|s_1)p(s_2|s_1,a_1)p_\theta(a_2|s_2)p(s_3|s_2,a_2)\cdots$$

$$= p(s_1)\prod_{t=1}^{T} p_\theta(a_t|s_t)p(s_{t+1}|s_t,a_t)$$

$p\left(s_{t+1}^{(i)}|s_t^{(i)},a_t^{(i)}\right)$ 由Environment决定

$p_\theta\left(a_t^{(i)}|s_t^{(i)}\right)$ 由Agent的策略决定

策略梯度的推导

$$
\begin{aligned}
\nabla_\theta \log p_\theta\left(\tau^{(i)}\right) &= \nabla_\theta \log\left[p(s_1^{(i)}) \prod_{t=1}^{T} p_\theta(a_t^{(i)}|s_t^{(i)}) p(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)})\right] \\
&= \nabla_\theta \log\left[\prod_{t=1}^{T} p_\theta\left(a_t^{(i)}|s_t^{(i)}\right) p\left(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}\right)\right] \\
&= \nabla_\theta\left[\sum_{t=1}^{T} \log p\left(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}\right) + \sum_{t=1}^{T} \log p_\theta\left(a_t^{(i)}|s_t^{(i)}\right)\right] \\
&= \nabla_\theta\left[\sum_{t=1}^{T} \log p_\theta\left(a_t^{(i)}|s_t^{(i)}\right)\right] \\
&= \sum_{t=1}^{T} \nabla_\theta \log p_\theta\left(a_t^{(i)}|s_t^{(i)}\right)
\end{aligned}
$$

策略梯度的推导

根据之前的推导，我们可以在仅有可微分的策略模型 π_θ 的情况下，求得 $\nabla_\theta U(\theta)$

$$\hat{\eta} = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log p_\theta\left(\tau^{(i)}\right) R\left(\tau^{(i)}\right)$$

这里 $\nabla_\theta \log p_\theta\left(\tau^{(i)}\right) = \sum_{t=1}^{T} \nabla_\theta \log p_\theta\left(a_t^{(i)} | s_t^{(i)}\right)$

所以 $\hat{\eta} = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{T} \nabla_\theta \log p_\theta\left(a_t^{(i)} | s_t^{(i)}\right) R\left(\tau^{(i)}\right)$

$\hat{\eta}$ 是 $\nabla_\theta U(\theta)$ 的无偏估计 $\mathbb{E}[\hat{\eta}] = \nabla_\theta U(\theta)$

策略梯度的推导

$$\nabla \bar{R}_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)\nabla log p_\theta(\tau)]$$

Policy Gradient

Given policy π_θ

τ^1: (s_1^1, a_1^1) $R(\tau^1)$

(s_2^1, a_2^1) $R(\tau^1)$

τ^2: (s_1^2, a_1^2) $R(\tau^2)$

(s_2^2, a_2^2) $R(\tau^2)$

only used once

Update Model

$$\theta \leftarrow \theta + \eta \nabla \bar{R}_\theta$$

$$\nabla \bar{R}_\theta = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} R(\tau^n) \nabla log p_\theta(a_t^n | s_t^n)$$

Data Collection

评分函数/迹向量（Score Function）

我们现在计算策略梯度：假设策略π_θ在非零时是可微的，并且我们知道梯度

$\nabla_\theta \pi_\theta(s,a)$ 似然比利用以下变形：

已知函数在某个变量θ处的梯度等于该处函数值与该函数的对数函数在此处梯度的乘积：

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

我们定义Score function为：

$$\nabla_\theta \log \pi_\theta(s,a)$$

Softmax策略

Softmax策略是针对一些具有离散的行为常用的一个策略。我们希望有平滑的参数化的策略来决策：针对每一个离散的行为，应该以什么样的概率来执行它。

为此，我们把行为看成是多个特征在一定权重下的线性代数和：$\phi(s,a)^\top\theta$

而我们采取某一具体行为的概率与e的该值次幂成正比：$\pi_\theta(s,a) \propto e^{\phi(s,a)^\top\theta}$

Score function为：

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

高斯策略

与Softmax策略不同的是，高斯策略常应用于连续行为空间，打个比方：如果控制机器人行走，要调整流经控制某个电机的电流值，而这是一个连续的取值。

使用高斯策略时，我们通常对于均值有一个参数化的表示，同样可以是一些特征的线性代数和： $\mu(s) = \phi(s)^\top \theta$

方差可以是固定值 σ^2 ，也可以用参数化表示。

行为对应于一个具体的数值，该数值从以μ(s)为均值、σ为标准差的高斯分布中随机采样产生： $a \sim \mathcal{N}(\mu(s), \sigma^2)$

对应的Score函数是： $\nabla_\theta \log \pi_\theta(s, a) = \dfrac{(a - \mu(s))\phi(s)}{\sigma^2}$

单步MDPs

先考虑如下一个非常简单的单步MDP问题：从一个分布 $d(s)$ 中采样得到一个状态s，从s开始，采取一个行为a，得到即时奖励 $r = \mathcal{R}_{s,a}$ 后终止。整个MDP只有一个状态、行为、即时奖励。在这个MDP过程中，如何最大化奖励？

利用似然比去计算策略梯度：

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

策略梯度定理

策略梯度定理将似然比方法推广到多步MDP

用长期价值 $Q^{\pi}(s,a)$ 代替瞬时奖励r

策略梯度定理适用于start value目标、average reward和average value目标

定理

对于任何可微的策略 $\pi_\theta(s,a)$ ，对于任何策略的目标函数 $J = J_1$ ，J_{avR} 或者 $\frac{1}{1-\gamma}J_{avV}$ ，策略梯度都是：

$$\nabla_\theta J(\theta) = E_{\pi\theta}[\nabla_\theta \log \pi_\theta(s,a) Q^{\pi\theta}(s,a)]$$

蒙特卡洛策略梯度 (REINFORCE)

通过随机梯度上升来更新参数，利用策略梯度定理，使用返回值v_t作为 $Q^{\pi_\theta}(s_t, a_t)$ 的无偏样本 $\Delta\theta_t = \alpha\nabla_\theta \log \pi_\theta(s_t, a_t)v_t$

算法如下：

function REINFORCE

Initialise θ arbitrarily

for each episode $\{s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

for $t = 1$ to $T - 1$ **do**

$\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(s_t, a_t)v_t$

end for

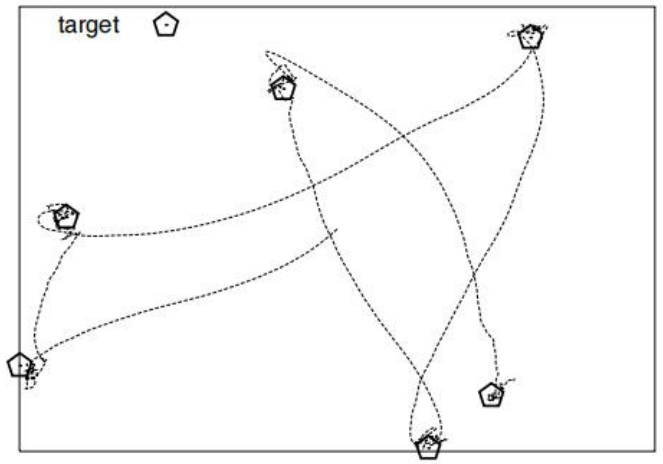
end for

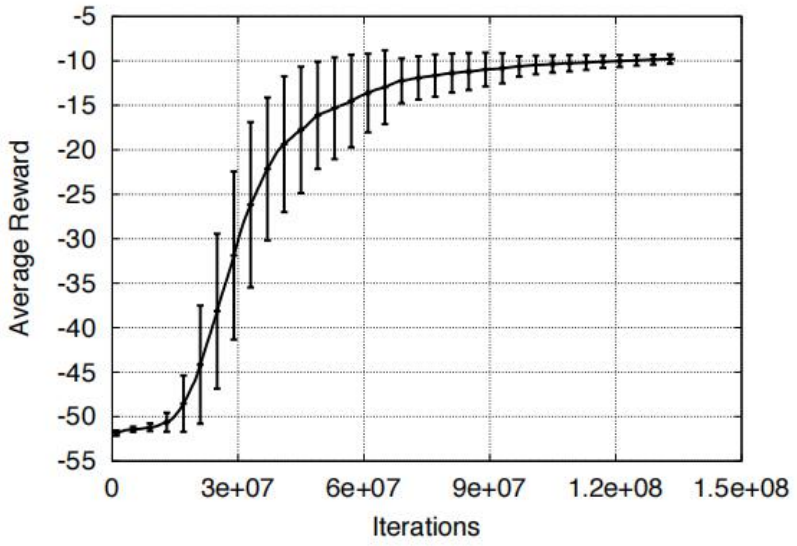
return θ

end function

每一个增量更新都正比于回报v_t和一个向量的乘积，这个向量是选取动作的概率的梯度除以这个概率本身。
这个向量是参数空间中使得将来在状态S_t下重复选择动作A_t的概率增加最大的方向。
这个更新使得参数向量沿着这个方向增加，更新大小正比于回报，反比于选择动作的概率。前者的意义在于它使得参数向着更有利于产生最大回报的动作的方向更新。后者有意义是因为如果不这样的话，频繁被选择的动作会占优（在这些方向更新更频繁），即使这些动作不是产生最大回报的动作，最后可能也会胜出，因而影响性能指标的优化。

例子：冰球世界





连续的动作对冰球施加很小的力

冰球因接近目标而得到奖励

目标位置每30秒重置一次

策略是使用蒙特卡罗策略梯度的变体进行训练的

使用评判器来降低方差

蒙特卡罗策略梯度仍然具有很高的方差，因此我们使用一个评判器来估计动作价值函数 $Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$

Actor-Critic算法维护了两组参数

评判器：更新动作价值函数参数**w**

行动器：按照评判器建议的方向更新策略参数**θ**

Actor-Critic算法遵循一个近似的策略梯度

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)\right]$$

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(s,a)\ Q_w(s,a)$$

评估动作价值函数

评判器正在解决一个熟悉的问题：策略评估。即策略 π_θ 对当前参数θ的效果如何？

这个问题在之前的两节课中都进行了探讨：

- 蒙特卡洛策略评估
- 时序差分学习
- TD(λ)

可以使用例如最小二乘策略评估。

动作价值“Actor-Critic”算法

一个简单的actor-critic算法可以使用基于行为价值的critic；使用线性价值函数来近似状态行为价值函数：

$$Q_w(s,a) = \phi(s,a)^\top w$$

其中Critic通过线性近似的TD(0)更新w，Actor通过策略梯度更新θ。具体算法流程如下：

function QAC

Initialise s, θ

Sample $a \sim \pi_\theta$

for each step **do**

Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s,\cdot}^a$

Sample action $a' \sim \pi_\theta(s', a')$

$\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$

$w \leftarrow w + \beta \delta \phi(s, a)$

$a \leftarrow a', s \leftarrow s'$

end for

end function

“Actor-Critic”算法的偏差

近似策略梯度会引入偏差；

一个有偏差的策略梯度可能无法找到正确的解

幸运的是，如果我们仔细选择值函数近似，那么我们就可以避免引入任何偏差。

也就是说我们仍然可以遵循精确的策略梯度。

兼容近似函数

那么怎样才算是一个小心设计了的 $Q_w(s,a)$ 呢？需要满足下面两个条件：

兼容近似函数定理：

1. 近似价值函数算子与策略兼容：

$$\nabla_w Q_w(s,a) = \nabla_\theta \log \pi_\theta(s,a)$$

2. 价值函数参数w使得均方差最小：

$$\varepsilon = E_{\pi\theta}[(Q^{\pi\theta}(s,a) - Q_w(s,a))^2]$$

符合这两个条件，则策略梯度是精确的，此时：

$$\nabla_\theta J(\theta) = E_{\pi\theta}[\nabla_\theta \log \pi_\theta(s,a) Q_w(s,a)]$$

兼容近似函数定理的证明

如果选择w来最小化均方误差，则ε针对w的梯度必然为零，

$$\nabla_w \varepsilon = 0$$

$$\mathbb{E}_{\pi_\theta}\left[(Q^\theta(s,a) - Q_w(s,a))\nabla_w Q_w(s,a)\right] = 0$$

$$\mathbb{E}_{\pi_\theta}\left[(Q^\theta(s,a) - Q_w(s,a))\nabla_\theta \log \pi_\theta(s,a)\right] = 0$$

$$\mathbb{E}_{\pi_\theta}\left[Q^\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)\right] = \mathbb{E}_{\pi_\theta}\left[Q_w(s,a)\nabla_\theta \log \pi_\theta(s,a)\right]$$

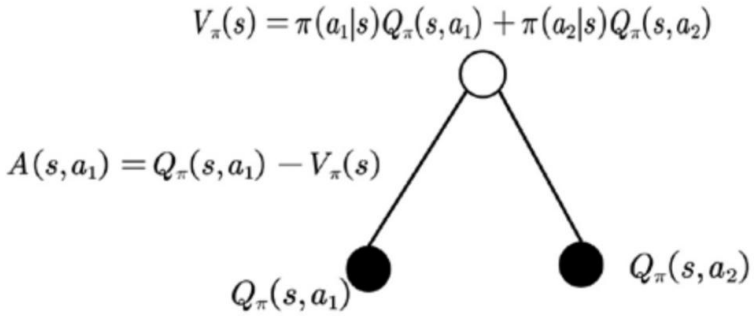
所以 $Q_w(s,a)$ 可以直接替换到策略梯度中，

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

通过使用基线的方式来减少方差

我们从策略梯度中减去一个基线函数B(s)，这可以减少方差，而不改变期望

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)B(s)\right] &= \sum_{s\in\mathcal{S}} d^{\pi_\theta}(s)\sum_a \nabla_\theta \pi_\theta(s,a)B(s) \\ &= \sum_{s\in\mathcal{S}} d^{\pi_\theta} B(s)\nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta(s,a) \\ &= 0\end{aligned}$$



一个很好的基线是状态值函数 $B(s) = V^{\pi_\theta}(s)$

所以我们可以使用优势函数 $A^{\pi_\theta}(s,a)$ 来重写策略梯度

function $A^{\pi_\theta}(s,a)$

$$A^{\pi_\theta}(s,a) = Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

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

值函数V(s)是该状态下所有动作值函数关于动作概率的平均值。动作值函数Q(s,a)是单个动作所对应的值函数。优势函数是评价当前动作值函数相对于平均值的大小。优势指的是动作值函数相比于当前状态的值函数的优势。

估计优势函数（1）

- **Advantage 函数可以明显减少策略梯度的方差，因此算法的Critic部分可以去估计advantage函数而不是仅仅估计行为价值函数。例如，通过估计 $V^{\pi_\theta}(s)$ 和 $Q^{\pi_\theta}(s,a)$ 使用两个函数近似器和两个参数向量，**

$$V_v(s) \approx V^{\pi_\theta}(s)$$
$$Q_w(s,a) \approx Q^{\pi_\theta}(s,a)$$
$$A(s,a) = Q_w(s,a) - V_v(s)$$

- **通过例如TD学习的方法来更新这两个价值函数**

估计优势函数（2）

- 根据真实的状态价值函数 $V^{\pi_\theta}(s)$ 算出 $\delta^{\pi_\theta} = r + \gamma V^{\pi_\theta}(s') - V^{\pi_\theta}(s)$
- 这样得到的TD误差是advantage函数的无偏估计，这同样是根据行为价值函数的定义推导成立的，即：

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\delta^{\pi_\theta}|s,a\right] &= \mathbb{E}_{\pi_\theta}\left[r + \gamma V^{\pi_\theta}(s')|s,a\right] - V^{\pi_\theta}(s) \\ &= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s) \\ &= A^{\pi_\theta}(s,a)\end{aligned}$$

- 如此，我们就可以使用TD误差来计算策略梯度：

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

- 实际运用时，我们使用一个近似的TD误差，即用状态函数的近似函数来代替实际的状态函数：

$$\delta_v = r + \gamma V_v(s') - V_v(s)$$

- 这样做的好处就是，我们只需要一套参数描述状态价值函数，而不再需要针对行为价值近似函数了。

不同时间尺度的Critic

Critic能够通过不同时间尺度的多种目标值来评估价值函数 $V_\theta(s)$

- MC - 直至Episode结束： $\Delta\theta = \alpha(v_t - V_\theta(s))\phi(s)$
- TD(0) - 一步： $\Delta\theta = \alpha(r + \gamma V(s') - V_\theta(s))\phi(s)$
- TD(λ)的前向视角 - 需要至Episode结束： $\Delta\theta = \alpha(v_t^\lambda - V_\theta(s))\phi(s)$
- TD(λ)的后向视角 - 实时，具备频率记忆和近时记忆功能：

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$
$$e_t = \gamma\lambda e_{t-1} + \phi(s_t)$$
$$\Delta\theta = \alpha\delta_t e_t$$

不同时间尺度的Actor

- 策略梯度同样可以在不同时间尺度评估：

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

- MC - 直至Episode结束：

$$\Delta\theta = \alpha(v_t - V_v(s_t))\nabla_\theta \log \pi_\theta(s_t,a_t)$$

- TD(0) - 一步TD error

$$\Delta\theta = \alpha(r + \gamma V_v(s_{t+1}) - V_v(s_t))\nabla_\theta \log \pi_\theta(s_t,a_t)$$

具有资格迹的策略梯度

- TD(λ)的前向视角 - 需要至Episode结束：

$$\Delta\theta = \alpha(v_t^{\lambda} - V_v(s_t))\nabla_\theta \log \pi_\theta(s_t, a_t)$$

- TD(λ)的后向视角 - 实时，具备频率记忆和及时记忆功能：

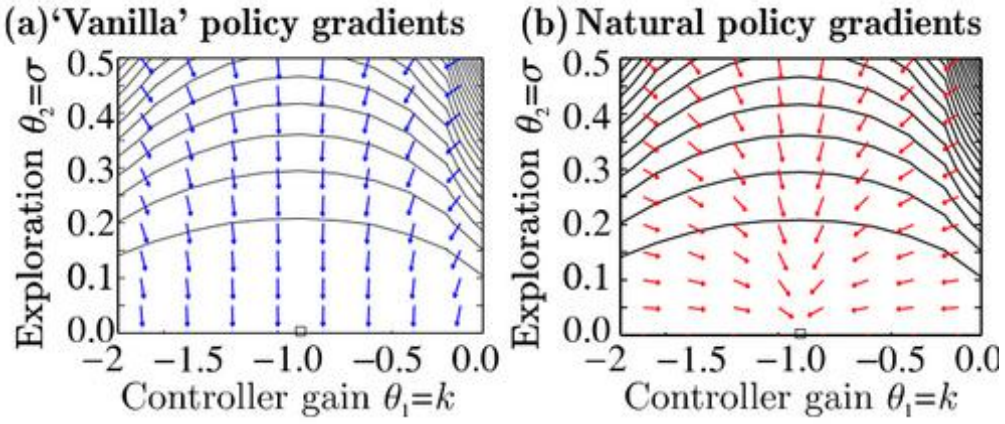
$$\begin{aligned}\delta &= r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)\\ e_{t+1} &= \lambda e_t + \nabla_\theta \log \pi_\theta(s, a)\\ \Delta\theta &= \alpha\delta e_t\end{aligned}$$

- 此更新可以在线应用于不完整的序列

可替代的策略梯度方向

- 梯度上升算法可以遵循任何上升方向。
- 良好的上升方向可以显著加快收敛速度。
- 同时，策略通常可以重参数化而不改变动作概率。
- 例如，增加softmax策略中所有操作的分数
- vanilla梯度对重参数化比较敏感

自然策略梯度



自然策略梯度是独立于重参数化的，当针对策略进行一个小的、固定量的改变时，它会找到最接近**vanilla**梯度的上升方向

$$\nabla_\theta^{nat}\pi_\theta(s,a) = G_\theta^{-1}\nabla_\theta\pi_\theta(s,a)$$

其中 G_θ 是**Fisher**信息矩阵

$$G_\theta = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\nabla_\theta \log \pi_\theta(s,a)^T\right]$$

自然策略梯度

- 在VPG中，我们统一设定步长为α。这意味着在有的θ处，我们走α的步长可能只会导致J(π)有微弱的变化，而在某些θ处，可能走同样α的步长就会导致巨大的变化。将α设的太小会导致无法收敛，而将α设的太大则有可能经常发生“模型崩溃”。这就使得我们很难调整出合适的α。
- 自然梯度法中，由于采用了FIM的逆矩阵，这使得我们对于步长的衡量更加合理，也使得我们在调整步长的时候相当于在同一尺度统一调整。每一步迭代中策略的变化都是差不多的，这可以使得算法的收敛性更好。
- 当然，这也意味着我们要付出更大的计算量（因为要计算G(s,θ)关于s的期望还要求逆矩阵）。正因为这个算法是建立在策略之间很自然的度量方法（流形上的度量）之上的，所以这个算法又被称作自然梯度法。

https://zhuanlan.zhihu.com/p/228099600

自然Actor-Critic

- 使用兼容的近似函数，即

$$\nabla_w A_w(s,a) = \nabla_\theta \log \pi_\theta(s,a)$$

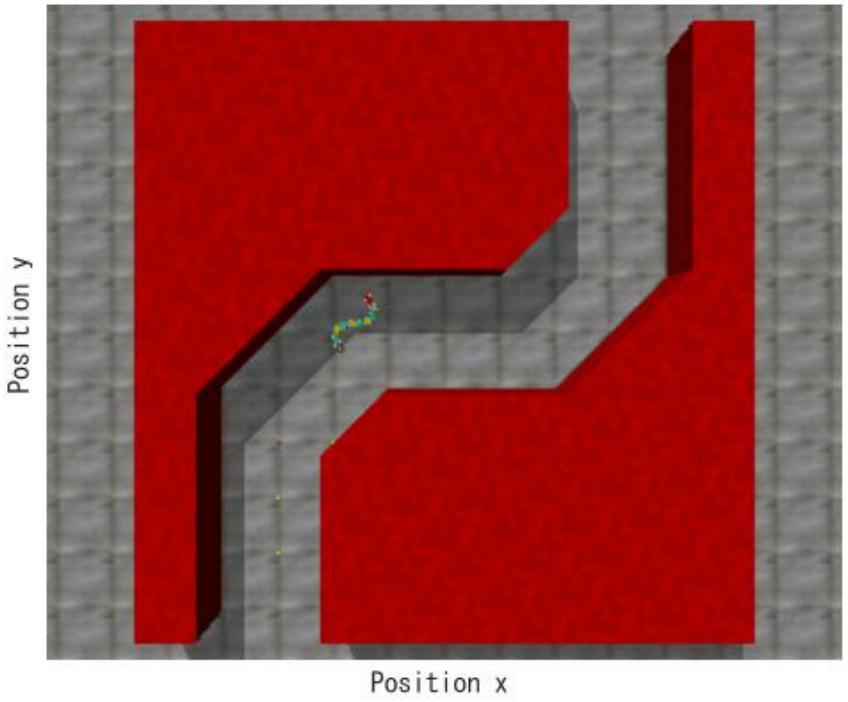
- 自然策略梯度简化为，

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) A^{\pi_\theta}(s,a)\right] \\ &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a) \nabla_\theta \log \pi_\theta(s,a)^T w\right] \\ &= G_\theta w\end{aligned}$$

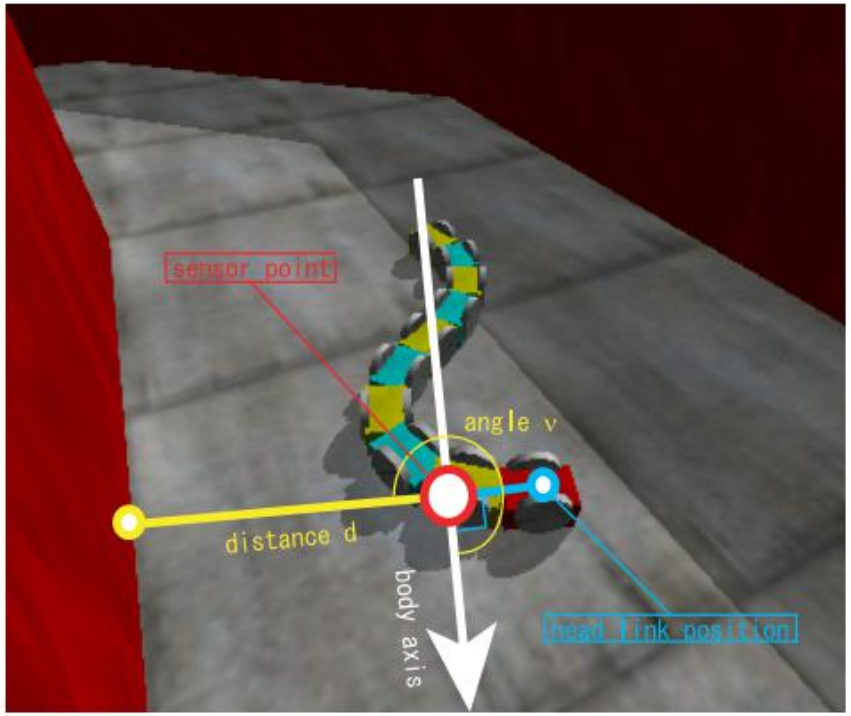
$$\nabla_\theta^{nat} J(\theta) = w$$

- 也即，在Critic参数的方向上更新Actor参数

自然Actor-Critic in Snake Domain

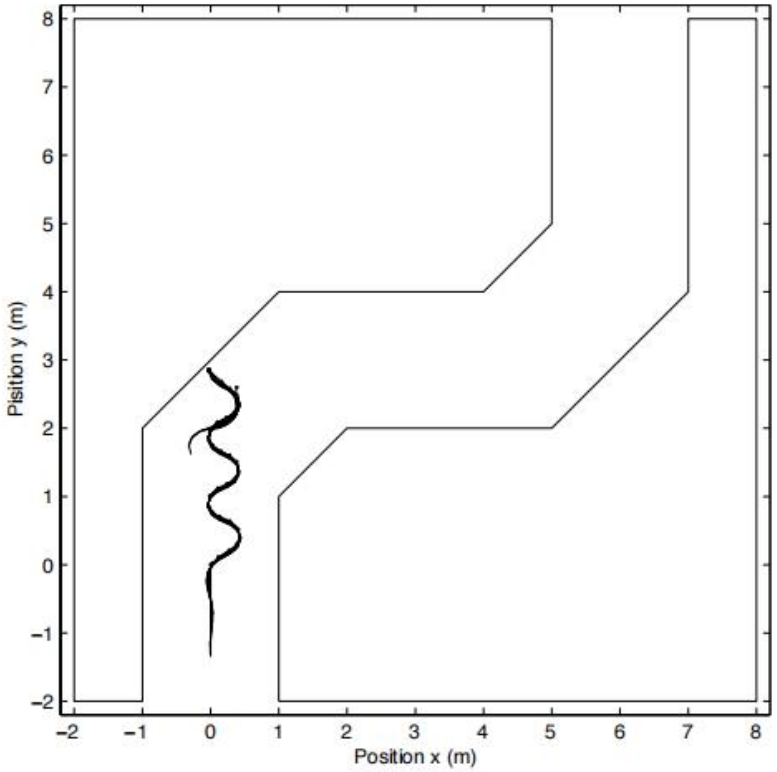


(a) Crank course

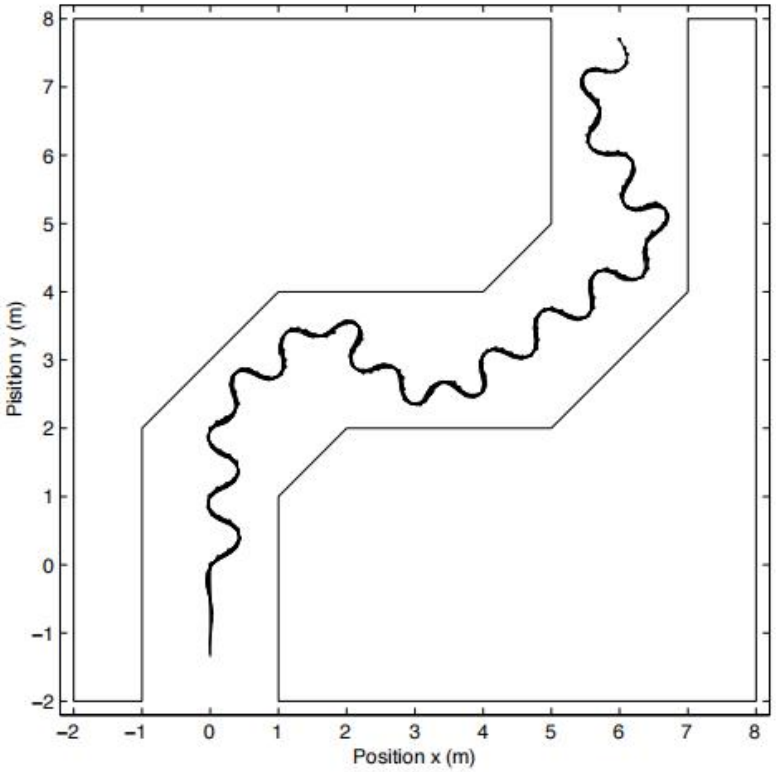


(b) Sensor setting

自然Actor-Critic in Snake Domain（2）

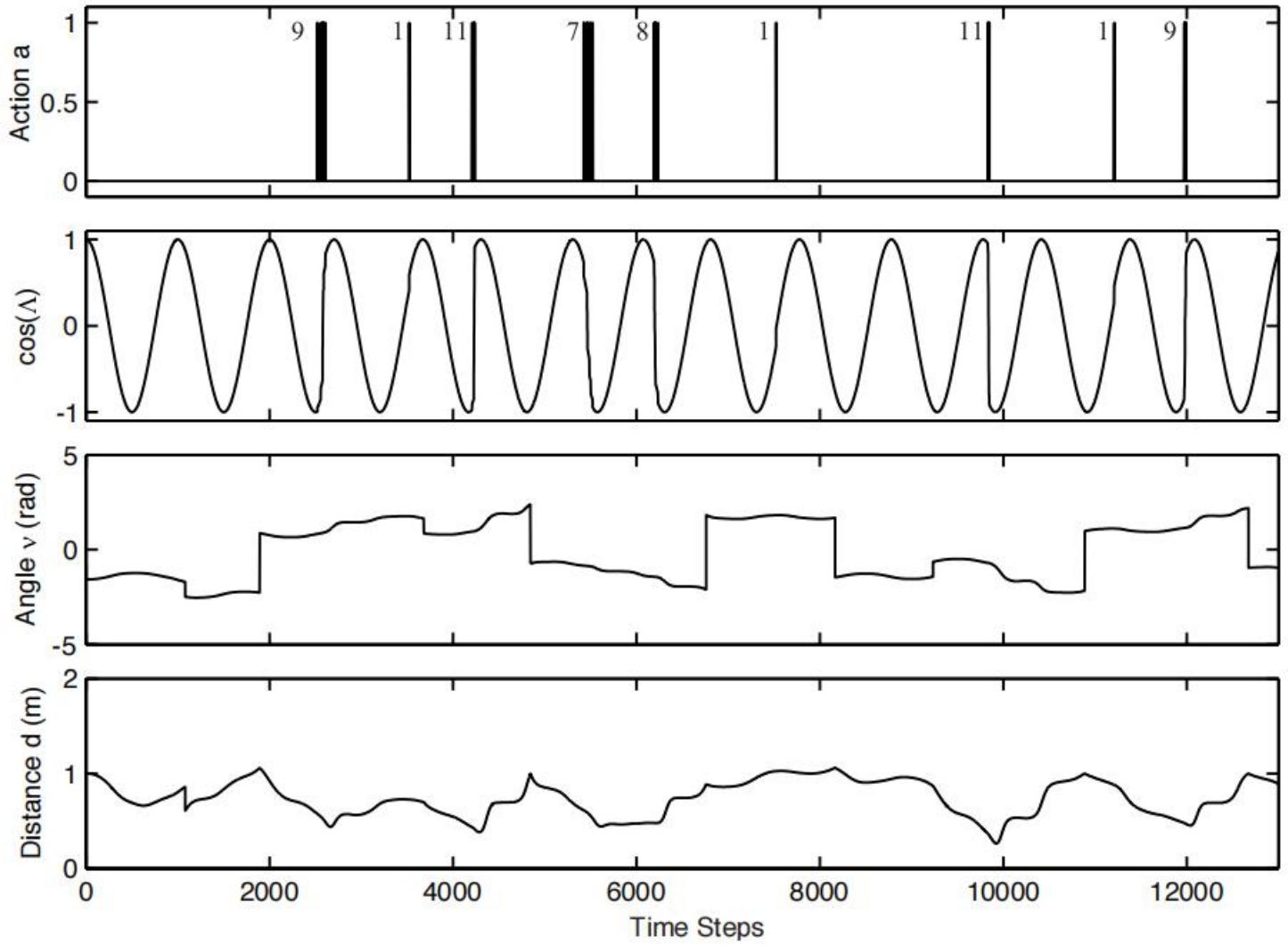


(a) Before learning



(b) After learning

自然Actor-Critic in Snake Domain（3）



策略梯度算法的总结

- 策略梯度有许多等价形式

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ v_t\right] && \text{REINFORCE}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ Q^w(s,a)\right] && \text{Q Actor-Critic}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ A^w(s,a)\right] && \text{Advantage Actor-Critic}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta\right] && \text{TD Actor-Critic}\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(s,a)\ \delta e\right] && \text{TD}(\lambda)\ \text{Actor-Critic}\\
G_\theta^{-1}\nabla_\theta J(\theta) &= w && \text{Natural Actor-Critic}
\end{aligned}
$$

- 每种形式都对应一个随机梯度上升算法
- Critic使用策略评估（例如**MC**或**TD**学习）来估计 $Q^\pi(s,a)$, $A^\pi(s,a)$ 或 $V^\pi(s)$

The End