机器学习: Part 5

- 决策论规划
- 强化学习：基于值的方法
- 强化学习：基于策略的方法

*Slides based on those of D. Poole and A. Mackworth

什么是强化学习？

- 强化学习：一个智能体在环境中学习如何行动从而最大化一个数值奖励信号。
- 学习者不被告知应该采取什么行动，而是通过尝试行动并观察奖励来发现最佳行动。

强化学习的例子

- 直升机飞行特技演示
- 在十五子棋比赛中击败世界冠军
- 管理投资组合
- 控制发电站
- 使人形机器人行走
- 在多种不同的Atari 游戏中超越人类

奖励的例子

- 直升机飞行特技演示
  - 正奖励：遵循期望轨迹
  - 负奖励：碰撞
- 在十五子棋比赛中击败世界冠军
  - 正奖励：赢得比赛
  - 负奖励：输掉比赛
- 管理投资组合
  - 正奖励：收益
- 控制发电站
  - 正奖励：生产电力
  - 负奖励：超过安全阈值

奖励的例子

- 使人形机器人行走
  - 正奖励：向前移动
  - 负奖励：跌倒
- 在多种不同的Atari 游戏中超越人类
  - 正奖励：加分
  - 负奖励：减分

强化学习的特点

强化学习与其他机器学习范式的不同之处?

- 没有监督者，只有奖励信号。
- 反馈是延迟的，不是即时的。
- 时间非常重要（顺序性、非独立同分布数据）。
- 智能体的行动影响其接收的后续数据。

决策论规划

智能体应该如何行动当

- 智能体接收到奖励（和惩罚）并试图最大化其所获得的奖励
- 行动可以是随机的；行动的结果无法完全预测。
- 存在一个模型，该模型指定了行动的（概率性）结果和奖励。
- 世界是完全可观察的（智能体通过观察知道世界的状态）

马尔可夫决策过程（MDP）

MDP 包括：

- 状态集合S。
- 动作集合A。
- $P(s'|s,a)$ 指定了在智能体处于状态s 并执行动作a 时转移到状态s' 的概率。
- $R(s,a,s')$ 是智能体处于状态s，执行动作a 并最终进入状态s' 时所获得的期望奖励。
- $0 \leq \gamma \leq 1$ 是折扣因子。

例子：锻炼与否？

每周*Sam* 都需要决定是否进行锻炼：

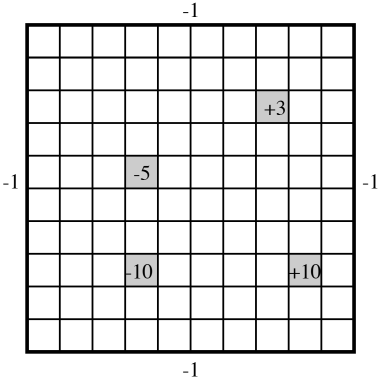
- 状态: $\{fit, unfit\}$
- 动作: $\{exercise, relax\}$
- 动态:

状态	动作	$P(fit\|G\breve{\ },)$
fit	exercise	0.99
fit	relax	0.7
unfit	exercise	0.2
unfit	relax	0.0

- 奖励（不依赖于结果状态）:

状态	动作	奖励
fit	exercise	8
fit	relax	10
unfit	exercise	0
unfit	relax	5

网格世界模型



网格世界模型

- 动作：上、下、左、右。
- 机器人的位置对应着100个状态。
- 机器人以0.7的概率朝着期望的方向前进，
- 以0.1的概率朝着其他三个方向之一前进。
- 如果机器人撞到外墙，它会保持在当前位置，并且获得奖励为-1。
- 有四个特殊的奖励状态：当机器人在该状态下执行一个动作时，它会获得相应奖励。
- 在状态（9,8）中，无论做什么动作，它都会被随机投掷到四个角落中的一个。

奖励和价值

假设智能体接收到一系列的奖励$r_1, r_2, r_3, r_4, \ldots$。效用应该如何计算？

- 总奖励$V = \sum_{i=1}^{\infty} r_i$
  但如果总和是无限的，就无法比较这样的序列。
- 平均奖励$V = \lim_{n\to\infty} (r_1 + \cdots + r_n)/n$
  然而，当总奖励是有限的时候，平均奖励为零，因此无法比较这样的序列。
- 折扣回报$V = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$
  在这个准则下，未来的奖励价值小于当前的奖励。

折扣奖励的性质

- 对于奖励$r_1, r_2, r_3, r_4, \ldots$ 的折扣回报为：
$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= r_1 + \gamma(r_2 + \gamma(r_3 + \gamma(r_4 + \ldots))) \end{aligned}$$
- 如果V_t 是从时间步骤t 获得的价值
$$V_t = r_t + \gamma V_{t+1}$$
- $1 + \gamma + \gamma^2 + \gamma^3 + \cdots = 1/(1-\gamma)$
因此$\dfrac{\text{minimum reward}}{1-\gamma} \leq V_t \leq \dfrac{\text{maximum reward}}{1-\gamma}$
- 我们可以用前k 项来近似V，误差为:
$$V - (r_1 + \gamma r_2 + \cdots + \gamma^{k-1} r_k) = \gamma^k V_{k+1}$$

策略(Policies)

- 一个确定性策略是一个函数:

  $$\pi : S \to A$$

  给定一个状态s, $\pi(s)$ 指定了智能体将执行的动作。
- 一个最优策略是一个具有最大期望折扣奖励的策略。

有多少个确定性策略?

- 每周*Sam* 都必须决定是否锻炼:
  - 状态: $\{fit, unfit\}$
  - 动作: $\{exercise, relax\}$
- 100个状态和4个动作的网格世界。

策略的值

给定一个策略π:

- $Q^{\pi}(s,a)$: 在状态s 下执行动作a，然后遵循策略π 的期望价值。
- $V^{\pi}(s)$: 在状态s下遵循策略π的预期值。
- Q^{π} 和V^{π} 可以相互递归定义：

$$\begin{aligned} Q^{\pi}(s,a) &= \sum_{s'} P(s'|a,s)\left(R(s,a,s') + \gamma V^{\pi}(s')\right) \\ V^{\pi}(s) &= Q(s,\pi(s)) \end{aligned}$$

最优策略的值

- $Q^*(s,a)$: 在状态s下执行动作a，然后遵循最优策略的期望价值。
- $V^*(s)$: 在状态s下遵循最优策略的期望价值。
- Q^* 和V^*: 可以相互递归定义：

$$
\begin{aligned}
Q^*(s,a) &= \sum_{s'} P(s'|a,s)\left(R(s,a,s') + \gamma V^*(s')\right) \\
V^*(s) &= \max_a Q^*(s,a) \\
\pi^*(s) &= argmax_a Q^*(s,a)
\end{aligned}
$$

贝尔曼方程（Bellman equations）

$$
\begin{aligned}
V^*(s) &= \max_a \sum_{s'} P(s'|a,s)\,(R(s,a,s') + \gamma V^*(s')) \\
Q^*(s,a) &= \sum_{s'} P(s'|a,s)\left(R(s,a,s') + \gamma \max_{a'} Q^*(s',a')\right)
\end{aligned}
$$

求解V或Q的贝尔曼方程可以找到一个最优策略。

值迭代

- 如果存在n个状态，那么就会有n个贝尔曼方程，每个状态对应一个方程。
- 可以用线性代数技术快速求解线性方程组。
- 然而，这些方程是非线性的，因为“max”操作符不是线性操作符。
- 可以使用一种迭代的方法。

值迭代

- 我们从价值的任意初始值开始。
- 令$V_i(s)$ 表示第i次迭代时状态s的值。
- 使用贝尔曼更新:

$$V_{i+1}(s) = \max_a \sum_{s'} P(s'|a,s)\left(R(s,a,s') + \gamma V_i(s')\right)$$

- 如果无限次应用贝尔曼更新，保证会达到一个均衡点，而且最终的价值是贝尔曼方程的解。
- 事实上，也是唯一的解，而且对应的策略是最优的。

异步值迭代

- 不是逐批更新所有状态，而是单独更新每个状态的值
- 如果每个状态和动作都被访问无穷多次，则会收敛到最优值函数
- 可以存储 $V[s]$ 或 $Q[s,a]$
- Repeat forever:
  - Select sate s
  - $V[s] \leftarrow \max_a \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma V[s']\right)$
- Repeat forever:
  - Select sate s and action a
  - $Q[s,a] \leftarrow \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma \max_{a'} Q[s',a']\right)$
- 异步值迭代比值迭代收敛更快，是一些强化学习算法的基础

示例：是否锻炼？

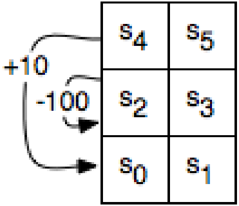
令$\gamma = 0.9$

- 迭代0: $\bar{V} = (0, 0)$
- 迭代1: $\bar{V} = (10, 5)$
  - $(f, e) : 8$, $(f, r) : 10$
  - $(u, e) : 0$, $(u, r) : 5$
- 迭代2: $\bar{V} = (17.65, 9.5)$
  - $(f, e) : 0.99(8 + 0.9 \cdot 10) + 0.01(8 + 0.9 \cdot 5) = 16.955$
  - $(f, r) : 0.7(10 + 0.9 \cdot 10) + 0.3(10 + 0.9 \cdot 5) = 17.65$
  - $(u, e) : 0.2(0.9 \cdot 10) + 0.8(0.9 \cdot 5) = 5.4$
  - $(u, r) : (5 + 0.9 \cdot 5) = 9.5$
- 迭代3: $\bar{V} = (23.812, 13.55)$
  - $(f, e) : 0.99(8+0.9 \cdot 17.65)+0.01(8+0.9 \cdot 9.5) = 23.812$
  - $(f, r) : 0.7(10+0.9 \cdot 17.65)+0.3(10+0.9 \cdot 9.5) = 23.685$
  - $(u, e) : 0.2(0.9 \cdot 17.65) + 0.8(0.9 \cdot 9.5) = 10.017$
  - $(u, r) : (5 + 0.9 \cdot 9.5) = 13.55$

强化学习（Reinforcement learning）

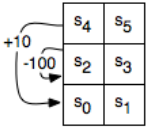
- 类似于决策论规划，但没有给出动态模型和奖励模型。
- 强化学习研究如何让智能体与环境交互，从中学会最优决策。
- 智能体在和环境的交互过程中，根据当前的状态、环境的奖惩而采取相应的动作，通过学习使环境的回报最大化。

一个小例子



- 有6个状态$s_0, \ldots, s_5$.
- 智能体有4个动作: UpC, Up, Left, Right.
- upC ("up carefully"): 上移,除了在状态s4和s5时智能体保持不动并且获得-1的奖励。

一个小例子



- right: 在状态s0、s2和s4向右移动，奖励为0，在其他状态保持不动，奖励为-1。
- left: 在s1、s3和s5向左移动。在s0保持不动，奖励为-1。在s2保持不动，奖励为-100。在s4, 移动到s0，奖励为10。
- up: 以0.8的概率和upC一样的效果，除了奖励为0外。以0.1的概率和left一样效果，以0.1的概率和right一样效果。

智能体应该如何行动？

强化学习的基本元素

- 环境（Environment）
- 状态（State）
- 动作（Action）
- 奖励（Reward）
- 策略（Policy）
- 值函数（Value function）

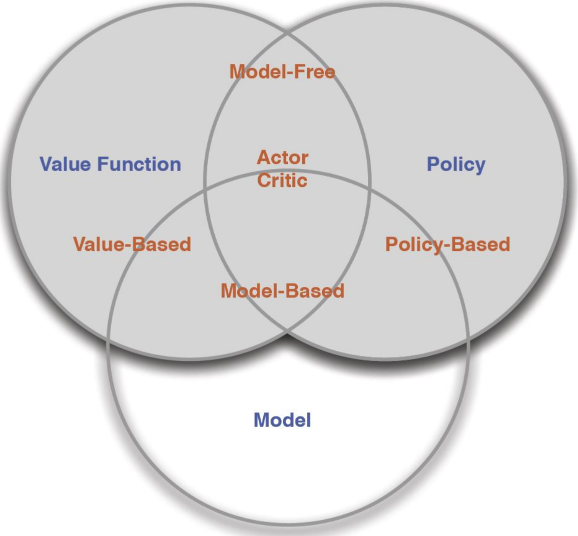
强化学习的主要方法

分类1

- 基于值的：没有策略，有值函数
- 基于策略的：有策略，没有值函数
- 演员-评论家（Actor-Critic）：有策略，有值函数

分类2

- 无模型的（Model-Free）：使用策略和/或值函数，没有模型
- 基于模型的（Model-Based）：使用策略和/或值函数，有模型


Model-Free
Value Function
Actor
Critic
Policy
Value-Based
Policy-Based
Model-Based
Model

经验异步值迭代

initialize $Q[S, A]$ arbitrarily
observe current state s
repeat forever:

select and carry out an action a
observe reward r and state s'
$Q[s, a] \leftarrow r + \gamma \max_{a'} Q[s', a']$
$s \leftarrow s'$

时序差分(Temporal Differences)

- 假设我们有一系列的值：$v_1, v_2, v_3, \ldots,$ 目标是根据所有先前的值来预测下一个值。
- 一种方法是取前k 个值的平均值：

$$A_k = \frac{v_1 + \cdots + v_k}{k}$$

- *e.g.*, 给定一系列学生成绩并旨在预测下一个成绩，一个合理的预测是平均成绩。

Temporal Differences (cont)

- 假设我们已知A_{k-1}，一个新值v_k 到达：
$$A_k = \frac{v_1 + \cdots + v_{k-1} + v_k}{k} = \frac{k-1}{k} A_{k-1} + \frac{1}{k} v_k$$
- 令$\alpha_k = \frac{1}{k}$，则
$$A_k = (1 - \alpha_k) A_{k-1} + \alpha_k v_k = A_{k-1} + \alpha_k (v_k - A_{k-1})$$
- $v_k - A_{k-1}$ 被称为时序差分错误或TD 错误
- 它指示了新值v_k 与旧预测A_{k-1} 之间的差异有多大。

TD 公式

$A_k = A_{k-1} + \alpha_k(v_k - A_{k-1})$

- 为了获得新的估计值，对旧估计值增加α_k 乘TD 错误。
- 思路：如果新值比旧预测值更高，增加预测值；
- 如果新值比旧预测值更低，减少预测值。

α_k 的选择

- 设置$\alpha_k = \frac{1}{k}$ 假设所有的值具有相等的权重
- 在强化学习中，v_i 后面的值比前面的值更准确，应该给予更多权重
- 一种将后面的样例赋予更多权重的方法是将α 设置为一个常数$(0 < \alpha \leq 1)$。
- 但是这种方法不会收敛到平均值
- 以下条件保证收敛

$$\sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

Q-learning

- 思路：存储$Q[State, Action]$; 像异步值迭代那样更新它，但使用经验
- 假设智能体有一个经验$\langle s, a, r, s' \rangle$
- 这为更新$Q[s, a]$ 提供了一条数据。
- 经验$\langle s, a, r, s' \rangle$ 为$Q^*(s, a)$ 提供了一个新的估计：

$$r + \gamma \max_{a'} Q[s', a']$$

这可以用于TD 公式，得到：

$$Q[s, a] \leftarrow Q[s, a] + \alpha \left(r + \gamma \max_{a'} Q[s', a'] - Q[s, a] \right)$$

Q-learning

initialize $Q[S, A]$ arbitrarily
observe current state s
repeat forever:
    select and carry out an action a
    observe reward r and state s'
    $Q[s, a] \leftarrow Q[s, a] + \alpha\,(r + \gamma\,\max_{a'} Q[s', a'] - Q[s, a])$
    $s \leftarrow s'$

小例子

令$\gamma = 0.9$, $\alpha = 0.2$; 所有的Q 值初始化为0.

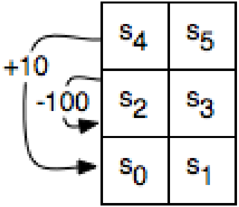
下面是一系列经验和更新的过程：

s	a	r	s'	Update
s0	upC	-1	s2	Q[s0,upC]=-0.2
s2	up	0	s4	Q[s2,up]=0
s4	left	10	s0	Q[s4,left]=2.0
s0	upC	-1	s2	Q[s0,upC]=-0.36
s2	up	0	s4	Q[s2,up]=0.36
s4	left	10	s0	Q[s4,left]=3.6
s0	up	0	s2	Q[s0,upC]=0.06
s2	up	-100	s2	Q[s2,up]=-19.65
s2	up	0	s4	Q[s2,up]=-15.07
s4	left	10	s0	Q[s4,left]=4.89

$$0.8 \times 0.36 + 0.2 \times (-100 + 0.9 \times 0.36) = -19.65$$

$$0.8 \times -19.65 + 0.2 \times (0 + 0.9 \times 3.6) = -15.07$$

小例子



最优策略

- up in state s0, upC in state s2,
- up in states s1 and s3,
- and left in states s4 and s5.

Q-learning的性质

- Q-learning会收敛到一个最优策略，只要智能体在每个状态下尝试每个动作足够多次。
- 但智能体应该如何选择？
  - 利用（exploit）：在状态s下，选择使$Q[s,a]$最大化的动作。
  - 探索（explore）：选择其他动作。

探索与利用（Exploration and Exploitation）

- 强化学习类似于试错学习。
- 智能体应该通过与环境的交互来发现一个好的策略，同时尽量减少损失的奖励。
- 探索可以获得关于环境的更多信息。
- 利用可以利用已知信息来最大化奖励。
- 通常来说，探索和利用同样重要。

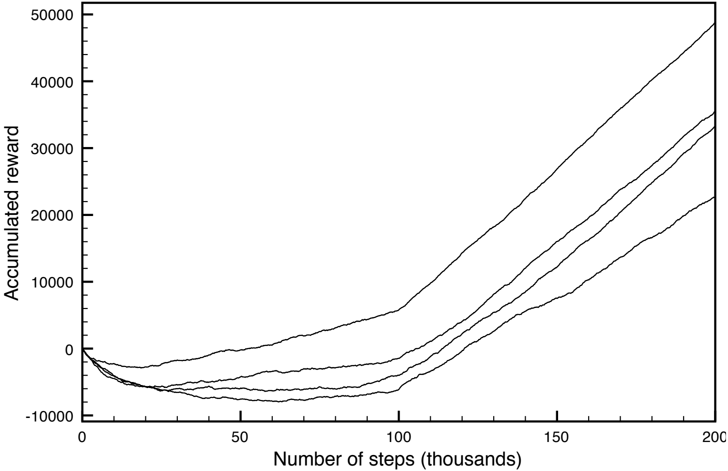
探索策略

- ϵ-贪心策略（ϵ-greedy strategy）
  以概率ϵ 随机选择一个动作，以概率$1-\epsilon$ 选择最佳动作。
- softmax动作选择：在状态s 下，以概率

  $$\frac{e^{Q[s,a]/\tau}}{\sum_a e^{Q[s,a]/\tau}}$$

  选择动作a，其中$\tau > 0$ 是温度.
  较好的动作被选择的概率比较差的动作更高。
  τ 定义了Q-值差异对概率差异的影响程度

评估强化学习算法

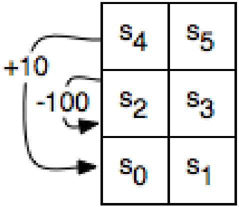


一个算法优于另一个算法，如果其折线始终位于另一个的上方

同策略（On-policy）学习

- Q-learning是一种异策略（Off-policy）学习方法：它学习的是最优策略的价值
- 如果探索策略具有危险性（存在较大的负回报），这会比较糟糕。
- 同策略学习学习的是正在遵循的策略的价值。例如，80% 的时间进行贪婪动作选择，20% 的时间进行随机动作选择。
- 为什么呢？如果智能体真的要进行探索，优化实际要执行的策略可能会更好。
- SARSA使用经验$\langle s, a, r, s', a' \rangle$ 来更新$Q[s, a]$，其中a' 是智能体在s'中决定要执行的动作。

小例子

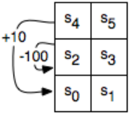


- 最优策略是在状态s0中向上移动
- 然而，如果智能体正在进行探索，这可能不是一个好的选择，因为
- 从状态s2开始进行探索是非常危险的：向左移动会得到一个−100的回报。

SARSA (state-action-reward-state-action)

initialize $Q[S, A]$ arbitrarily
observe current state s
select action a using a policy based on Q
repeat forever:
    carry out action a
    observe reward r and state s'
    select action a' using a policy based on Q
    $Q[s, a] \leftarrow Q[s, a] + \alpha\,(r + \gamma Q[s', a'] - Q[s, a])$
    $s \leftarrow s'$
    $a \leftarrow a'$

小例子



Algorithm	$Q[s_0, right]$	$Q[s_0, up]$	$Q[s_2, upC]$	$Q[s_2, up]$	$Q[s_4, left]$
Q-learning	19.48	23.28	26.86	16.9	30.95
SARSA (20%)	9.27	7.9	14.8	4.43	18.09
SARSA (10%)	13.04	13.95	18.9	8.93	22.47

- 使用SARSA算法进行20%的探索时，最优策略是在状态s0中向右移动。
- 使用10%的探索时，最优策略是在s0中向上移动。
- 然而，如果减少探索，找到最优策略可能需要更长的时间。

Q值更新公式的总结

- 值迭代: 已知$P(s'|s,a)$ 和$R(s,a,s')$

$$Q[s,a] \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q[s',a'])$$

- Q-learning: 使用经验$\langle s,a,r,s' \rangle$

$$Q[s,a] \leftarrow Q[s,a] + \alpha \left(r + \gamma \max_{a'} Q[s',a'] - Q[s,a] \right)$$

- Sarsa: 使用经验$\langle s,a,r,s',a' \rangle$

$$Q[s,a] \leftarrow Q[s,a] + \alpha \left(r + \gamma Q[s',a'] - Q[s,a] \right)$$

策略函数

- 目前考虑的策略函数是确定的：$\pi : S \to A$，其中$\pi(s)$表示在状态s下采取的动作
- 更一般的策略函数$\pi : S \times A \to [0, 1]$，其中$\pi(s, a)$表示在状态$s$下采取动作$a$的概率

$$G = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$$

- 价值函数（**value function**）：$V: S \mapsto \mathbb{R}$，其中$V_\pi(s) = \mathbb{E}_\pi[G|S = s]$，表示智能体在处于状态$s$时，按照策略$\pi$采取行动时所获得回报的期望。
- 动作-价值函数(**action-value function**)：$q: S \times A \mapsto \mathbb{R}$，其中$q_\pi(s, a) = \mathbb{E}_\pi[G|S = s, A = a]$，表示智能体处于状态$s$时，选择了动作$a$后，根据策略$\pi$采取行动所获得回报的期望。

强化学习的形式化

- 策略$\pi : S \times A \to [0, 1]$, 其中$\pi(s, a)$是在状态$s$下采取动作$a$的概率
- 值函数$V : S \to \mathbb{R}$, 其中$V_\pi(s) = \mathbb{E}_\pi[G|S = s]$, 表示在状态$s$下遵循$\pi$的期望折扣奖励
- 给定一个环境MDP (S, A, P, R, γ), 其中P 和R 对于智能体来说是未知的，寻找一个最大化$V_\pi(s_0)$的策略π。

基于策略的强化学习

- 基于价值的强化学习：以对价值函数V或动作-价值函数Q的建模为核心
- 基于策略的强化学习：直接参数化策略函数，求解参数化的策略函数的梯度
- 参数化策略函数可以表示为$\pi_\theta(s,a)$，其中θ为一组参数

策略梯度定理

最大化目标： $MAX\ J(\boldsymbol{\theta}) := V_{\pi_{\boldsymbol{\theta}}}(s_0)$

策略梯度定理（**Policy Gradient Theorem**）：

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \propto \sum_s \mu_{\pi_{\boldsymbol{\theta}}}(s) \sum_a q_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s,a)$$

- 其中$\mu_{\pi_\theta}(s)$称为状态的平稳分布

$$\mu_{\pi_\theta}(s) = \sum_{t=0}^{+\infty} \gamma^t p(s_t = s | s_0, \pi)$$

基于蒙特卡洛采样的策略梯度法

首先要对策略梯度公式进行如下的适当变形：

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &\propto \sum_s \mu_{\pi_{\boldsymbol{\theta}}}(s) \sum_a q_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s,a) \\
&= \sum_s \mu_{\pi_{\boldsymbol{\theta}}}(s) \sum_a \pi_{\boldsymbol{\theta}}(s,a) \left[q_{\pi_{\boldsymbol{\theta}}}(s,a) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}}(s,a)} \right] \\
&= \mathbb{E}_{s,a\sim\pi} \left[q_{\pi_{\boldsymbol{\theta}}}(s,a) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(s,a)}{\pi_{\boldsymbol{\theta}}(s,a)} \right] \\
&= \mathbb{E}_{s,a\sim\pi} \left[q_{\pi_{\boldsymbol{\theta}}}(s,a) \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a) \right] \quad \text{（这里利用了} \nabla_{\boldsymbol{\theta}} \ln Z = \frac{1}{Z} \nabla_{\boldsymbol{\theta}} Z \text{)}
\end{aligned}$$

进一步将公式变形

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &\propto \mathbb{E}_{s,a\sim\pi} \left[\mathbb{E}_{\mathcal{T}(s,a)\sim\pi} [G_t | s,a] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a) \right] \\
&= \mathbb{E}_{s,a,\mathcal{T}(s,a)\sim\pi} \left[G_t \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a) \right]
\end{aligned}$$

其中$\mathcal{T}(s,a)$表示从状态s开始执行动作a得到的一条轨迹（不包括s和a），G_t为从状态s开始沿着轨迹$\mathcal{T}(s,a)$运动所得回报。

可以使用蒙特卡洛采样法来求解公式，即算法只需根据策略来采样一个状态s、一个动作a和将来的轨迹，就能构造公式中求取期望所对应的一个样本。

REINFORCE算法

随机初始化 $\boldsymbol{\theta}$
repeat
  根据策略 $\pi_{\boldsymbol{\theta}}$ 采样一个片段 $s_0, a_0, R_1, s_1, \cdots, s_{T-1}, a_{T-1}, R_T$
  for $t \leftarrow 0$ **to** $T-1$ **do**
    $G \leftarrow \sum_{k=1}^{T-t} \gamma^{k-1} R_{t+k}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \gamma^t G \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s_t, a_t)$
  end
until $\boldsymbol{\theta}$ 收敛

一个从初始状态到终止状态的完整轨迹称为一个片段或回合（episode）

Actor-Critic算法

- 结合基于价值的方法和基于策略的方法
- Actor(演员): 策略函数$\pi_\theta(a, s)$，根据Critic的评分修改选行为的概率
- Critic(评论员)：价值函数$V_w(s)$，基于Actor的行为评判行为的得分
- 可以单步更新参数，不需要等到回合结束

基于时序差分的策略梯度法：Actor-Critic算法

Critic以$R+\gamma V_w(s')$为目标，极小化平方错误

随机初始化 $\boldsymbol{\theta}, \boldsymbol{w}$
repeat
  $s \leftarrow$ 初始状态
  $t \leftarrow 0$
  repeat
    $a \sim \pi_{\boldsymbol{\theta}}(s,\cdot)$
    执行动作 a，观察奖励 R 和下一时刻状态 s'
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}}\gamma^t[R+\gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)]\nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}}\gamma^t[R+\gamma V_{\boldsymbol{w}}(s')]\nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s,a)$
    $t \leftarrow t+1$
  until s 是终止状态
until θ 收敛

Advantage Actor-Critic算法：A2C

与AC算法的不同：actor网络求梯度时使用优势函数$Q(s,a) - V(s)$，表达在状态s下，动作a相对于平均而言的优势

随机初始化 $\boldsymbol{\theta}, \boldsymbol{w}$
repeat
  $s \leftarrow$ 初始状态
  $t \leftarrow 0$
  repeat
    $a \sim \pi_{\boldsymbol{\theta}}(s, \cdot)$
    执行动作 a，观察奖励 R 和下一时刻状态 s'
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta_{\boldsymbol{w}} \gamma^t [R + \gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)] \nabla_{\boldsymbol{w}} V_{\boldsymbol{w}}(s)$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \gamma^t [R + \gamma V_{\boldsymbol{w}}(s') - V_{\boldsymbol{w}}(s)] \nabla_{\boldsymbol{\theta}} \ln \pi_{\boldsymbol{\theta}}(s, a)$
    $t \leftarrow t + 1$
  until s 是终止状态
until θ 收敛