RLChina 2024

RLChina 2024（广州） 3小时教学课程

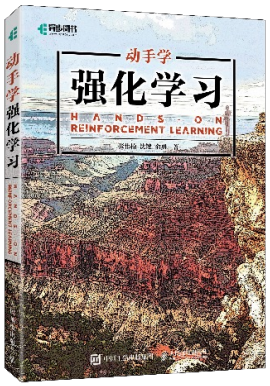
强化学习入门课程

张伟楠 – 上海交通大学

所有学习材料请见：
讲义：https://wnzhang.net/teaching/sjtu-rl-2024
视频：https://space.bilibili.com/3546754433681656
动手学：https://hrl.boyuai.com

讲者主页：
http://wnzhang.net

《动手学强化学习》



目标：减小RL学习原理与实践的gap

- 基于python notebook，给出可重复的实验代码
- 完全对齐教学的原理点和作业的实践点

前言

强化学习基础篇

多臂老虎机

• 马尔可夫决策过程

动态规划算法

时序差分算法

Dyna-Q算法

强化学习进阶篇

DQN 算法

DQN 改进算法

策略梯度算法

Actor-Critic 算法

敬请期待

强化学习前沿篇

敬请期待

写在最后

第 3 章 马尔可夫决策过程

运行@天池　运行@和鲸

3.1 简介

马尔可夫决策过程（Markov decision process，MDP）是强化学习的重要概念。要学好强化学习，我们首先要掌握马尔可夫决策过程的基础知识。前两章所说的强化学习中的环境一般就是一个马尔可夫决策过程。与多臂老虎机问题不同，马尔可夫决策过程包含状态信息以及状态之间的转移机制。如果要用强化学习去解决一个实际问题，第一步要做的事情就是把这个实际问题抽象为一个马尔可夫决策过程，也就是明确马尔可夫决策过程的各个组成要素。本章将从马尔可夫过程出发，一步一步地进行介绍，最后引出马尔可夫决策过程。

3.2 马尔可夫过程

3.2.1 随机过程

随机过程（stochastic process）是概率论的"动力学"部分。概率论的研究对象是静态的随机现象，而随机过程的研究对象是随时间演变的随机现象（例如天气随时间的变化、城市交通随时间的变化）。在随机过程中，随机现象在某时刻t的取值是一个向量随机变量，用S_t表示，所有可能的状态组成状态集合$\mathcal{S}$。随机现象便是状态的变化过程。在某时刻t的状态S_t通常取决于t时刻之前的状态。我们将已知历史信息$(S_1,\dots,S_t)$时下一个时刻状态为S_{t+1}的概率表示成$P(S_{t+1}|S_1,\dots,S_t)$。

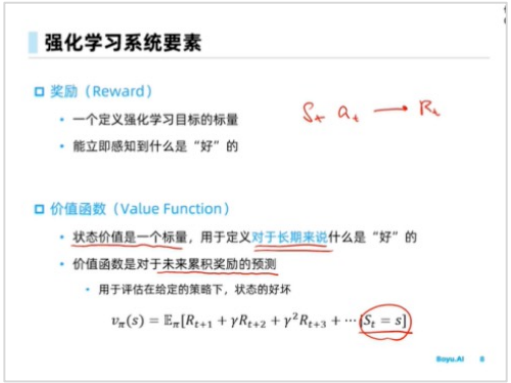
3.2.2 马尔可夫性质

当且仅当某时刻的状态只取决于上一时刻的状态时，一个随机过程被称为具有**马尔可夫性质**（Markov property），用公式表示为$P(S_{t+1}|S_t)=P(S_{t+1}|S_1,\dots,S_t)$。也就是说，当前状态是未来的充分统计量，即下一个状态只取决于当前状态，而不会受到过去状态的影响。需要明确的是，具有马尔可夫性并不代表这个随机过程就和历史完全没有关系。因为虽然$t+1$时刻的状态只与t时刻的状态有关，但是t时刻的状态其实包含了$t-1$时刻的状态的信息，通过这种链式的关系，历史的信息被传递到了现在。马尔可夫性可以大大简化运算，因为只要当前状态可知，所有的历史信息都不再需要了，利用当前状态信息就可以决定未来。

3.2.3 马尔可夫过程

马尔可夫过程（Markov process）指具有马尔可夫性质的随机过程，也被称为**马尔可夫链**（Markov chain）。我们通常用元组$(\mathcal{S},\mathcal{P})$描述一个马尔可夫过程，其中$\mathcal{S}$是有限数量的状态集合，$\mathcal{P}$是**状态转移矩阵**（state transition matrix）。假设一共有n个状态，此时$\mathcal{S}=\{s_1,s_2,\dots,s_n\}$。状态转移矩阵$\mathcal{P}$定义了所有状态对之间的转移概率，即

3.1 简介
3.2 马尔可夫过程
3.2.1 随机过程
3.2.2 马尔可夫性质
3.2.3 马尔可夫过程
3.3 马尔可夫奖励过程
3.3.1 回报
3.3.2 价值函数
3.4. 马尔可夫决策过程
3.4.1 策略
3.4.2 状态价值函数
3.4.3 动作价值函数
3.4.4 贝尔曼期望方程
3.5 蒙特卡洛方法
3.6 占用度量
3.7 最优策略
3.7.1 贝尔曼最优方程
3.8 总结
3.9 参考文献



强化学习系统要素

- 奖励（Reward）
  - 一个定义强化学习目标的标量
  - 能立即感知到什么是“好”的
- 价值函数（Value Function）
  - 状态价值是一个标量，用于定义对于长期来说什么是“好”的
  - 价值函数是对于未来累积奖励的预测
    - 用于评估在给定的策略下，状态的好坏

$$v_\pi(s)=\mathbb{E}_\pi[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\cdots|S_t=s]$$

Boyu.AI

选择题

1.在策略学习过程中，往往需要进行新策略探索与旧策略的利用，其目的分别是什么？

- 尝试不同策略，以进行策略提升/提升对旧策略的评估能力
- 提高策略多样性/让旧策略朝着最优策略的方向进行优化

2.以下说法正确的是

- 策略探索在策略学习过程中总是有利的
- 一个具有次线性total regret收敛保证的策略探索算法总是能够在有限时间内让强化学习算法收敛
- 在MAB问题中，使用增量式蒙特卡洛进行奖励值估计能够使得算法的时间复杂度从$O(N)$优化至$O(1)$

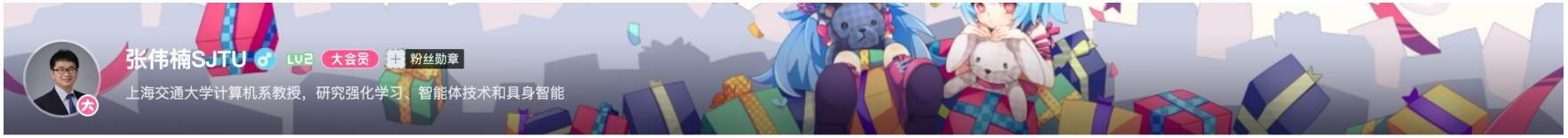
3.有关课程中讲解的几种策略探索方法，下面说法不正确的是

- 基于不确定性测度的方法，通常被选择次数越多的动作，其不确定性越低
- 对于积极初始化方法，虽然随着采样次数的增加，其估计偏差会越来越低，但仍然可能会面临一个收敛到局部最优的情况
- ϵ-greedy算法具有次线性收敛保证

填空题

1.对于ϵ-greedy策略探索方式，更高的ϵ？优于更低ϵ探索方式让算法获得的最终奖励值

B站up主 张伟楠SJTU 的《上交强化学习课》



主页 动态 投稿 33 合集和列表 1 收藏 1

关注数	粉丝数	获赞数	播放数
1	6842	1877	6.3万

我的合集和视频列表 › 上交强化学习课

播放全部

33个视频 | 9-30更新

课堂实录-上海交通大学2024年春季强化学习课程

默认排序 升序排序 编辑

添加视频

上海交大张伟楠强化学习课程第1
讲：强化学习简介I
1.2万 9-7

上海交大张伟楠强化学习课程第1
讲：强化学习简介II
2122 9-7

上海交大张伟楠强化学习课程第1
讲：强化学习简介III
4830 9-7

上海交大张伟楠强化学习课程第2
讲：探索与利用
1714 9-7

上海交大张伟楠强化学习课程第3
讲：马尔可夫决策过程I
3755 9-7

上海交大张伟楠强化学习课程第3
讲：马尔可夫决策过程II
1541 9-7

上海交大张伟楠强化学习课程第4
讲：动态规划
964 9-7

上海交大张伟楠强化学习课程第5
讲：值函数估计I
750 9-7

上海交大张伟楠强化学习课程第5
讲：值函数估计II
662 9-7

上海交大张伟楠强化学习课程第6
讲：无模型控制方法I
799 9-7

上海交大张伟楠强化学习课程第6
讲：无模型控制方法II
3408 9-7

https://space.bilibili.com/3546754433681656

课程大纲

强化学习基础

1. **强化学习技术概览**

白盒环境强化学习
2. 马尔可夫决策过程
3. 动态规划

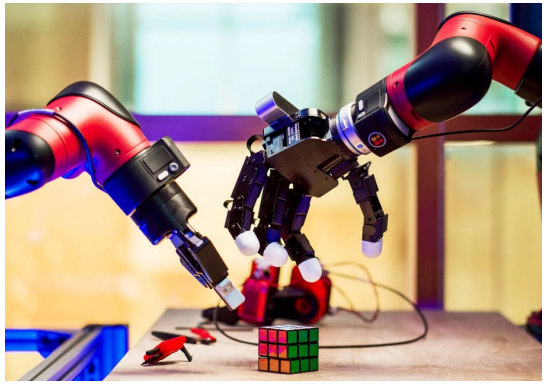
黑盒环境强化学习
4. 值函数估计
5. 无模型控制方法

参数化近似方法
6. 参数化值函数
7. 策略梯度

深度强化学习
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

两种人工智能任务类型

- 预测型任务
  - 根据数据预测所需输出（有监督学习）
  - 生成数据实例（无监督学习）
- 决策型任务
  - 在动态环境中采取行动（强化学习）
    - 转变到新的状态
    - 获得即时奖励
    - 随着时间的推移最大化累计奖励
    - Learning from interaction in a trial-and-error manner



决策智能的任务和技术分类

- 根据决策环境的动态性和透明性，决策任务大致分为以下四个部分，对应具体的技术方案

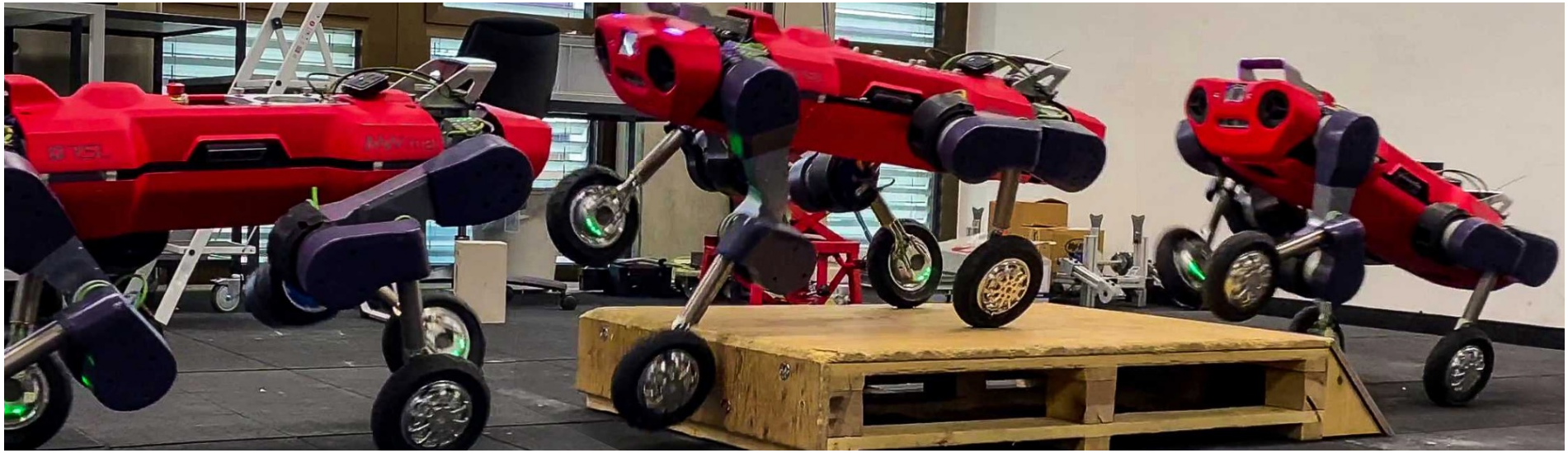
环境特性	**白盒环境** • 变量和目标之间的关系可以用具体公式表示	**黑盒环境** • 变量和目标之间的关系无法用具体公式表示
静态环境 • 环境没有转移的状态 • 单步决策	运筹优化 • （混合整数）线性规划 • 非线形优化	黑盒优化 • 神经网络替代模型优化 • 贝叶斯优化
动态环境 • 环境有可转移的状态 • 多步决策	动态规划 • MDP直接求解 • 树、图搜索	强化学习 • 策略优化 • Bandits、序贯黑盒

序贯决策

序贯决策（Sequential Decision Making）

- 序贯决策中，智能体序贯地做出一个个决策，并接续看到新的观测，直到最终任务结束

智能体   动态环境 $\max_{\pi} \mathbb{E}_{\pi,\mathrm{Env}}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right]$

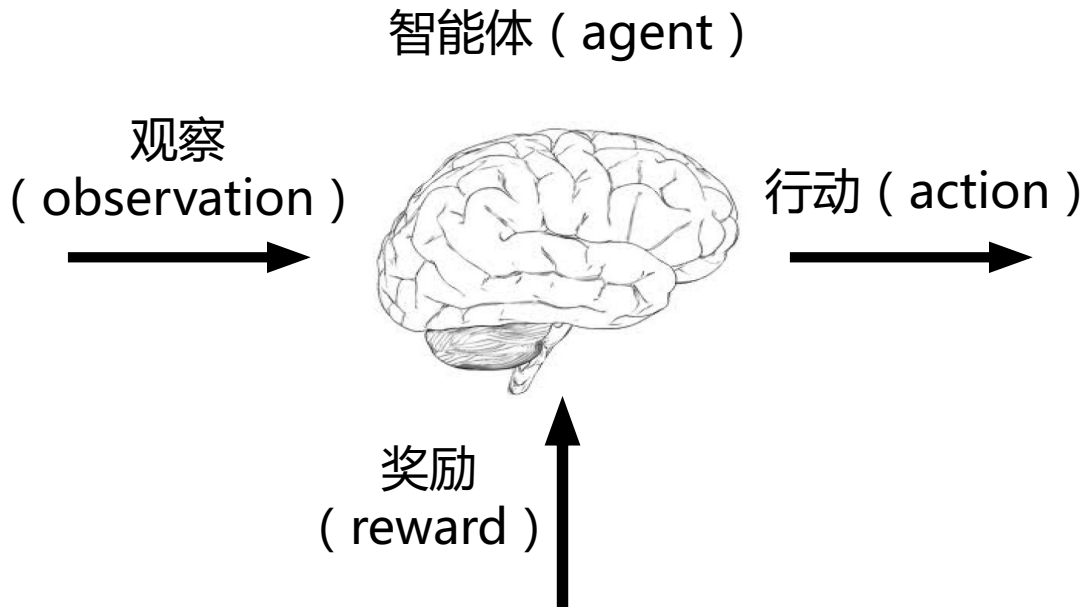


机器狗例子：操作轮足和地形持续交互，完成越过障碍物的任务

绝大多数序贯决策问题，可以用强化学习来解

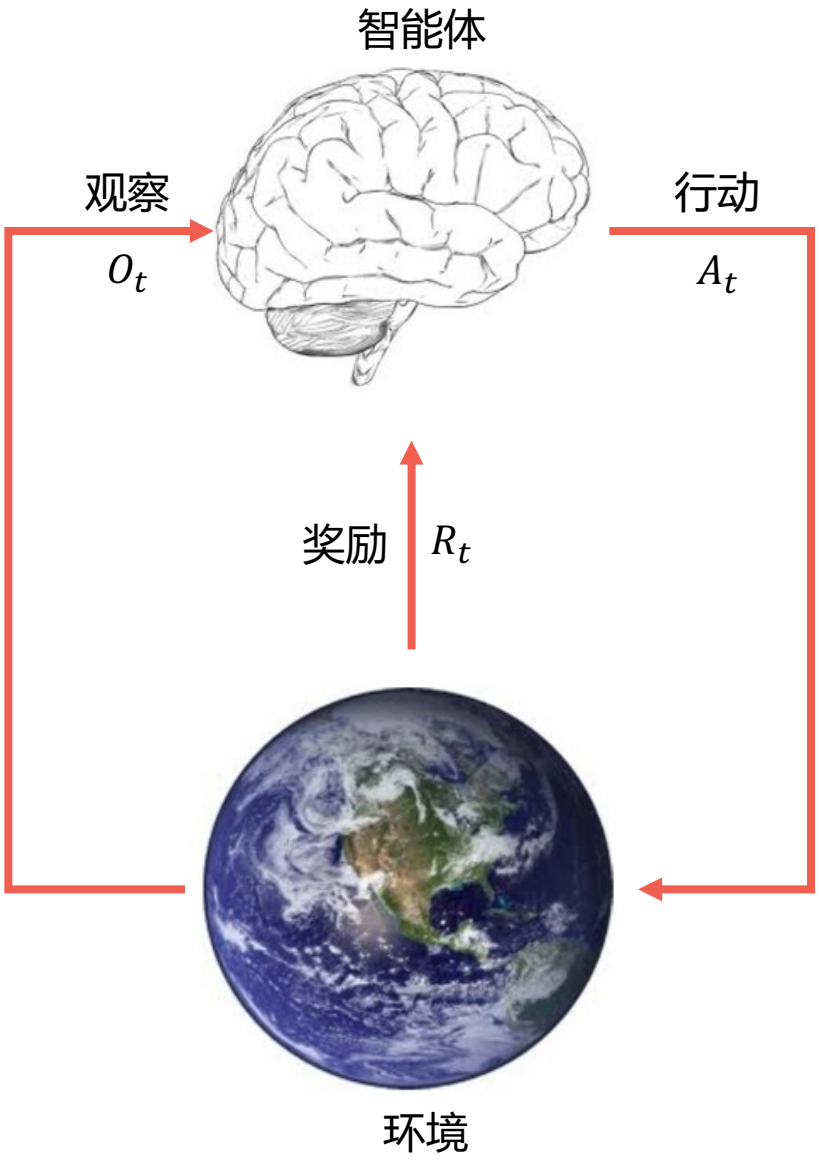
强化学习定义

- 通过从交互中学习来实现目标的计算方法



- 三个方面：
  - 感知：在某种程度上感知环境的状态
  - 行动：可以采取行动来影响状态或者达到目标
  - 目标：随着时间推移最大化累积奖励

强化学习交互过程



- 在每一步 t，智能体：
  - 获得观察 O_t
  - 获得奖励 R_t
  - 执行行动 A_t
- 环境：
  - 获得行动 A_t
  - 给出观察 O_{t+1}
  - 给出奖励 R_{t+1}
- t 在环境这一步增加

在与动态环境的交互中学习

有监督、无监督学习

Model ⟵

Fixed Data

强化学习

Agent ⟷

Dynamic Environment

Agent不同，交互出的数据也不同！

课程大纲

强化学习基础

1. 强化学习技术概览
2. **马尔可夫决策过程**
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化值函数
7. 策略梯度
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

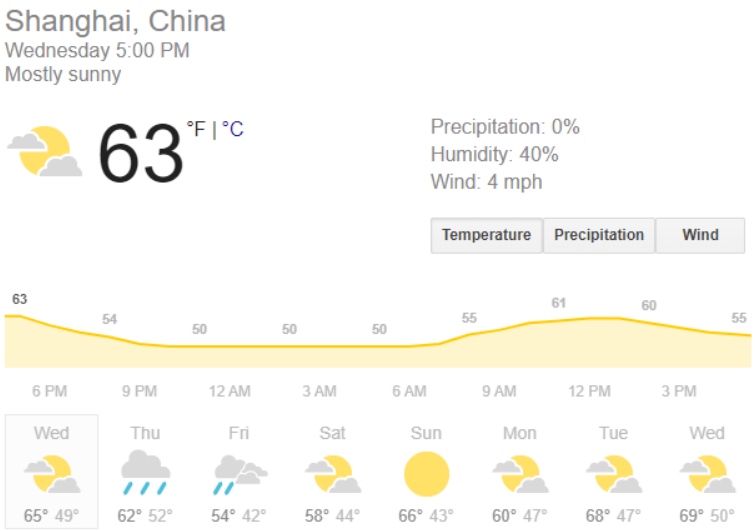
随机过程

- 随机过程是一个或多个事件、随机系统或者随机现象随时间发生演变的过程

$$\mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

  - 概率论研究静态随机现象的统计规律
  - 随机过程研究动态随机现象的统计规律

布朗运动



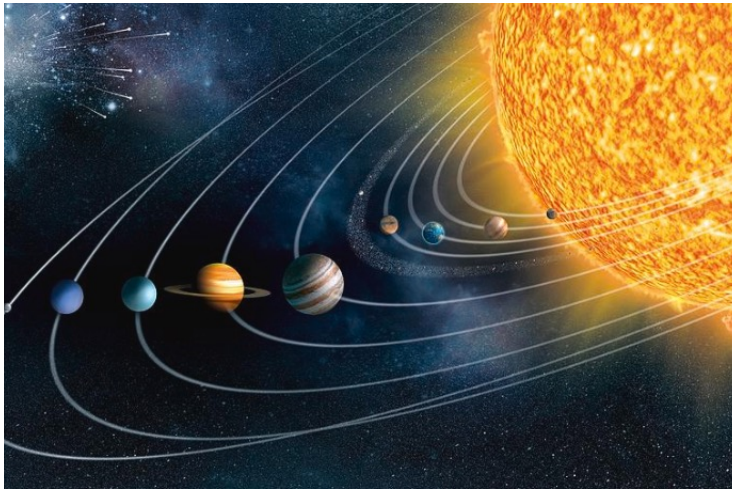
天气变化

随机过程

足球比赛

城市交通

生态系统



星系

马尔可夫过程

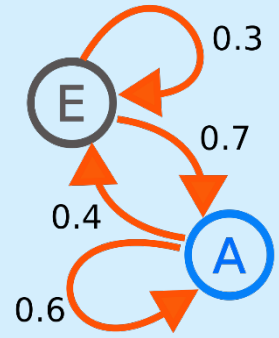
- 马尔可夫过程（Markov Process）是具有马尔可夫性质的随机过程

  “The future is independent of the past given the present”

- 定义：
  - 状态S_t是马尔可夫的，当且仅当

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

- 性质：
  - 状态从历史（history）中捕获了所有相关信息
  - 当状态已知的时候，可以抛开历史不管
  - 也就是说，当前状态是未来的充分统计量



一个具有两个转换状态的马尔可夫链

马尔可夫决策过程

- 马尔可夫决策过程（Markov Decision Process，MDP）
  - 提供了一套为在结果部分随机、部分在决策者的控制下的决策过程建模的数学框架

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

$$\mathbb{P}[S_{t+1}|S_t, A_t]$$

- MDP形式化地描述了一种强化学习的环境
  - 环境完全可观测
  - 即，当前状态可以完全表征过程（马尔可夫性质）

MDP五元组

- MDP可以由一个五元组表示 $(\mathrm{S}, A, \{P_{sa}\}, \gamma, R)$
  - S是状态的集合
    - 比如，迷宫中的位置，Atari游戏中的当前屏幕显示
  - A是动作的集合
    - 比如，向N、E、S、W移动，手柄操纵杆方向和按钮
  - P_{sa}是状态转移概率
    - 对每个状态$s \in S$和动作$a \in A$，P_{sa}是下一个状态在S中的概率分布
  - $\gamma \in [0,1]$是对未来奖励的折扣因子
  - $R: S \times A \mapsto \mathbb{R}$ 是奖励函数
    - 有时奖励只和状态相关

MDP的动态

- MDP的动态如下所示：
  - 从状态s_0开始
  - 智能体选择某个动作$a_0 \in A$
  - 智能体得到奖励$R(s_0, a_0)$
  - MDP随机转移到下一个状态$s_1 \sim P_{s_0 a_0}$
  - 这个过程不断进行

$$s_0 \xrightarrow{a_0, R(s_0,a_0)} s_1 \xrightarrow{a_1, R(s_1,a_1)} s_2 \xrightarrow{a_2, R(s_2,a_2)} s_3 \cdots$$

  - 直到终止状态s_T出现为止，或者无止尽地进行下去
  - 智能体的总回报为

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots$$

MDP的动态性

- 在部分情况下，奖励只和状态相关
  - 比如，在迷宫游戏中，奖励只和位置相关
  - 在围棋中，奖励只基于最终所围地盘的大小有关
- 这时，奖励函数为 $R(s): S \mapsto \mathbb{R}$
- MDP的过程为

$$s_0 \xrightarrow{a_0, R(s_0)} s_1 \xrightarrow{a_1, R(s_1)} s_2 \xrightarrow{a_2, R(s_2)} s_3 \quad \cdots$$

- 累积奖励为

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$

和动态环境交互产生的数据分布

- 给定同一个动态环境（即MDP），不同的策略采样出来的(状态-行动)对的分布是不同的

- 占用度量（Occupancy Measure）

Indicator function $\mathbb{I}(z) = 1$ 表示事件z发生，否则取0

$$\rho^{\pi}(s,a) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \, \mathbb{I}(S_t = s, A_t = a) \mid \pi\right], \forall s \in S, a \in A$$

$$= \sum_{t=0}^{T} \gamma^t \, P(S_t = s, A_t = a | s_0, \pi)$$

占用度量和策略

- 占用度量（Occupancy Measure）

$$\rho^{\pi}(s,a) = \mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s, a_t = a)\right]$$

- 定理1：和同一个动态环境交互的两个策略π_1和π_2得到的占用度量ρ^{π_1}和ρ^{π_2}满足

$$\rho^{\pi_1} = \rho^{\pi_2} \text{ 当且仅当 } \pi_1 = \pi_2$$

- 定理2：给定一占用度量ρ，可生成该占用度量的唯一策略是

$$\pi_{\rho} = \frac{\rho(s,a)}{\sum_{a'}\rho(s,a')}$$

U. Syed, M. Bowling, and R. E. Schapire. Apprenticeship learning using linear programming. ICML 2008.

占用度量和策略

- 占用度量（Occupancy Measure）

$$\rho^{\pi}(s,a) = \mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s, a_t = a)\right]$$

- 状态占用度量

$$\rho^{\pi}(s) = \mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s)\right]$$

$$= \mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s)\sum_{a'} p(a_t = a'|s_t = s)\right]$$

$$= \sum_{a'}\mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s, a_t = a'))\right]$$

$$= \sum_{a'}\rho^{\pi}(s,a')$$

U. Syed, M. Bowling, and R. E. Schapire. Apprenticeship learning using linear programming. ICML 2008.

占用度量和累计奖励

- 占用度量（Occupancy Measure）

$$\rho^{\pi}(s,a) = \mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s, a_t = a)\right]$$

- 策略的累积奖励为

$$V(\pi) = \mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}[R(s_0,a_0) + \gamma R(s_1,a_1) + \gamma^2 R(s_2,a_2) + \cdots]$$

$$= \sum_{s,a}\mathbb{E}_{a\sim\pi(\cdot|s),\, s'\sim p(\cdot|s,a)}\left[\sum_{t=0}^{T}\gamma^{t}\, p(s_t = s, a_t = a)\right]R(s,a)$$

$$= \sum_{s,a}\rho^{\pi}(s,a)R(s,a) \;=\; \mathbb{E}_{\pi}[R(s,a)]$$

强化学习中的简写

U. Syed, M. Bowling, and R. E. Schapire. Apprenticeship learning using linear programming. ICML 2008.

课程大纲

强化学习基础

1. 强化学习技术概览
2. 马尔可夫决策过程
3. **策略提升与动态规划**
4. 值函数估计
5. 无模型控制方法
6. 参数化值函数
7. 策略梯度
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

MDP中策略的目标

- 策略学习的目标：选择能够最大化累积奖励期望的动作

$$\max_{\pi} \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots \mid s_0, \pi]$$

$$= \sum_{s,a} \rho^{\pi}(s, a)\, r(s, a)$$

- 如何达到以上学习目标？
  - 策略π和其占用度量ρ^{π}的对应关系是黑盒的，因此以上优化目标并没有直接对π更新方向的指导
  - 在每一个状态s下，策略改变了动作的选择后，策略整体是否变得更优秀了？

“思想总是走在行动的前面，就好像闪电总是走在雷鸣之前。”

德国诗人海涅

策略评估与策略提升

讲师：张伟楠 – 上海交通大学

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值 $V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$

$$= \mathbb{E}_{a \sim \pi(s)} \left[r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s') \right]$$

$$= \mathbb{E}_{a \sim \pi(s)} [Q^{\pi}(s,a)]$$

动作价值 $Q^{\pi}(s,a) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, A_0 = a, \pi]$

$$= r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

策略提升（Policy Improvement）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有

$$Q^{\pi}\left(s,\pi'(s)\right) \geq V^{\pi}(s)$$

以π来记回报

- 一种特例：给定环境MDP和两个策略π，π'，如果满足如下性质：
  1. 在某个状态s下，两策略的输出不同，并且有

$$\pi'(s) \neq \pi(s) \qquad Q^{\pi}\left(s,\pi'(s)\right) > Q^{\pi}\left(s,\pi(s)\right) = V^{\pi}(s)$$

  2. 在其他所有状态s'下，两策略输出相同，即

$$\pi'(s') = \pi(s') \qquad Q^{\pi}\left(s,\pi'(s)\right) = Q^{\pi}\left(s,\pi(s)\right) = V^{\pi}(s)$$

那么π'是π的一种策略提升

策略提升定理（Policy Improvement Theorem）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有

$$Q^{\pi}\left(s,\pi'(s)\right) \geq V^{\pi}(s)$$

以π来记回报

- 进而，π和π'满足：对任何状态s，有

$$V^{\pi'}(s) \geq V^{\pi}(s)$$

以π'来记回报

也即是π'的策略价值（期望回报）超过π，π'比π更加优秀。

策略提升定理（Policy Improvement Theorem）

- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有$Q^{\pi}\big(s,\pi'(s)\big) \geq V^{\pi}(s)$，因此有$V^{\pi'}(s) \geq V^{\pi}(s)$
- 证明：

$$
\begin{aligned}
V^{\pi}(s) &\leq Q^{\pi}(s,\pi'(s)) \\
&= \mathbb{E}_{\text{Env}}[R_t + \gamma V^{\pi}(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma V^{\pi}(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma Q^{\pi}(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma \mathbb{E}_{\text{Env}}[R_{t+1} + \gamma V^{\pi}(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 V^{\pi}(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 V^{\pi}(S_{t+3}) | S_t = s] \\
&\cdots \\
&\leq \mathbb{E}_{\pi'}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots | S_t = s] \\
&= V^{\pi}(s)
\end{aligned}
$$

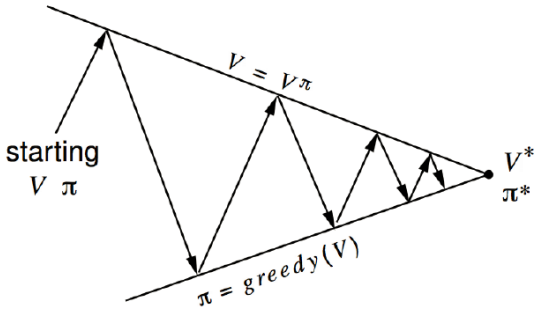
策略提升定理（Policy Improvement Theorem）

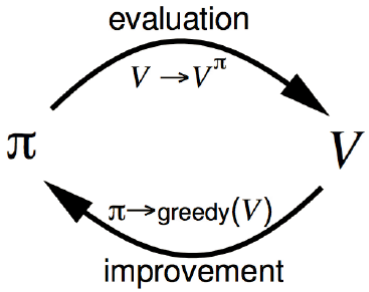
- 对于两个策略π，π'，如果满足如下性质，π'是π的策略提升：
  - 对于任何状态s，有$Q^{\pi}\left(s,\pi'(s)\right) \geq V^{\pi}(s)$
  - 因此有$V^{\pi'}(s) \geq V^{\pi}(s)$

- 策略提升定理带给我们的启示

[找到(s,a)使得 $Q^{\pi}(s,a) \geq V^{\pi}(s)$]

策略 π → 评估策略价值 $V^{\pi}(\cdot)$ → 寻找到新策略π'使得 $Q^{\pi}\left(s,\pi'(s)\right) \geq V^{\pi}(s)$





价值评估指导
策略提升

ε-Greedy 策略提升定理

- 对于m个动作的ε-Greedy策略π

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \arg\max_{a \in A} Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

- 如果另一个ε-Greedy策略π'是基于Q^π的提升，那么有$V^{\pi'}(s) \geq V^\pi(s)$

如前面多步推导得出

$$V^{\pi'}(s) \geq Q^\pi(s, \pi'(s)) = \sum_{a \in A} \pi'(a|s) Q^\pi(s,a)$$

m actions

$$= \frac{\epsilon}{m} \sum_{a \in A} Q^\pi(s,a) + (1-\epsilon) \max_{a \in A} Q^\pi(s,a)$$

$$\geq \frac{\epsilon}{m} \sum_{a \in A} Q^\pi(s,a) + (1-\epsilon) \sum_{a \in A} \frac{\pi(a|s) - \epsilon/m}{1-\epsilon} Q^\pi(s,a)$$

$$= \sum_{a \in A} \pi(a|s) Q^\pi(s,a) = V^\pi(s)$$

注意：
1. ε不能大
2. π并不是ε-Greedy(Q^π)，π'才是

- 延升思考：对于随机策略，将策略的动作选择分布移向价值更高的动作

基于动态规划的强化学习

讲师：张伟楠 – 上海交通大学

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值 $V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$

$$= \mathbb{E}_{a \sim \pi(s)} \left[r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s') \right]$$

Bellman等式

立即奖励　状态转移

时间折扣　下一个状态的价值

策略值函数估计（Policy Evaluation）

- 给定环境MDP和策略π，策略值函数估计如下

状态价值 $V^{\pi}(s) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, \pi]$

$$= \mathbb{E}_{a \sim \pi(s)} \left[r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s') \right]$$ Bellman等式

$$= \mathbb{E}_{a \sim \pi(s)} [Q^{\pi}(s,a)]$$

动作价值 $Q^{\pi}(s,a) = \mathbb{E}[r(S_0, A_0) + \gamma r(S_1, A_1) + \gamma^2 r(S_2, A_2) + \cdots | S_0 = s, A_0 = a, \pi]$

$$= r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

$$= r(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') \sum_{a' \in A} \pi(a'|s') \, Q^{\pi}(s', a')$$ Bellman等式

智能体目标和策略

- 目标：选择能够最大化累积奖励期望的动作

$$\mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots]$$

  - $\gamma \in [0,1]$是未来奖励的折扣因子，使得和未来奖励相比起来智能体更重视即时奖励
    - 以金融为例，今天的$1比明天的$1更有价值

- 给定一个特定的策略 $\pi(s): S \to A$，即在状态 s 下采取动作 $a = \pi(s)$

- 给策略π定义价值函数

状态价值 $V^{\pi}(s) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$

动作价值 $Q^{\pi}(s,a) = \mathbb{E}[R(s_0,a_0) + \gamma R(s_1,a_1) + \gamma^2 R(s_2,a_2) + \cdots | s_0 = s, a_0 = a, \pi]$

$$= R(s,a) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

寻找优化策略的方法：策略迭代和价值迭代

- 价值函数和策略相关

$$V^{\pi}(s) = r(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^{\pi}(s')$$

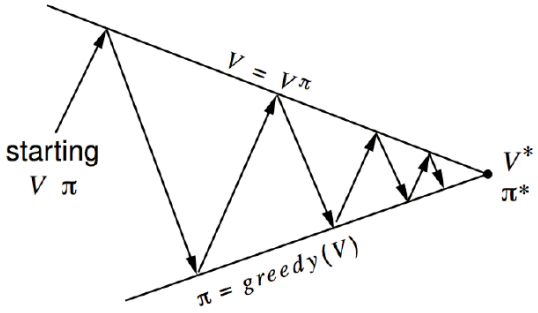
- 可以对最优价值函数和最优策略执行迭代更新
  - 策略迭代
  - 价值迭代

策略迭代

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 策略迭代过程（基于V价值函数）



1. 随机初始化策略 π
2. 重复以下过程直到收敛{
   a) 计算 $V := V^{\pi}$
   b) 对每个状态，更新

$$\pi(s) = \arg\max_{a \in A} r(s,a) + \gamma \sum_{s' \in S} P_{sa}(s') V(s')$$

}

更新价值函数会很耗时

策略迭代

- 对于一个动作空间和状态空间有限的MDP

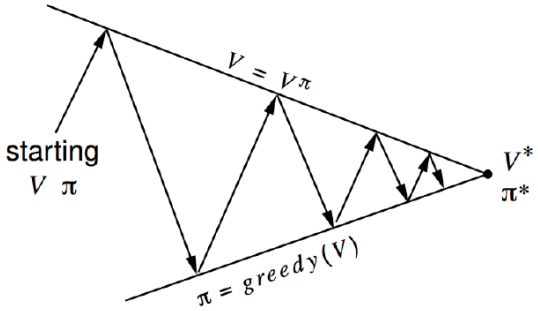
$$|S| < \infty, |A| < \infty$$

- 策略迭代过程（基于Q价值函数）

1. 随机初始化策略 π
2. 重复以下过程直到收敛{
   a) 计算 $Q := Q^{\pi}$
   b) 对每个状态，更新

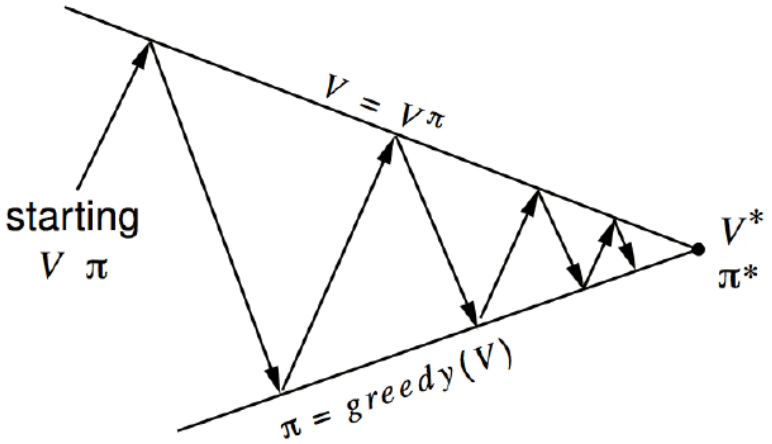
$$\pi(s) = \arg\max_{a \in A} Q(s, a)$$

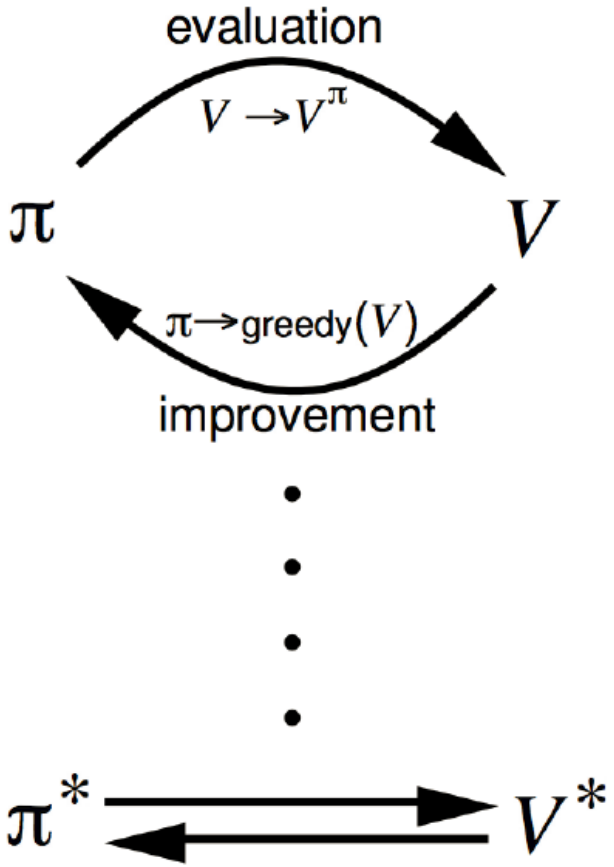
}



更新价值函数会很耗时

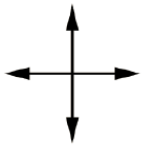
策略迭代





- 策略评估
  - 估计V^{π}
  - 迭代的评估策略
- 策略改进
  - 生成 $\pi' \geq \pi$
  - 贪心策略改进（后续进一步讨论）

举例：策略评估



动作

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

- 非折扣MDP（$\gamma = 1$）
- 非终止状态：1, 2, ...,14
- 两个终止状态（灰色方格）
- 如果动作指向所有方格以外，则这一步不动
- 奖励均为-1，直到到达终止状态
- 智能体的策略为均匀随机策略

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

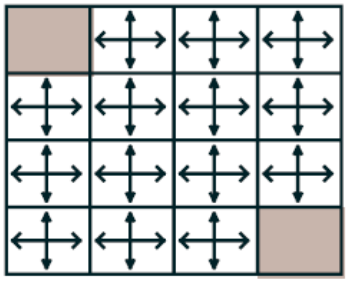
举例：策略评估

随机策略的 V_k

V_k对应的贪心策略

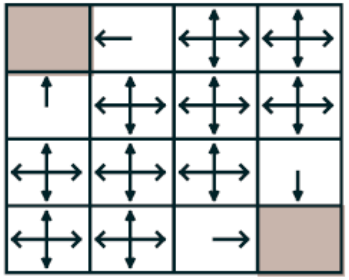
K=0

0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



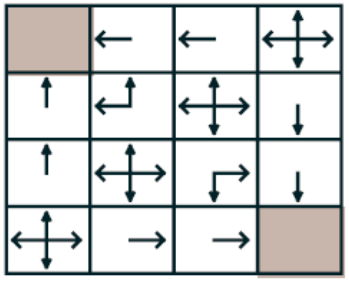
K=1

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



K=2

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



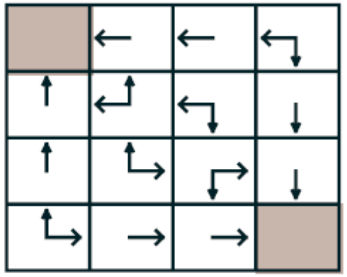
举例：策略评估

随机策略的 V_k

V_k对应的贪心策略

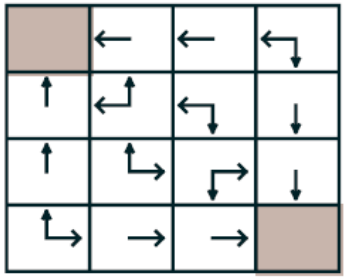
K=3

0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0



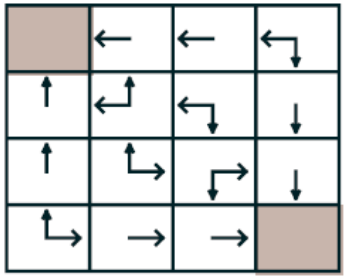
K=10

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0



K=∞

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0

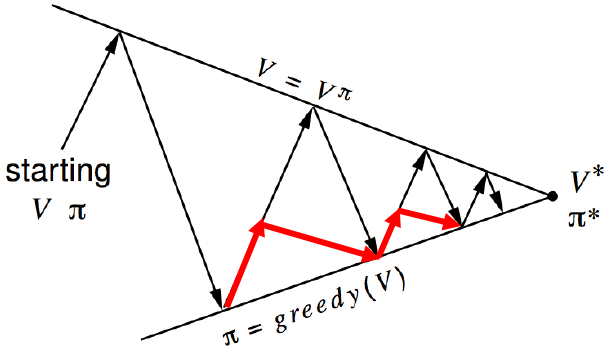


$V := V^{\pi}$
最优策略

思考：如何加速策略迭代方法？

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$



- 策略迭代过程（基于V价值函数）

1. 随机初始化策略 π
2. 重复以下过程直到收敛{
   a) 计算 $V := V^{\pi}$
   b) 对每个状态，更新

$$\pi(s) = \arg\max_{a \in A} r(s,a) + \gamma \sum_{s' \in S} P_{sa}(s')V(s')$$

}

更新价值函数会很耗时，但前面的例子中，迭代计算 V 价值函数为收敛时，其导出的策略已经是最优

价值迭代

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 价值迭代过程

1. 对每个状态s，初始化 $V(s) = 0$
2. 重复以下过程直到收敛 {

对每个状态，更新

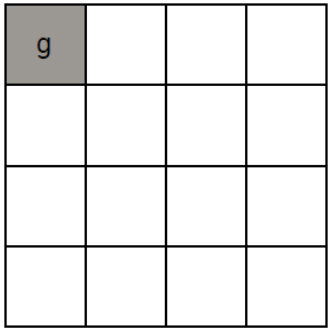
$$V(s) = r(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V(s')$$

}

注意：在以上的计算中没有明确的策略

价值迭代收敛性证明：https://towardsdatascience.com/mathematical-analysis-of-reinforcement-learning-bellman-equation-ac9f0954e19f

价值迭代例子：最短路径



Problem

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

V_1

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

V_2

0	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

V_3

0	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

V_4

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

V_5

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

V_6

0	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

V_7

最优价值函数

- 对状态s来说的最优价值函数是所有策略可获得的最大可能折扣奖励的和

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- 最优价值函数的Bellman等式（Bellman optimality equation）

$$V^*(s) = r(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s') V^*(s')$$

- 最优策略

$$\pi^*(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s') V^*(s')$$

- 对状态s和策略π

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s)$$

价值迭代 vs. 策略迭代

价值迭代

1. 对每个状态s，初始化 $V(s) = 0$
2. 重复以下过程直到收敛 {
   对每个状态，更新
   $$V(s) = r(s) + \max_{a \in A} \gamma \sum_{s' \in S} P_{sa}(s')V(s')$$
   }

策略迭代

1. 随机初始化策略 π
2. 重复以下过程直到收敛 {
   a) 计算价值 $V := V^{\pi}$
   b) 对每个状态，更新
   $$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s')V(s')$$
   }

备注：

1. 价值迭代是贪心更新法
   - 相当于策略评估中进行一轮价值更新，然后直接根据更新后的价值进行策略提升
2. 策略迭代中，用Bellman等式更新价值函数代价很大
3. 对于空间较小的MDP，策略迭代通常很快收敛
4. 对于空间较大的MDP，价值迭代更实用（效率更高）
5. 如果没有状态转移循环，最好使用价值迭代

课程大纲

强化学习基础

1. 强化学习技术概览
2. 马尔可夫决策过程
3. 动态规划
4. **值函数估计**
5. 无模型控制方法
6. 参数化值函数
7. 策略梯度
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

无模型的强化学习（Model-Free RL）

- 在现实问题中，通常没有明确地给出状态转移和奖励函数
  - 例如，我们仅能观察到部分片段（episodes）

Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \dots s_T^{(1)}$

Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \dots s_T^{(2)}$

- 无模型的强化学习直接从经验中学习值（value）和策略（policy），而无需构建马尔可夫决策过程模型（MDP）
- 关键步骤：（1）估计值函数；（2）优化策略

值函数估计

- 在基于模型的强化学习（MDP）中，值函数能够通过动态规划计算获得

$$V^{\pi}(s) = \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$$
$$= R(s) + \gamma \sum_{s' \in S} P_{s\pi(s)}(s') V^{\pi}(s')$$

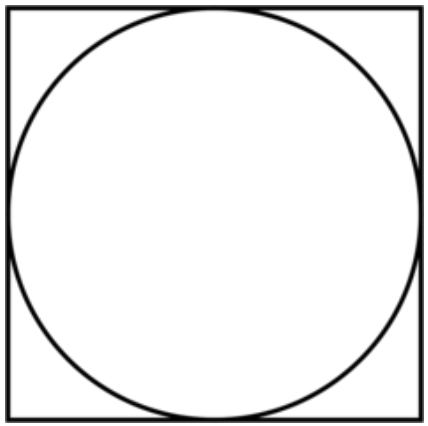
- 在无模型的强化学习中
  - 我们无法直接获得 P_{sa} 和 R
  - 但是，我们拥有一系列可以用来估计值函数的经验

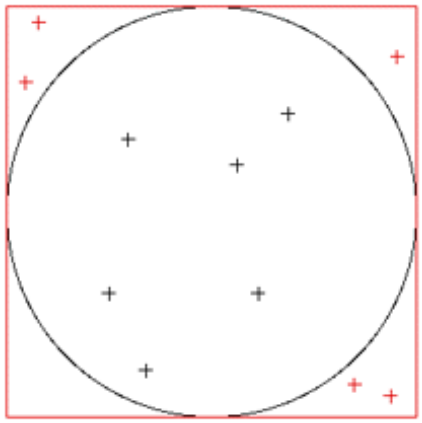
Episode 1: $s_0^{(1)} \xrightarrow[R(s_0)^{(1)}]{a_0^{(1)}} s_1^{(1)} \xrightarrow[R(s_1)^{(1)}]{a_1^{(1)}} s_2^{(1)} \xrightarrow[R(s_2)^{(1)}]{a_2^{(1)}} s_3^{(1)} \ldots s_T^{(1)}$

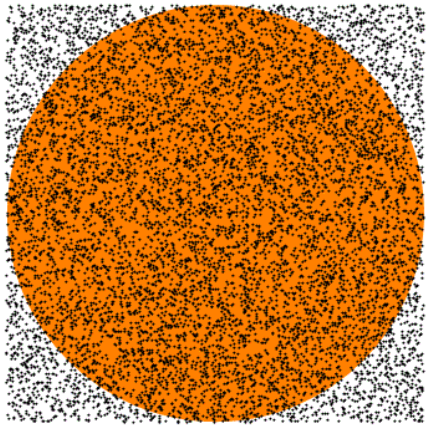
Episode 2: $s_0^{(2)} \xrightarrow[R(s_0)^{(2)}]{a_0^{(2)}} s_1^{(2)} \xrightarrow[R(s_1)^{(2)}]{a_1^{(2)}} s_2^{(2)} \xrightarrow[R(s_2)^{(2)}]{a_2^{(2)}} s_3^{(2)} \ldots s_T^{(2)}$

蒙特卡洛方法

- 蒙特卡洛方法（Monte-Carlo methods）是一类广泛的计算算法。生活中处处都是MC方法。
  - 依赖于重复随机抽样来获得数值结果
- 例如，计算圆的面积



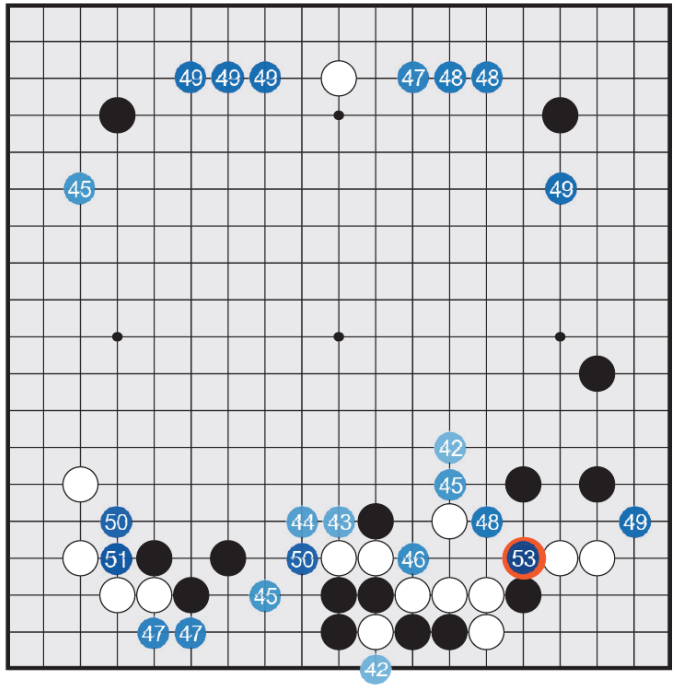


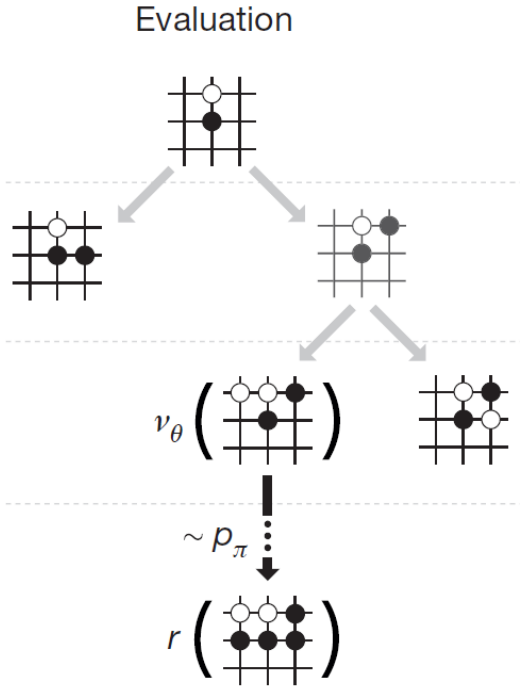


$$\text{Circle Surface} = \text{Square Surface} \times \frac{\#\text{points in circle}}{\#\text{points in total}}$$

蒙特卡洛方法

- 围棋对弈：估计当前状态下的胜率





$$\text{Win Rate}(s) = \frac{\#\text{win simulation cases started from } s}{\#\text{simulation cases started from } s \text{ in total}}$$

蒙特卡洛价值估计

- 目标：从策略 π 下的经验片段学习 V^{π}

$$s_0^{(i)} \xrightarrow[R_1^{(i)}]{a_0^{(i)}} s_1^{(i)} \xrightarrow[R_2^{(i)}]{a_1^{(i)}} s_2^{(i)} \xrightarrow[R_3^{(i)}]{a_2^{(i)}} s_3^{(i)} \ldots s_T^{(i)} \sim \pi$$

- 回顾：累计奖励（return）是总折扣奖励

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots \gamma^{T-1} R_T$$

- 回顾：值函数（value function）是期望累计奖励

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}[G_t | s_t = s, \pi] \\ &\simeq \frac{1}{N} \sum_{i=1}^{N} G_t^{(i)} \end{aligned}$$

- 使用策略π从状态s采样N个片段
- 计算平均累计奖励

- 蒙特卡洛策略评估使用经验均值累计奖励而不是期望累计奖励

蒙特卡洛价值估计

实现

- 使用策略π采样片段

$$s_0^{(i)} \xrightarrow[R_1^{(i)}]{a_0^{(i)}} s_1^{(i)} \xrightarrow[R_2^{(i)}]{a_1^{(i)}} s_2^{(i)} \xrightarrow[R_3^{(i)}]{a_2^{(i)}} s_3^{(i)} \dots s_T^{(i)} \sim \pi$$

- 在一个片段中的每个时间步长t的状态s都被访问
  - 增量计数器 $N(s) \leftarrow N(s) + 1$
  - 增量总累计奖励 $S(s) \leftarrow S(s) + G_t$
  - 价值被估计为累计奖励的均值 $V(s) = S(s)/N(s)$
  - 由大数定率有

$$V(s) \to V^{\pi}(s) \text{ as } N(s) \to \infty$$

增量蒙特卡洛更新

- 每个片段结束后逐步更新$V(s)$
- 对于每个状态S_t和对应累计奖励G_t

$$N(S_t) \leftarrow N(S_t) + 1$$
$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)}\big(G_t - V(S_t)\big)$$

- 对于非稳定的问题（即，环境会随时间发生变化），我们可以跟踪一个现阶段的平均值（即，不考虑过久之前的片段）

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

蒙特卡洛值估计

思路： $V(S_t) \simeq \frac{1}{N}\sum_{i=1}^{N} G_t^{(i)}$

实现： $V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$

- 蒙特卡洛方法：直接从经验片段进行学习
- 蒙特卡洛是无模型的：未知马尔可夫决策过程的状态转移/奖励
- 蒙特卡洛从完整的片段中进行学习：没有使用bootstrapping的方法
- 蒙特卡洛采用最简单的思想：值（value）= 平均累计奖励（mean return）
- 注意：只能将蒙特卡洛方法应用于有限长度的马尔可夫决策过程中
  - 即，所有的片段都有终止状态

时序差分学习

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = R_{t+1} + \gamma V(S_{t+1})$$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

观测值　　对未来的猜测

- 时序差分方法直接从经验片段中进行学习
- 时序差分方法属于无模型的强化学习范畴
  - 不需要预先获取马尔可夫决策过程的状态转移/奖励
- 通过bootstrapping，时序差分从不完整的片段中学习
- 时序差分更新当前预测值使之接近估计累计奖励（非真实值）

蒙特卡洛 vs. 时序差分（MC vs. TD)

相同的目标：从策略π下的经验片段学习V^{π}

- 增量地进行每次蒙特卡洛过程（MC）
  - 更新值函数$V(S_t)$使之接近准确累计奖励G_t

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

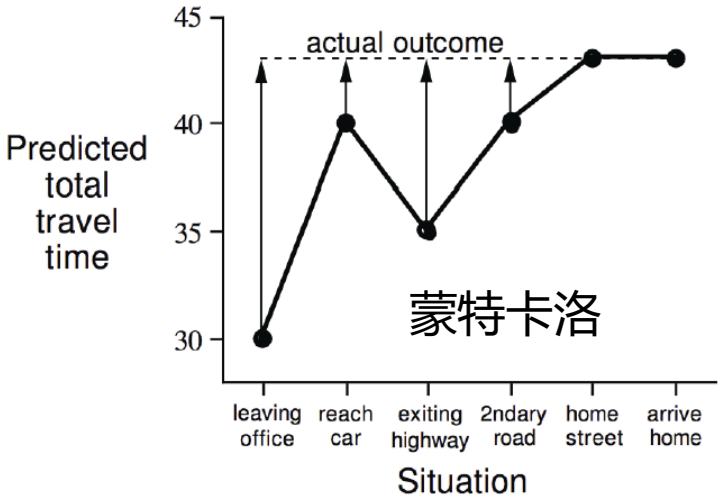
- 最简单的时序差分学习算法（TD）：
  - 更新$V(S_t)$使之接近估计累计奖励$R_{t+1} + \gamma V(S_{t+1})$

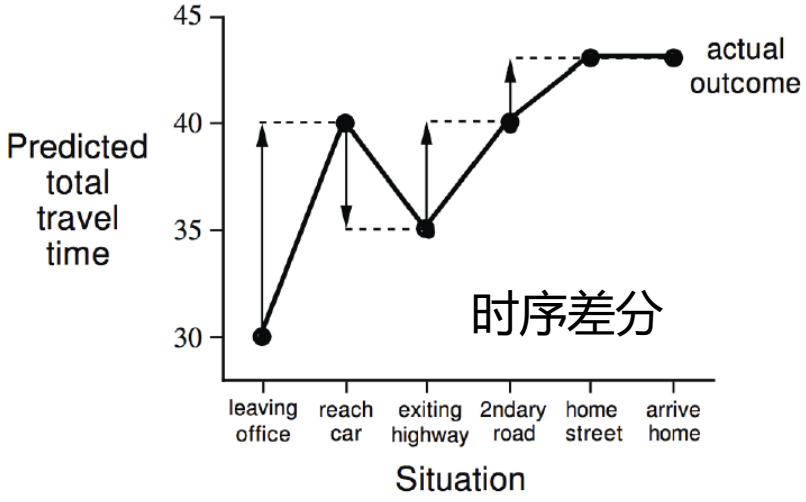
$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

  - 时序差分目标：$R_{t+1} + \gamma V(S_{t+1})$
  - 时序差分误差：$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

驾车回家的例子

状态	经过的时间(分钟)	预计所剩时间	预计总时间
离开公司	0	30	30
开始驾车，下雨	5	35	40
离开高速公路	20	15	35
卡车后跟车	30	10	40
到达家所在街道	40	3	43
直奔家门	43	0	43





蒙特卡洛（MC）和时序差分（TD）的优缺点

- 时序差分：能够在知道最后结果之前进行学习
  - 时序差分能够在每一步之后进行在线学习
  - 蒙特卡洛必须等待片段结束，直到累计奖励已知

- 时序差分：能够无需最后结果地进行学习
  - 时序差分能够从不完整的序列中学习
  - 蒙特卡洛只能从完整序列中学习
  - 时序差分在连续（无终止的）环境下工作
  - 蒙特卡洛只能在片段化的（有终止的）环境下工作

偏差（Bias）/方差（Variance）的权衡

- 累计奖励$G_t = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-1} R_T$是$V^{\pi}(S_t)$的无偏估计
- 时序差分真实目标$R_{t+1} + \gamma V^{\pi}(S_{t+1})$是$V^{\pi}(S_t)$的无偏估计
- 时序差分目标$R_{t+1} + \gamma \underbrace{V(S_{t+1})}_{\text{当前估计}}$是$V^{\pi}(S_t)$的有偏估计
- 时序差分目标具有比累计奖励更低的方差
  - 累计奖励——取决于多步随机动作，多步状态转移和多步奖励
  - 时序差分目标——取决于单步随机动作，单步状态转移和单步奖励

蒙特卡洛（MC）和时序差分（TD）的优缺点（2）

MC:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

蒙特卡洛具有高方差，无偏差

- 良好的收敛性质
  - 使用函数近似时依然如此
- 对初始值不敏感
- 易于理解和使用

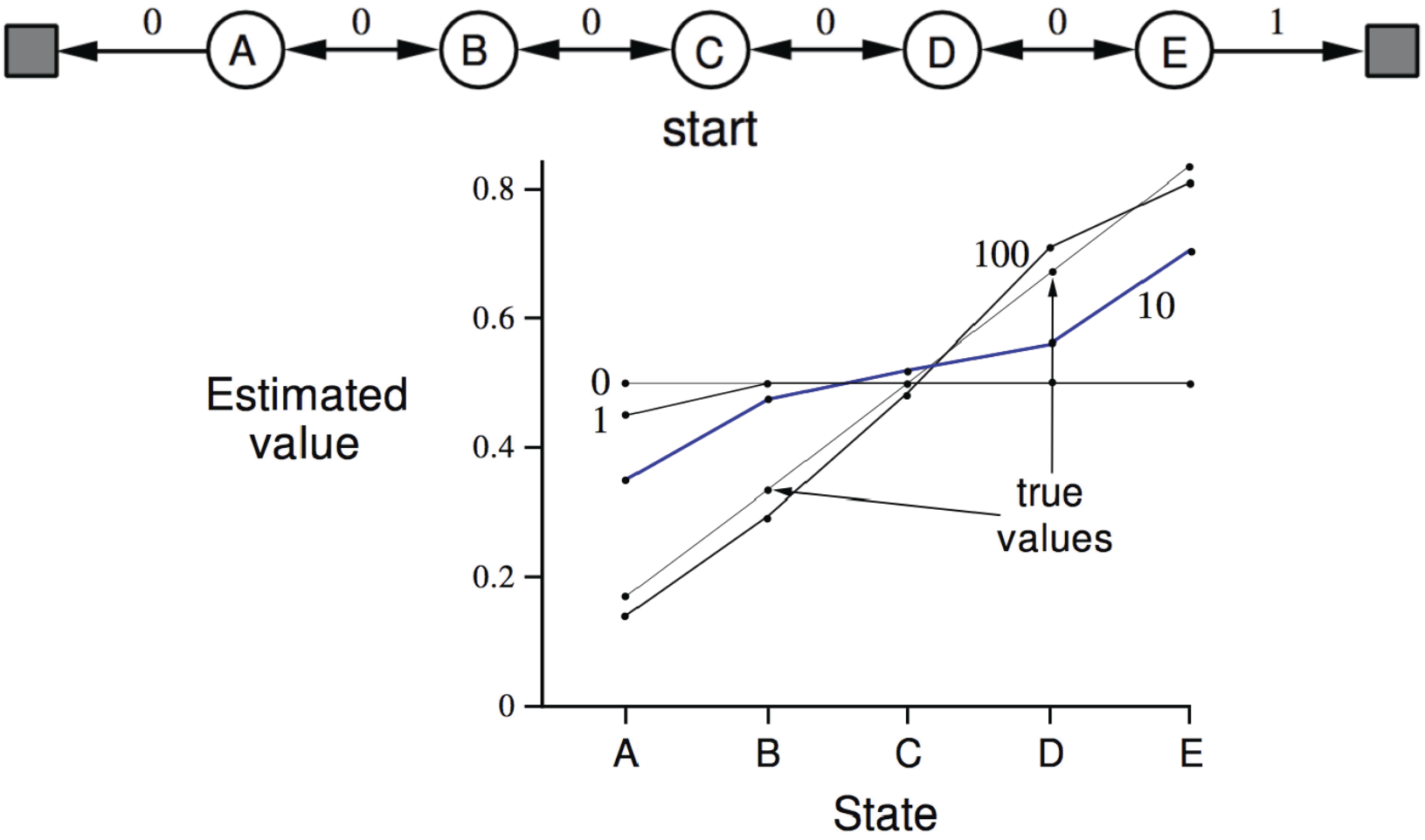
TD:

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

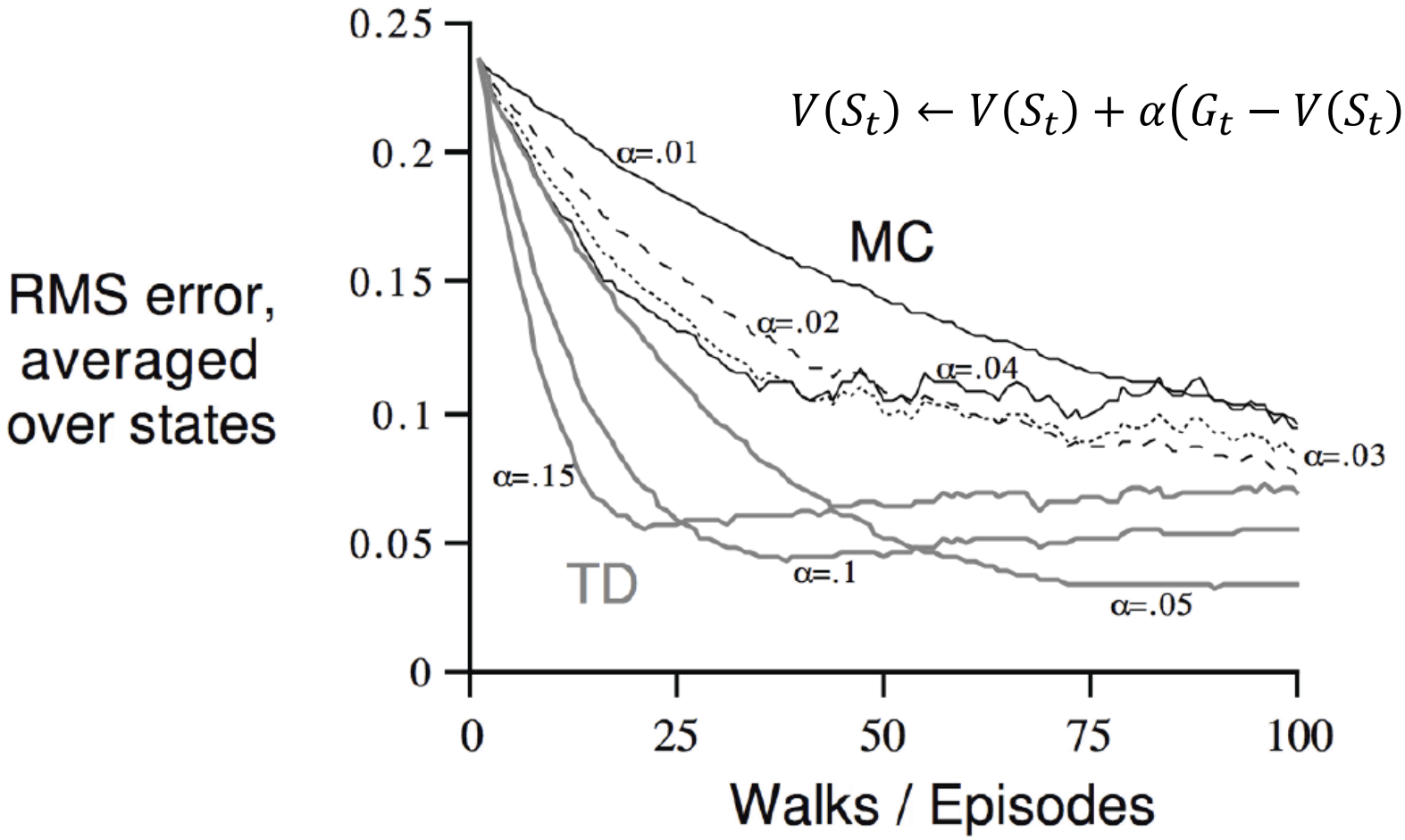
时序差分具有低方差，有偏差

- 通常比蒙特卡洛更加高效
- 时序差分最终收敛到$V^{\pi}(S_t)$
  - 但使用函数近似并不总是如此
- 比蒙特卡洛对初始值更加敏感

随机游走的例子


0
A
0
B
0
C
0
D
0
E
1
start
Estimated value
0.8
0.6
0.4
0.2
0
100
10
0
1
true values
A
B
C
D
E
State

随机游走的例子

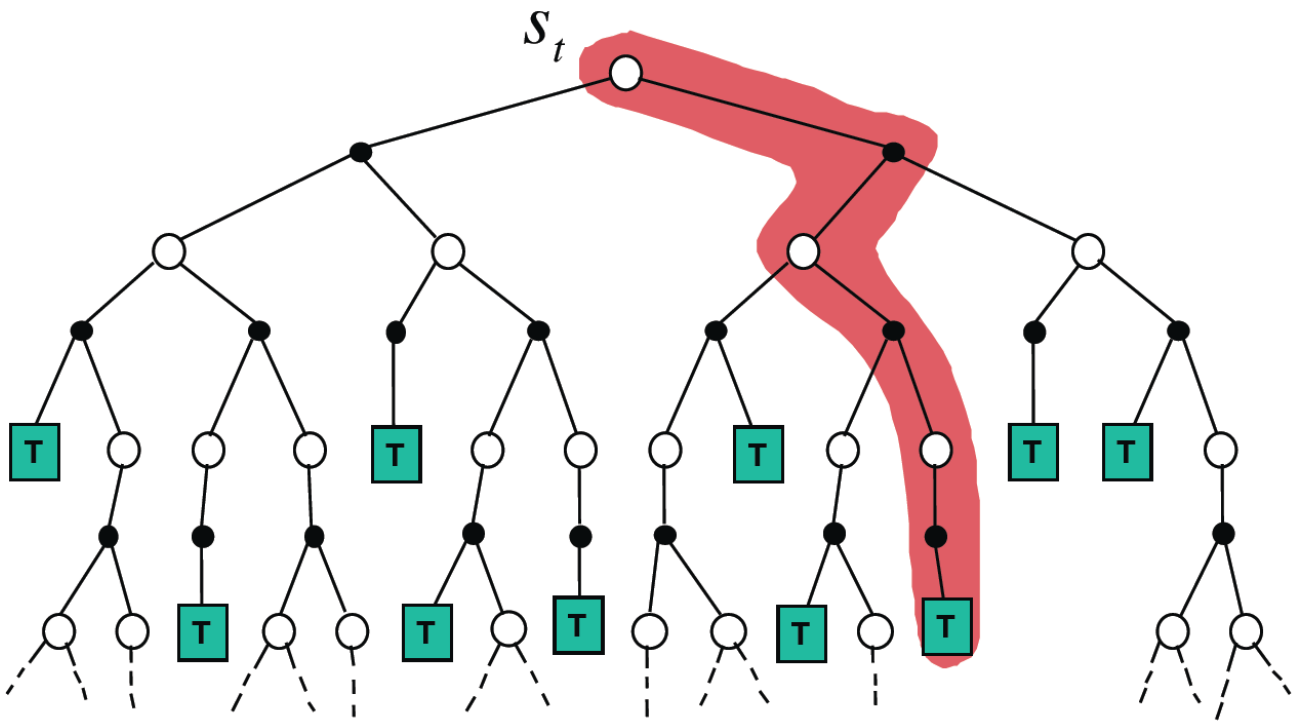


$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

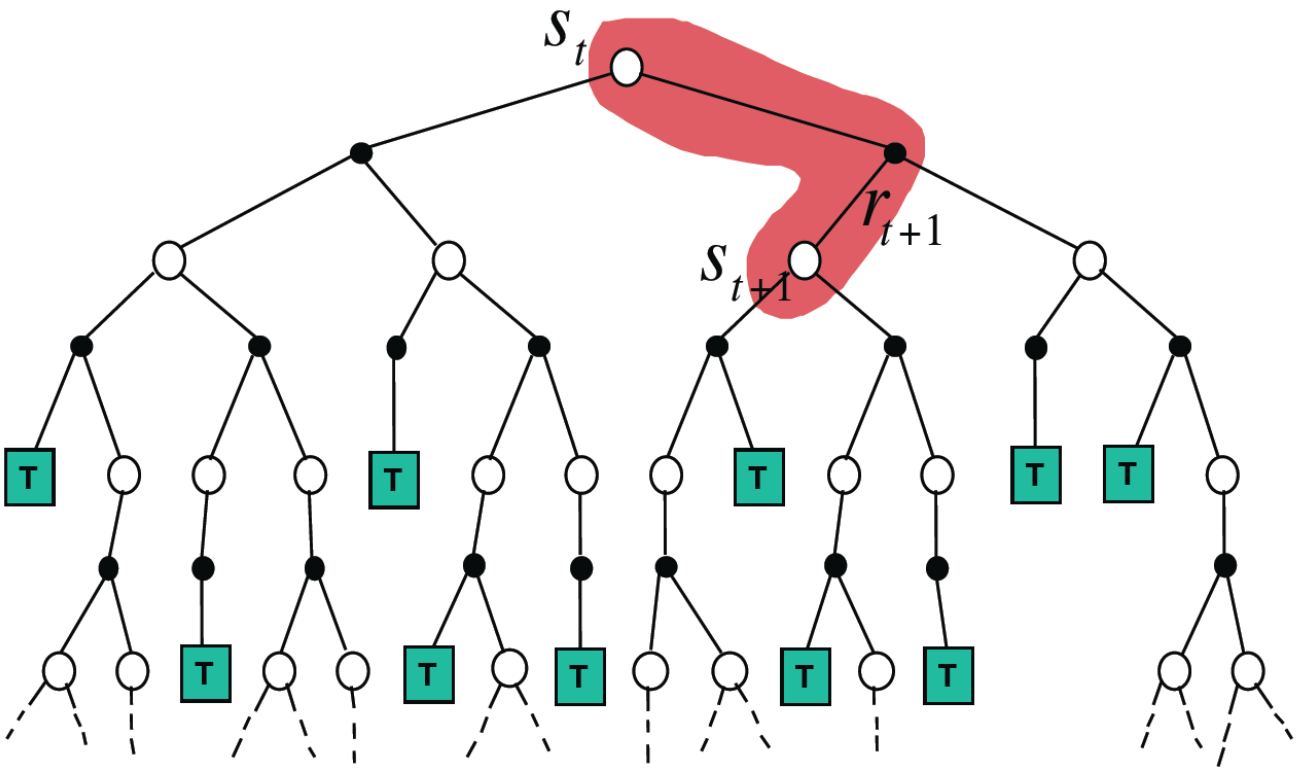
蒙特卡洛反向传播（Backup）

$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$



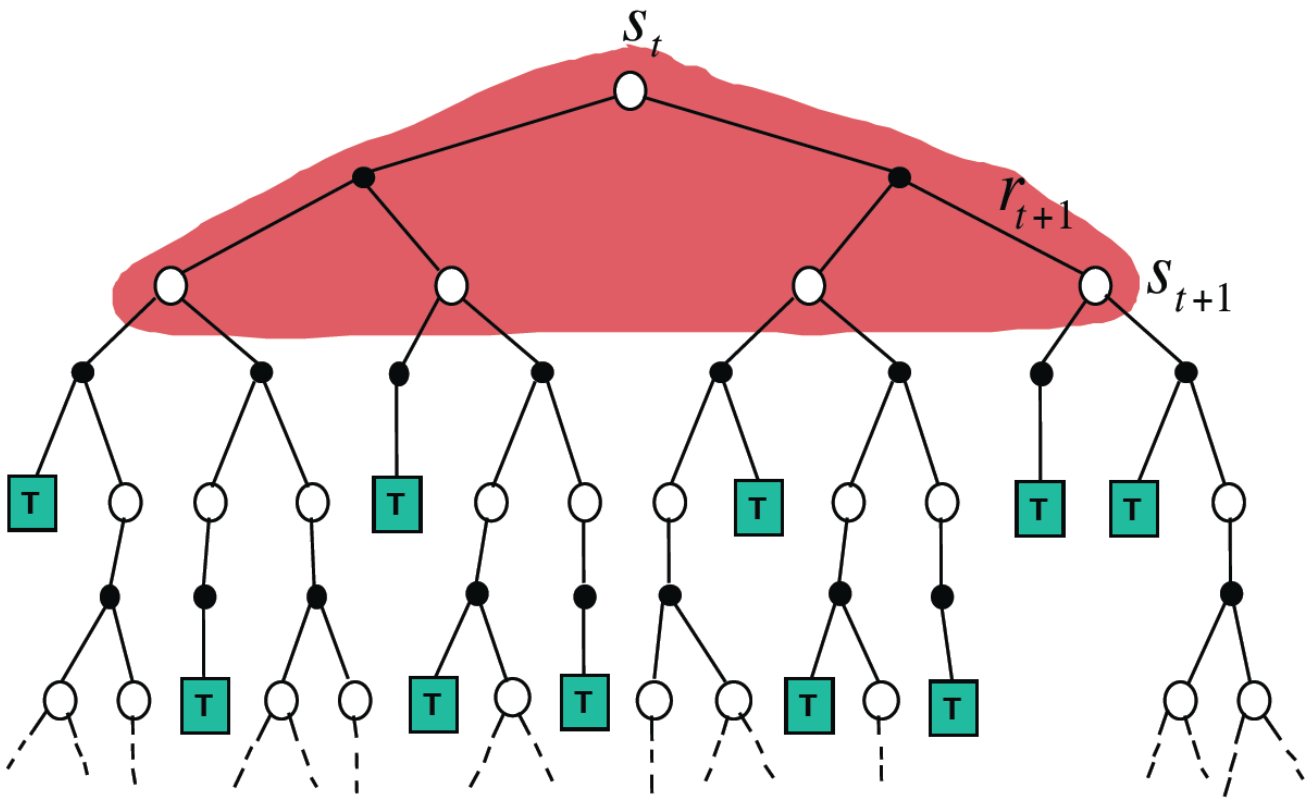
时序差分反向传播（Backup）

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$



动态规划反向传播（Backup）

$$V(S_t) \leftarrow \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})]$$



课程大纲

强化学习基础

1. 强化学习技术概览
2. 马尔可夫决策过程
3. 动态规划
4. 值函数估计
5. **无模型控制方法**
6. 参数化值函数
7. 策略梯度
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

从知道什么是好的，到如何做好行动

- 从知道什么是好的：估计$V^{\pi}(S_t)$

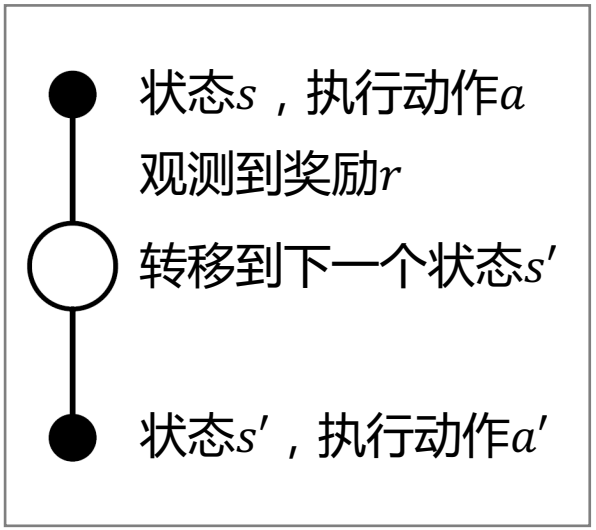
$$V(S_t) \leftarrow V(S_t) + \alpha\big(G_t - V(S_t)\big)$$

$$V(S_t) \leftarrow V(S_t) + \alpha\big(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\big)$$

- 基于V函数，如何选择好的行动？

$$\pi(s) = \arg\max_{a \in A} \sum_{s' \in S} P_{sa}(s')V(s')$$

需要知道环境模型



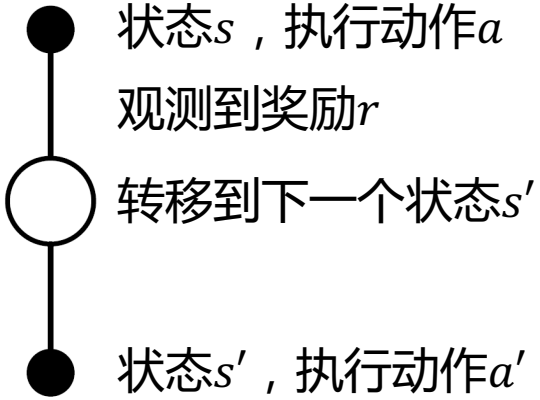
- 基于Q函数，如何选择好的行动？

$$\pi(s) = \arg\max_{a \in A} Q(s, a)$$

  - 因此，估计Q函数对直接做行动（控制）有直接的作用

SARSA

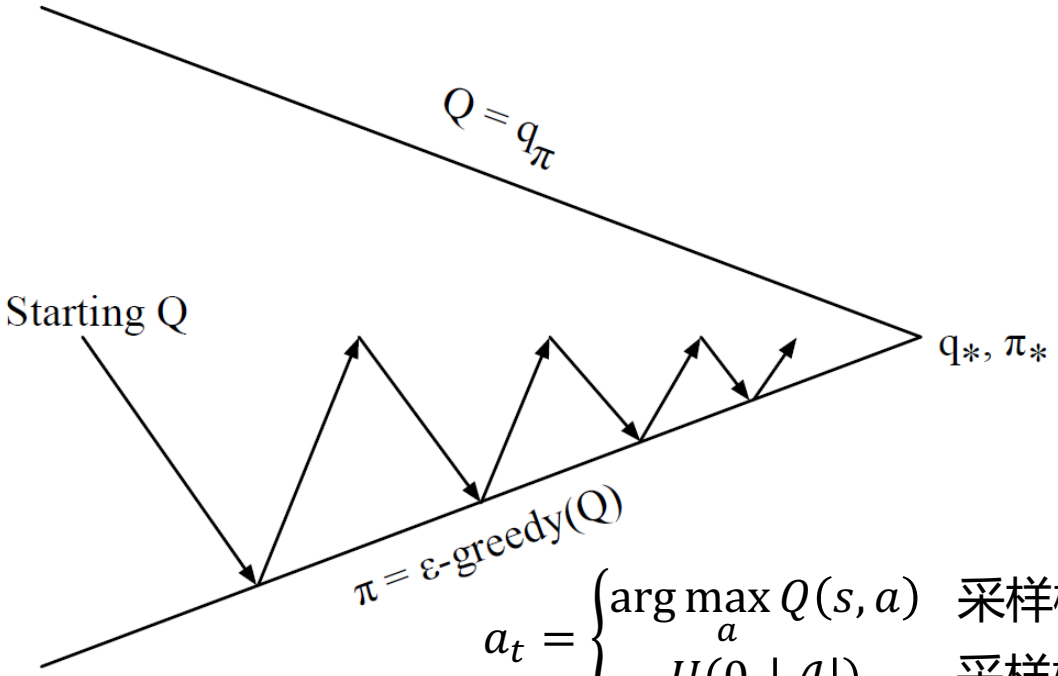
- 对于当前策略执行的每个（状态-动作-奖励-状态-动作）元组



- SARSA更新状态-动作值函数为

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

使用SARSA的同策略控制



$$a_t = \begin{cases} \arg\max_a Q(s,a) & \text{采样概率：} 1-\epsilon \\ U(0, |\mathcal{A}|) & \text{采样概率：} \epsilon \end{cases}$$

- 每个时间步长：
  - 策略评估：SARSA $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$
  - 策略改进：ε-greedy策略改进

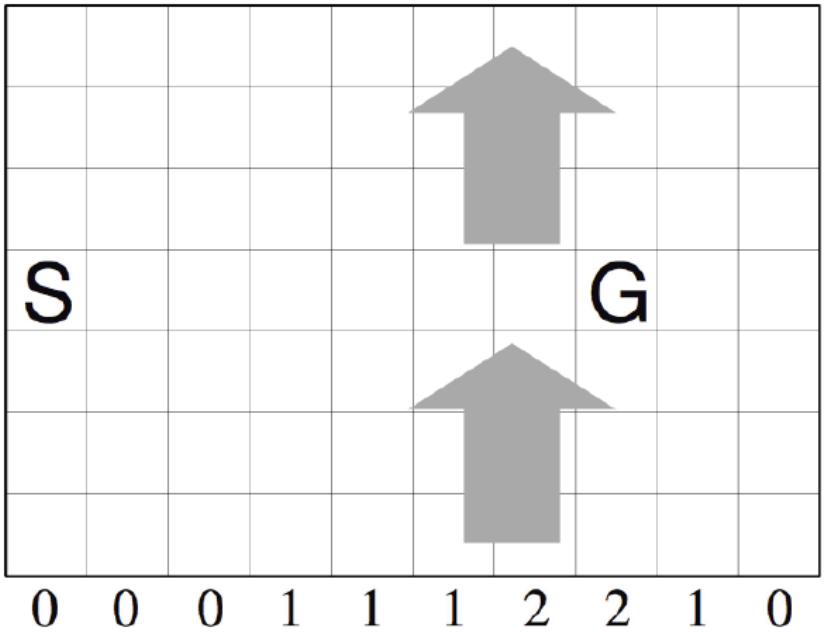
SARSA算法

Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

注：**同策略**时序差分控制（on-policy TD control）使用当前策略进行动作采样。即，SARSA算法中的两个“A”都是由当前策略选择的

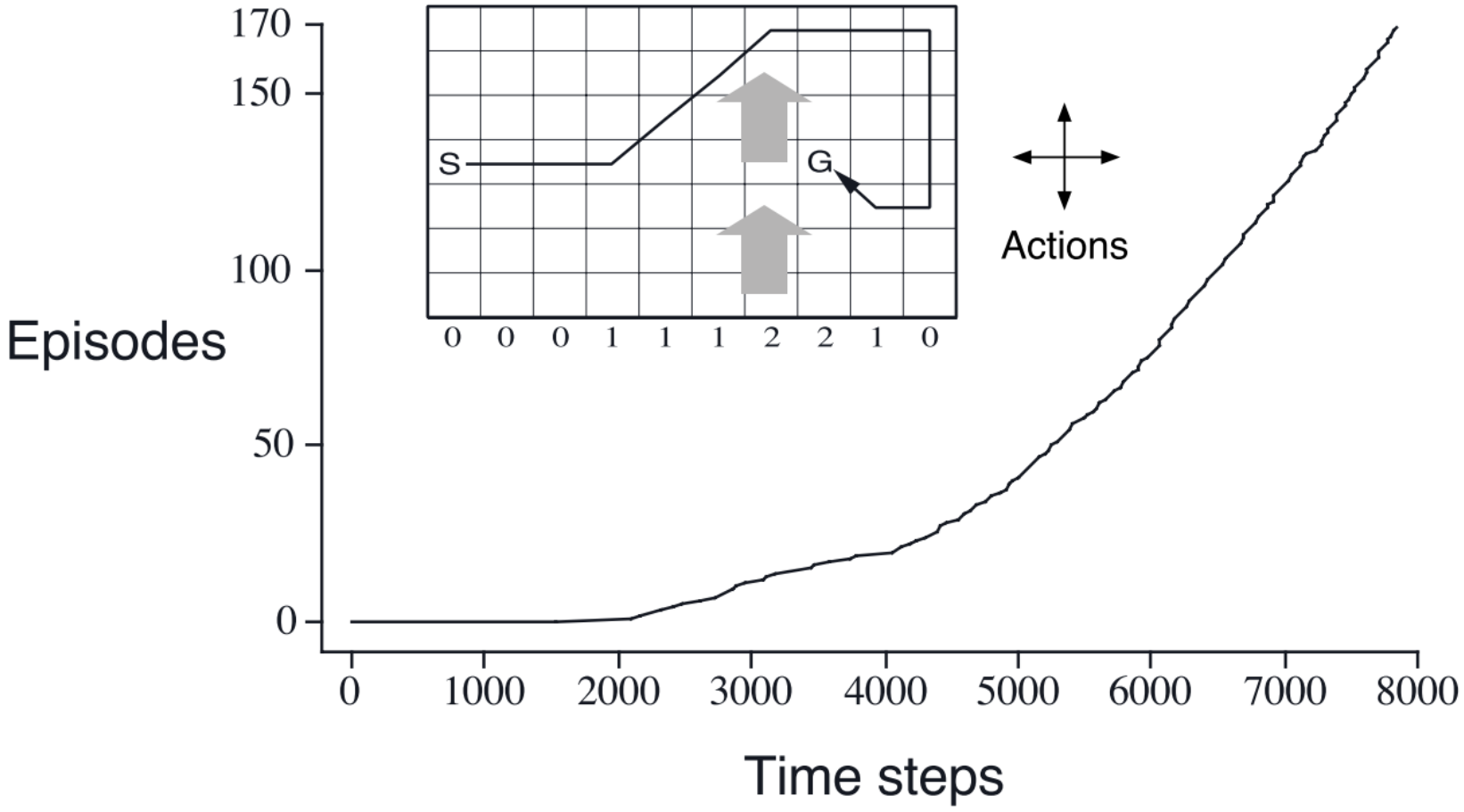
SARSA示例：Windy Gridworld



standard moves

- 每步的奖励 = -1，直到智能体抵达目标网格
- 无折扣因子

SARSA示例：Windy Gridworld



注意：随着训练的进行，SARSA策略越来越快速地抵达目标

Q-learning

- 学习状态-动作值函数 $Q(s,a) \in \mathbb{R}$，不直接优化策略
- 一种异策略（off-policy）学习方法

策略函数，一般是给定的策略，$\mu(\cdot|\, s_t,) \in \mathbb{R}^{|A|}$

动作空间，$a \sim A$

$$Q(s_t,a_t) = \sum_{t=0}^{T} \gamma^t R(s_t,a_t), a_t \sim \mu(s_t)$$

奖励函数，$R(s_t,a_t) \in \mathbb{R}$

迭代式：$Q(s_t,a_t) = R(s_t,a_t) + \gamma Q(s_{t+1},a_{t+1})$

异策略学习

什么是异策略学习

- 目标策略 $\pi(a|s)$ 进行值函数评估（ $V^{\pi}(s)$或$Q^{\pi}(s,a)$ ）
- 行为策略 $\mu(a|s)$ 收集数据： $\{s_1, a_1, r_2, s_2, a_2, \dots, S_T\} \sim \mu$

为什么使用异策略学习

- 平衡探索（ exploration ）和利用（ exploitation ）
- 通过观察人类或其他智能体学习策略
- 重用旧策略所产生的经验
- 遵循探索策略时学习最优策略
- 遵循一个策略时学习多个策略
- 在剑桥MSR研究时的一个例子（ xbox音乐推荐 ）

Q-learning

- 无需重要性采样（为什么？）
- 根据行为策略选择动作 $a_t \sim \mu(\cdot|s_t)$
- 根据目标策略选择后续动作 $a'_{t+1} \sim \pi(\cdot|s_t)$
  - 目标$Q^*(s_t, a_t) = r_t + \gamma Q(s_{t+1}, a'_{t+1})$
- 更新$Q(s_t, a_t)$的值以逼近目标状态-动作值

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t)\right)$$

a'_{t+1}为策略π的动作，而非策略μ

使用Q-learning的异策略控制

- 允许行为策略和目标策略都进行改进
- 目标策略π是关于$Q(s,a)$的贪心策略

$$\pi(s_{t+1}) = \arg\max_{a'} Q(s_{t+1}, a')$$

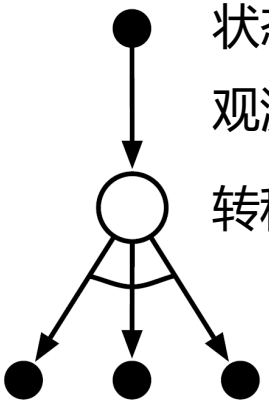
- 行为策略μ是关于$Q(s,a)$的ε-贪心策略
- Q-learning目标函数可以简化为

$$\begin{aligned} r_{t+1} + \gamma Q(s_{t+1}, a'_{t+1}) &= r_{t+1} + \gamma Q(s_{t+1}, \arg\max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1})) \\ &= r_{t+1} + \gamma \max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1}) \end{aligned}$$

- Q-learning更新方式

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'_{t+1}} Q(s_{t+1}, a'_{t+1}) - Q(s_t, a_t))$$

Q-learning控制算法



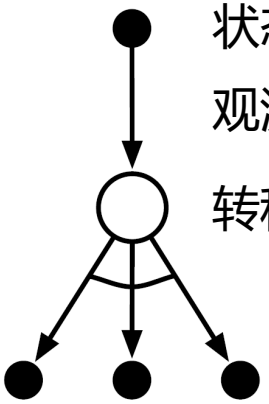
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

- 定理：Q-learning控制收敛到最优状态-动作值函数

$$Q(s, a) \rightarrow Q^*(s, a)$$

具体证明请见：https://hrl.boyuai.com/chapter/1/时序差分算法

Q-learning控制算法



$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t))$$

- 为什么不需要重要性采样？
  - 使用了状态-动作值函数而不是使用状态值函数
  - 没有任何一处使用了历史行为策略的采样过程

课程大纲

强化学习基础

1. **强化学习技术概览**
2. 马尔可夫决策过程
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. **参数化值函数**
7. 策略梯度
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

价值函数近似算法

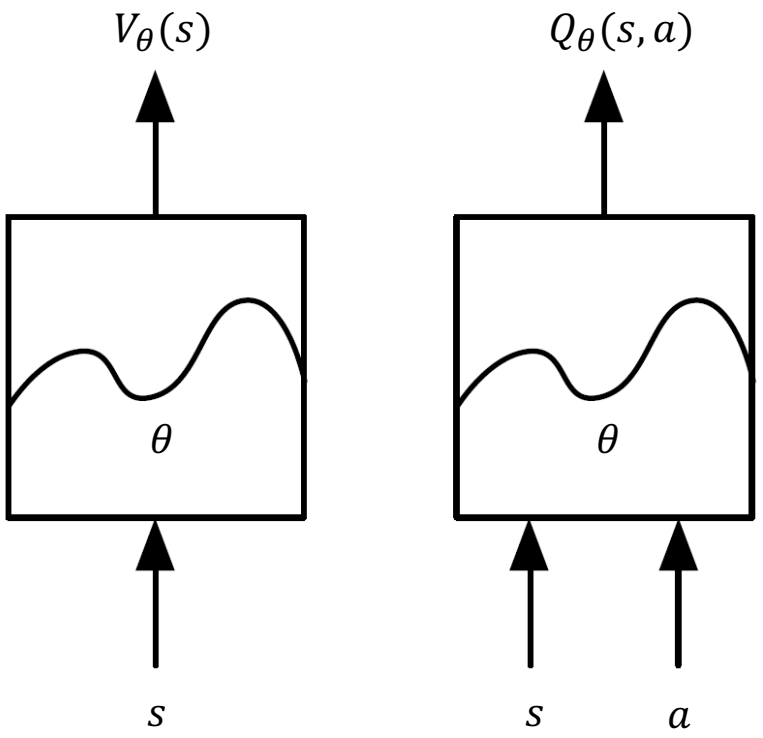
讲师：张伟楠 – 上海交通大学

参数化值函数近似

- 构建参数化（可学习的）函数来近似值函数

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

  - θ是近似函数的参数，可以通过强化学习进行更新
  - 参数化的方法将现有可见的状态泛化到没有见过的状态上



- 一些函数近似
  - （一般的）线性模型
  - 神经网络
  - 决策树
  - 最近邻
  - 傅立叶/小波基底
- 可微函数
  - （一般的）线性模型
  - 神经网络
- 我们希望模型适合在非稳态的，非独立同分布的数据上训练

基于随机梯度下降（SGD）的值函数近似

- 目标：找到参数向量θ最小化值函数近似值与真实值之间的均方误差

$$J(\theta) = \mathbb{E}_\pi[\frac{1}{2}\big(V^\pi(s) - V_\theta(s)\big)^2]$$

- 误差减小的梯度方向

$$-\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_\pi[\big(V^\pi(s) - V_\theta(s)\big)\frac{\partial V_\theta(s)}{\partial \theta}]$$

- 单次采样进行随机梯度下降

$$\begin{aligned}\theta &\leftarrow \theta - \alpha\frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha\big(V^\pi(s) - V_\theta(s)\big)\frac{\partial V_\theta(s)}{\partial \theta}\end{aligned}$$

特征化状态

- 用一个特征向量表示状态

$$x(s) = \begin{bmatrix} x_1(s) \\ \vdots \\ x_k(s) \end{bmatrix}$$

- 以直升机控制问题为例
  - 3D位置
  - 3D速度（位置的变化量）
  - 3D加速度（速度的变化量）

线性状态值函数近似

- 用特征的线性组合表示价值函数

$$V_\theta(s) = \theta^{\mathrm{T}} x(s)$$

- 目标函数是参数θ的二次函数

$$J(\theta) = \mathbb{E}_\pi\left[\frac{1}{2}\left(V^\pi(s) - \theta^{\mathrm{T}} x(s)\right)^2\right]$$

- 因而随机梯度下降能够收敛到全局最优解上

$$\begin{aligned}\theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha\left(V^\pi(s) - V_\theta(s)\right)x(s)\end{aligned}$$

步长　预测误差　特征值

蒙特卡洛状态值函数近似

$$\theta \leftarrow \theta + \alpha\big(V^{\pi}(s) - V_{\theta}(s)\big)x(s)$$

- 我们用$V^{\pi}(s)$表示真实的目标价值函数
- 在“训练数据”上运用监督学习对价值函数进行预测

$$\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \dots, \langle s_T, G_T\rangle$$

- 对于每个数据样本$\langle s_t, G_t\rangle$

$$\theta \leftarrow \theta + \alpha\big(G_t - V_{\theta}(s)\big)x(s_t)$$

- 蒙特卡洛预测至少能收敛到一个局部最优解
  - 在价值函数为线性的情况下可以收敛到全局最优

时序差分状态值函数近似

$$\theta \leftarrow \theta + \alpha\big(V^{\pi}(s) - V_{\theta}(s)\big)x(s)$$

- 时序差分算法的目标$r_{t+1} + \gamma V_{\theta}(s_{t+1})$是真实目标价值$V_{\pi}(s_t)$的有偏采样

- 在“训练数据”上运用监督学习

$$\langle s_1, r_2 + \gamma V_{\theta}(s_2)\rangle, \langle s_2, r_3 + \gamma V_{\theta}(s_3)\rangle, \dots, \langle s_T, r_T\rangle$$

- 对于每个数据样本$\langle s_t, r_{t+1} + \gamma V_{\theta}(s_{t+1})\rangle$

$$\theta \leftarrow \theta + \alpha\big(r_{t+1} + \gamma V_{\theta}(s_{t+1}) - V_{\theta}(s)\big)x(s_t)$$

- 线性情况下时序差分学习（接近）收敛到全局最优解

状态-动作值函数近似

- 对动作-状态值函数进行近似

$$Q_\theta(s,a) \simeq Q^\pi(s,a)$$

- 最小均方误差

$$J(\theta) = \mathbb{E}_\pi\left[\frac{1}{2}(Q^\pi(s,a) - Q_\theta(s,a))^2\right]$$

- 在单个样本上进行随机梯度下降

$$\theta \leftarrow \theta - \alpha\frac{\partial J(\theta)}{\partial\theta}$$

$$= \theta + \alpha\big(Q^\pi(s,a) - Q_\theta(s,a)\big)\frac{\partial Q_\theta(s,a)}{\partial\theta}$$

线性状态-动作值函数近似

- 用特征向量表示状态-动作对

$$x(s,a) = \begin{bmatrix} x_1(s,a) \\ \vdots \\ x_k(s,a) \end{bmatrix}$$

- 线性情况下，参数化后Q函数

$$Q_\theta(s,a) = \theta^{\mathrm{T}} x(s,a)$$

- 利用随机梯度下降更新

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta} \\ &= \theta + \alpha \left(Q^\pi(s,a) - \theta^{\mathrm{T}} x(s,a) \right) x(s,a) \end{aligned}$$

时序差分状态-动作值函数近似

$$\theta \leftarrow \theta + \alpha\left(Q^{\pi}(s,a) - Q_{\theta}(s,a)\right)\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

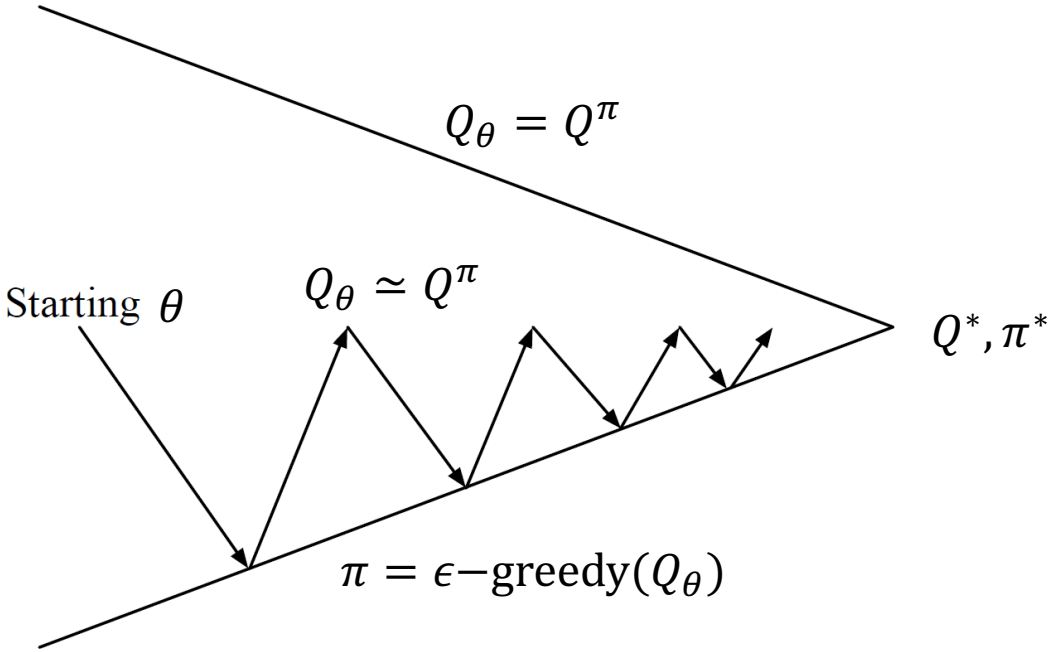
- 对于蒙特卡罗学习，目标是累计奖励G_t

$$\theta \leftarrow \theta + \alpha\left(G_t - Q_{\theta}(s,a)\right)\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

- 对于时序差分学习，目标是$r_{t+1} + \gamma Q_{\theta}(s_{t+1}, a_{t+1})$

$$\theta \leftarrow \theta + \alpha\left(r_{t+1} + \gamma Q_{\theta}(s_{t+1}, a_{t+1}) - Q_{\theta}(s,a)\right)\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

时序差分状态-动作值函数近似



- 策略评估：近似策略评估$Q_\theta \simeq Q^\pi$
- 策略改进：ϵ-贪心策略改进

时序差分学习参数更新过程

- 对于$TD(0)$，时序差分学习的目标是
  - 状态值函数

$$\theta \leftarrow \theta + \alpha\left(V^{\pi}(s_t) - V_{\theta}(s)\right)\frac{\partial V_{\theta}(s_t)}{\partial \theta}$$

$$= \theta + \alpha\left(r_{t+1} + \gamma V_{\theta}(s_{t+1}) - V_{\theta}(s)\right)\frac{\partial V_{\theta}(s_t)}{\partial \theta}$$

  - 动作-状态值函数

$$\theta \leftarrow \theta + \alpha\left(Q^{\pi}(s,a) - Q_{\theta}(s,a)\right)\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

$$= \theta + \alpha\left(r_{t+1} + \gamma Q_{\theta}(s_{t+1},a_{t+1}) - Q_{\theta}(s,a)\right)\frac{\partial Q_{\theta}(s,a)}{\partial \theta}$$

- 虽然θ在时序差分学习的目标中出现，但是我们并不需要计算目标函数的梯度。思考这是为什么？

课程大纲

强化学习基础

1. 强化学习技术概览
2. 马尔可夫决策过程
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化值函数
7. **策略梯度**
8. 深度强化学习 – 价值方法
9. 深度强化学习 – 策略方法

参数化策略

- 我们能够将策略参数化

$$\pi_\theta(a|s)$$

策略可以是确定性的

$$a = \pi_\theta(s)$$

也可以是随机的

$$\pi_\theta(a|s) = P(a|s;\theta)$$

- θ 是策略的参数
- 将可见的已知状态泛化到未知的状态上
- 这里主要讨论的是无模型的强化学习

基于策略的强化学习

优点

- 更加直接的学习目标，即最大化策略的期望回报
- 具有更好的收敛性质
- 在高维度或连续的动作空间中更有效
  - 最重要的因素：基于值函数的方法，通常需要取最大值
- 能够学习出随机策略

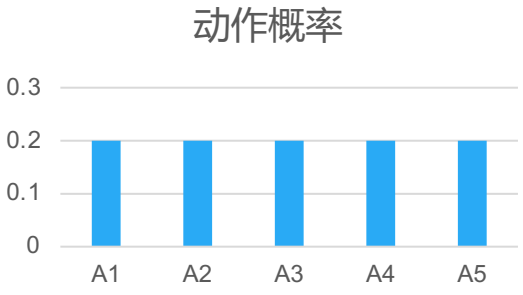
缺点

- 通常会收敛到局部最优而非全局最优
- 评估一个策略通常不够高效并具有较大的方差（variance）

策略梯度

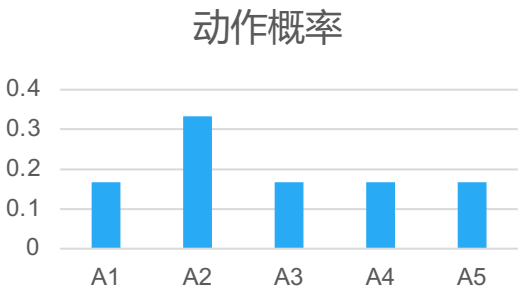
- 对于随机策略$\pi_\theta(a|s) = P(a|s;\theta)$
- 直觉上我们应该
  - 降低带来较低价值/奖励的动作出现的概率
  - 提高带来较高价值/奖励的动作出现的概率
- 一个离散动作空间维度为5的例子

1. 初始化θ



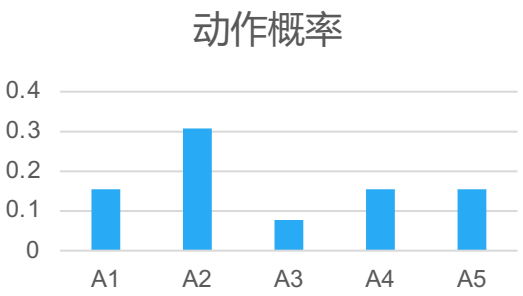
2. 采取动作A2
观察到正的奖励

3. 根据策略梯度更新θ



4. 采取动作A3
观察到负的奖励

5. 根据策略梯度更新θ



单步马尔可夫决策过程中的策略梯度

- 考虑一个简单的单步马尔可夫决策过程
  - 起始状态为$s \sim d(s)$
  - 决策过程在进行一步决策后结束，获得奖励值为r_{sa}
- 策略的价值期望

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum_{s \in S} d(s) \sum_{a \in A} \pi_\theta(a|s) r_{sa}$$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{s \in S} d(s) \sum_{a \in A} \frac{\partial \pi_\theta(a|s)}{\partial \theta} r_{sa}$$

似然比 (Likelihood Ratio)

- 似然比利用下列特性

$$\frac{\partial \pi_\theta(a|s)}{\partial \theta} = \pi_\theta(a|s)\frac{1}{\pi_\theta(a|s)}\frac{\partial \pi_\theta(a|s)}{\partial \theta}$$
$$= \pi_\theta(a|s)\frac{\partial \log \pi_\theta(a|s)}{\partial \theta}$$

- 所以策略的价值期望可以写成

$$J(\theta) = \mathbb{E}_{\pi_\theta}[r] = \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(a|s) r_{sa}$$

$$\frac{\partial J(\theta)}{\partial \theta} = \sum_{s\in S} d(s) \sum_{a\in A} \frac{\partial \pi_\theta(a|s)}{\partial \theta} r_{sa}$$
$$= \sum_{s\in S} d(s) \sum_{a\in A} \pi_\theta(a|s) \frac{\partial \log \pi_\theta(a|s)}{\partial \theta} r_{sa}$$
$$= \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} r_{sa}\right]$$

这一结果可以通过从d(s)中采样状态s和从πθ中采样动作a来近似估计

策略梯度定理

- 策略梯度定理把似然比的推导过程泛化到多步马尔可夫决策过程
  - 用长期的价值函数 $Q^{\pi_\theta}(s,a)$代替前面的瞬时奖励 r_{sa}
- 策略梯度定理涉及
  - 起始状态目标函数 J_1 ，平均奖励目标函数 J_{avR} ，和平均价值目标函数 J_{avV}
- 定理
  - 对任意可微的策略 $\pi_\theta(a|s)$ ，任意策略的目标函数 $J = J_1, J_{avR}, J_{avV}$ ，其策略梯度是

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

详细证明过程请参考:

1. Rich Sutton's Reinforcement Learning: An Introduction (2nd Edition)第13章
2. 动手学强化学习策略梯度的附录 https://hrl.boyuai.com/chapter/2/策略梯度算法

蒙特卡洛策略梯度（REINFORCE）

- 利用随机梯度上升更新参数
- 利用策略梯度定理
- 利用累计奖励值G_t作为$Q^{\pi_\theta}(s,a)$的无偏采样

$$\Delta\theta_t = \alpha \frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta} G_t$$

- REINFORCE算法

initialize 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

$$\theta \leftarrow \theta + \alpha \frac{\partial}{\partial \theta} \log \pi_\theta(a_t|s_t) G_t$$

end for

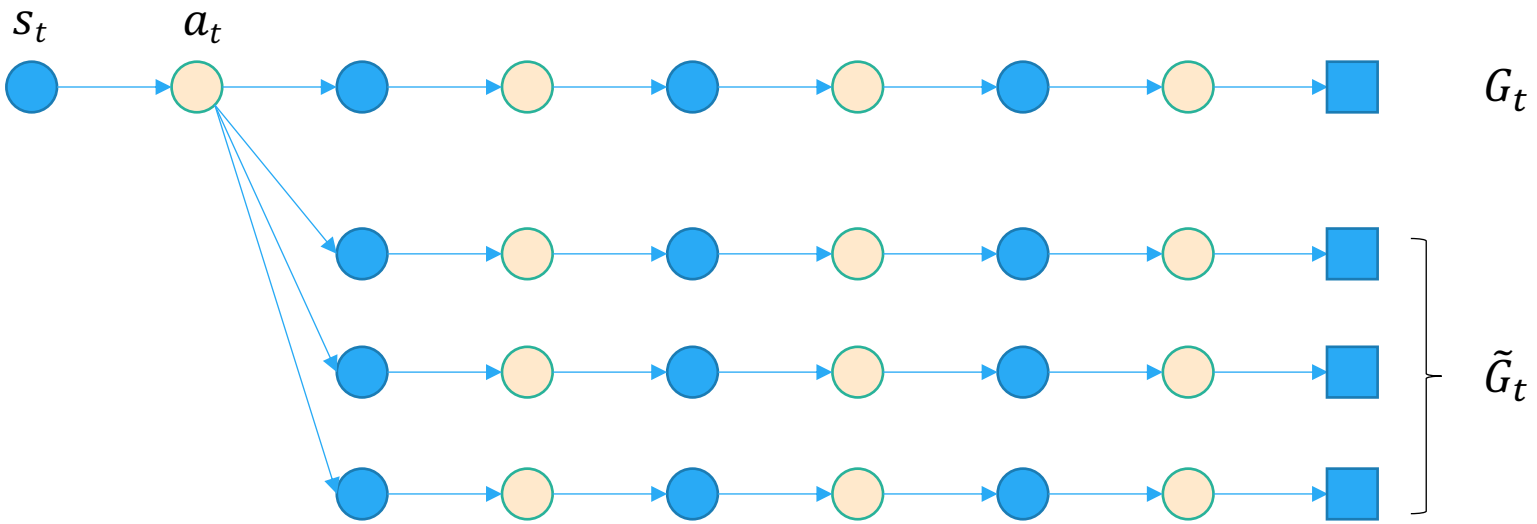
end for

return θ

蒙特卡洛策略梯度（REINFORCE）

$$\Delta\theta_t = \alpha \frac{\partial \log \pi_\theta(a_t|s_t)}{\partial \theta} G_t$$

- 可通过多次roll-out的G_t平均值来逼近$Q(s_t, a_t)$



$$\tilde{G}_t = \frac{1}{N} \sum_{i=1}^{n} G_t^{(i)}$$

回顾：REINFORCE 存在的问题

- 基于片段式数据的任务
  - 通常情况下，任务需要有终止状态，REINFORCE才能直接计算累计折扣奖励
- 低数据利用效率
  - 实际中，REINFORCE需要大量的训练数据
- 高训练方差（最重要的缺陷）
  - 从单个或多个片段中采样到的值函数具有很高的方差

Actor-Critic

- Actor-Critic的思想
  - REINFORCE策略梯度方法：使用蒙特卡洛采样直接估计(s_t, a_t)的值G_t
  - 为什么不建立一个可训练的值函数Q_Φ来完成这个估计过程？
- 演员（Actor）和评论家（Critic）

演员　$\pi_\theta(a|s)$

采取动作使评论家满意的策略

评论家　$Q_\Phi(s,a)$

学会准确估计演员策略所采取动作价值的值函数

Actor-Critic训练

- 评论家Critic: $Q_\Phi(s,a)$
  - 学会准确估计当前演员策略（actor policy）的动作价值

$$Q_\Phi(s,a) \simeq r(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a), a' \sim \pi_\theta(a'|s')}[Q_\Phi(s',a')]$$

- 演员Actor: $\pi_\theta(a|s)$
  - 学会采取使critic满意的动作

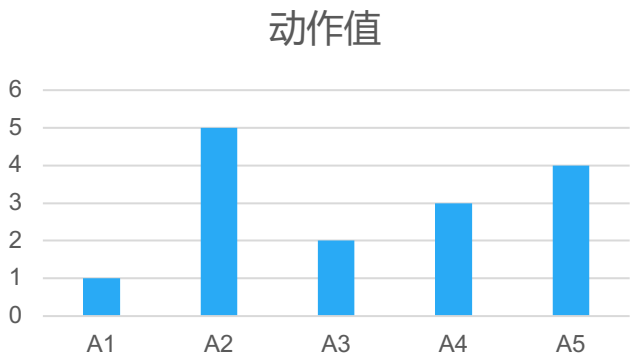
$$J(\theta) = \mathbb{E}_{s \sim p, \pi_\theta}[\pi_\theta(a|s) Q_\Phi(s,a)]$$

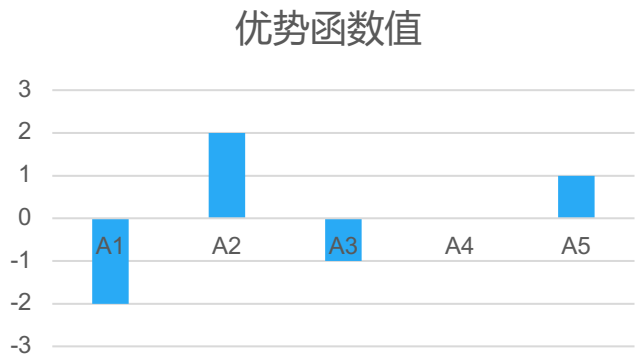
$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q_\Phi(s,a)\right]$$

A2C： Advantageous Actor-Critic

- 思想：通过减去一个基线函数来标准化评论家的打分
  - 更多信息指导：降低较差动作概率，提高较优动作概率
  - 进一步降低方差
- 优势函数（Advantage Function）

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





A2C：Advantageous Actor-Critic

- 状态-动作值和状态值函数

$$
\begin{aligned}
Q^{\pi}(s,a) &= r(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a), a' \sim \pi_{\theta}(a'|s')} \left[Q_{\Phi}(s', a') \right] \\
&= r(s,a) + \gamma \mathbb{E}_{s' \sim p(s'|s,a)} [V^{\pi}(s')]
\end{aligned}
$$

- 因此我们只需要拟合状态值函数来拟合优势函数

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

↑

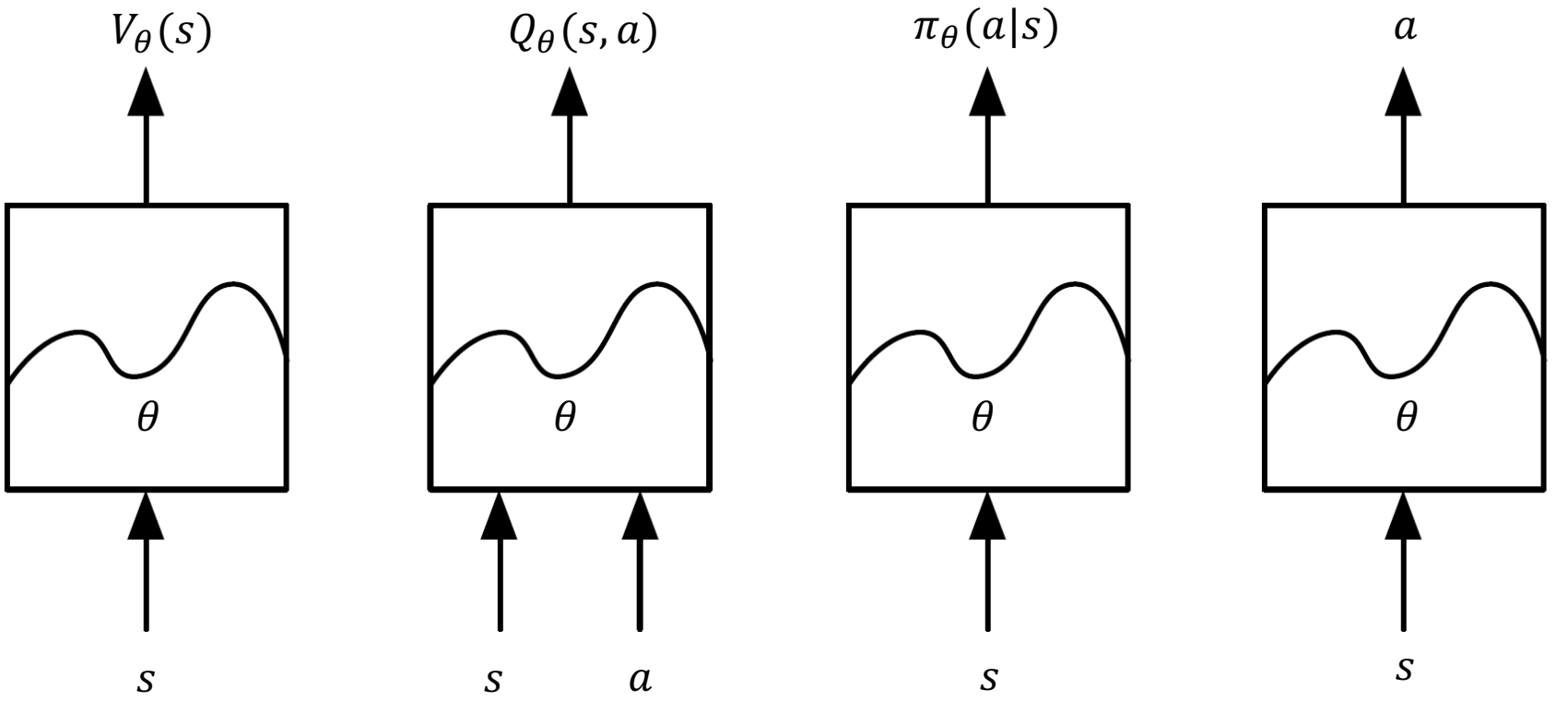
采样下一个状态s'

课程大纲

强化学习基础

1. 强化学习技术概览
2. 马尔可夫决策过程
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化值函数
7. 策略梯度
8. **深度强化学习 – 价值方法**
9. 深度强化学习 – 策略方法

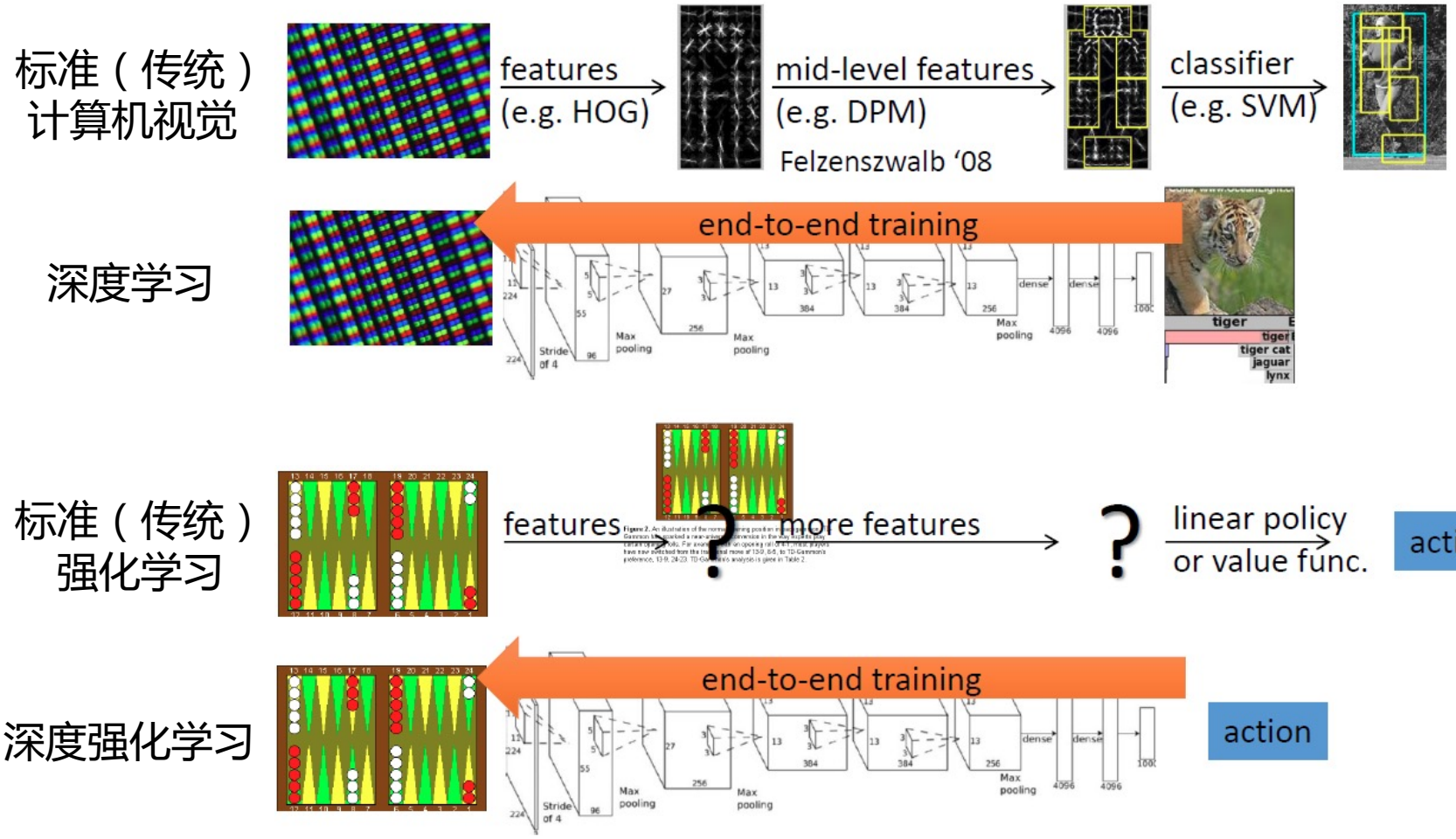
价值和策略近似



状态值函数和状态动作值函数近似　　随机策略近似　　确定性策略近似

- 假如我们直接使用深度神经网络建立这些近似函数呢？

端到端强化学习



深度强化学习使强化学习算法能够以端到端的方式解决复杂问题

Slide from Sergey Levine. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf

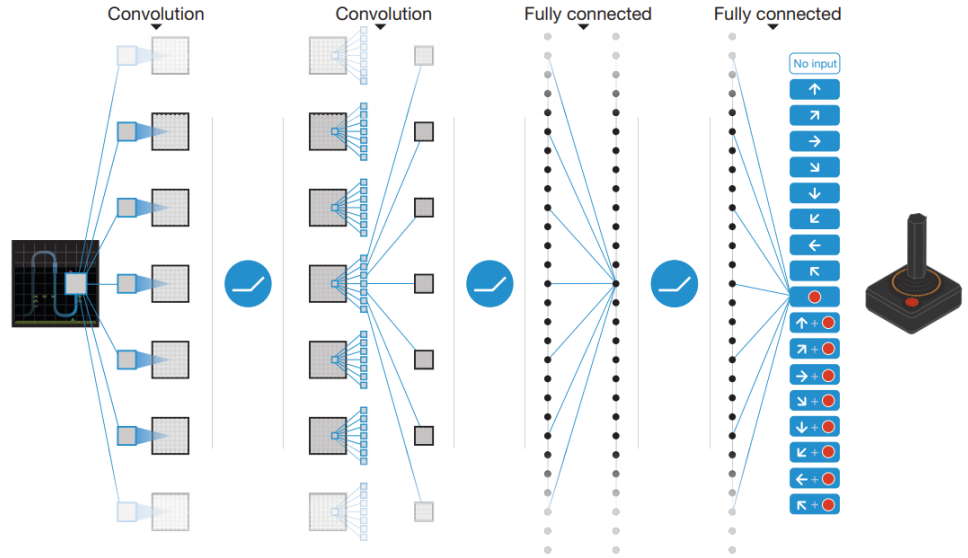
深度强化学习带来的关键变化

- 假如将深度学习（DL）和强化学习（RL）结合在一起会发生什么？
  - 价值函数和策略现在变成了深度神经网络
  - 相当高维的参数空间
  - 难以稳定地训练
  - 容易过拟合
  - 需要大量的数据
  - 需要高性能计算
  - CPU（用于收集经验数据）和GPU（用于训练神经网络）之间的平衡
  - …

这些新的问题促进着深度强化学习算法的创新

深度强化学习 – 价值方法DQN

- 深度Q网络（DQN）
  - 利用深度神经网络进行$Q(s,a)$函数的近似
  - 考虑到$Q(s,a)$函数计算代价高，DQN网络架构输入为s，输出为每个a的价值



Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

深度Q网络（DQN）

直观想法

- 使用神经网络来逼近上述 $Q_\theta(s,a)$
  - 算法不稳定
    - 连续采样得到的 $\{(s_t,a_t,s_{t+1},r_t)\}$ 不满足独立分布。
    - $\{(s_t,a_t,s_{t+1},r_t)\}$ 为状态-动作-下一状态-回报输入。
    - $Q_\theta(s,a)$ 的频繁更新。

解决办法

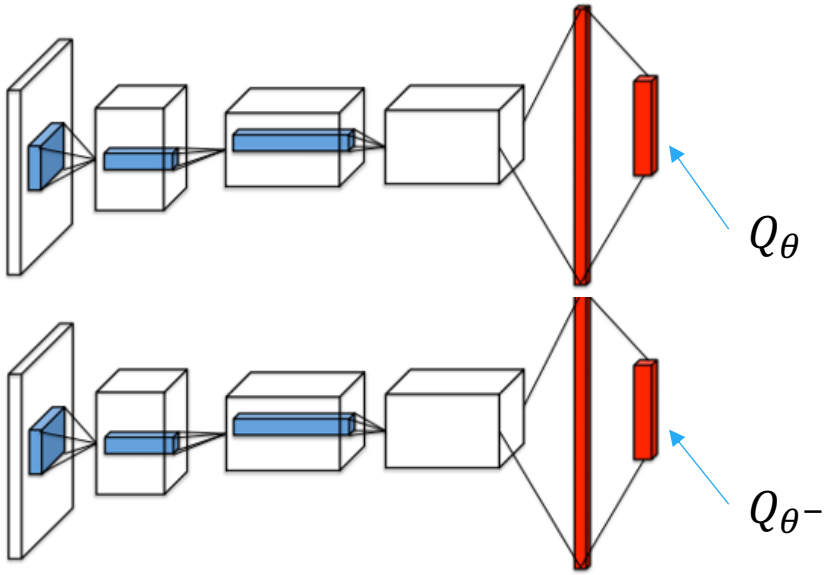
- 经验回放：存储训练过程中的每一步 $e_t=(s_t,a_t,s_{t+1},r_t)$ 于数据库 D 中，采样时服从均匀分布。
- 使用双网络结构：评估网络（evaluation network）和目标网络（target network）

“Human-Level Control Through Deep Reinforcement Learning”，Mnih, Kavukcuoglu, Silver et al. (2015)

目标网络

- 目标网络 $Q_{\theta^-}(s,a)$
  - 使用较旧的参数，记为 θ^-，每隔 C 步和训练网络的参数同步一次。
  - 第i次迭代的损失函数为

$$L_i(\theta_i) = \mathbb{E}_{s_t,a_t,s_{t+1},r_t,p_t\sim D}\left[\frac{1}{2}\omega_t(r_t + \gamma \max_{a'} Q_{\theta_i^-}(s_{t+1},a') - Q_{\theta_i}(s_t,a_t))^2\right]$$



"Human-Level Control Through Deep Reinforcement Learning"，Mnih, Kavukcuoglu, Silver et al. (2015)

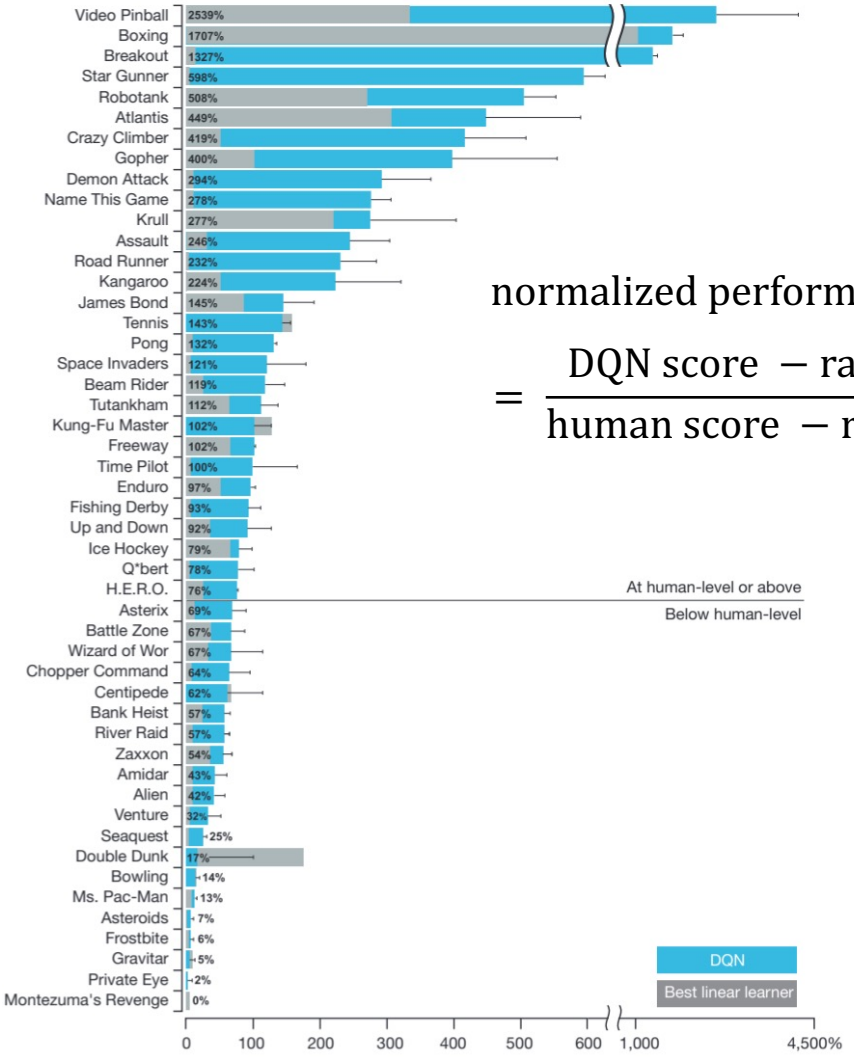
算法流程

Q-learning算法流程

1. 收集数据：使用 $\epsilon-$greedy 策略进行探索，将得到的状态动作组 (s_t, a_t, s_{t+1}, r_t) 放入经验池（replay-buffer）
2. 采样：从经验池中采样 k 个动作状态组
3. 更新网络
   - 用采样得到的数据计算 $Loss$
   - 更新 Q 函数网络 θ
   - 每 C 次迭代（更新Q函数网络）更新一次目标网络 θ^-

"Human-Level Control Through Deep Reinforcement Learning"，Mnih, Kavukcuoglu, Silver et al. (2015)

在 Atari 环境中的实验结果



$$\text{normalized performance} = \frac{\text{DQN score} - \text{random play score}}{\text{human score} - \text{random play score}}$$

The performance of DQN is normalized with respect to a professional human games tester (that is, 100% level)

"Human-Level Control Through Deep Reinforcement Learning", Mnih, Kavukcuoglu, Silver et al. (2015)

Q-learning中的过估计

- Q函数的过高估计
  - Target值 $y_t = r_t + \gamma \max_{a'} Q_\theta(s_{t+1}, a')$

max 操作使得 Q 函数的值越来越大，甚至高于真实值

- 过高估计的原因

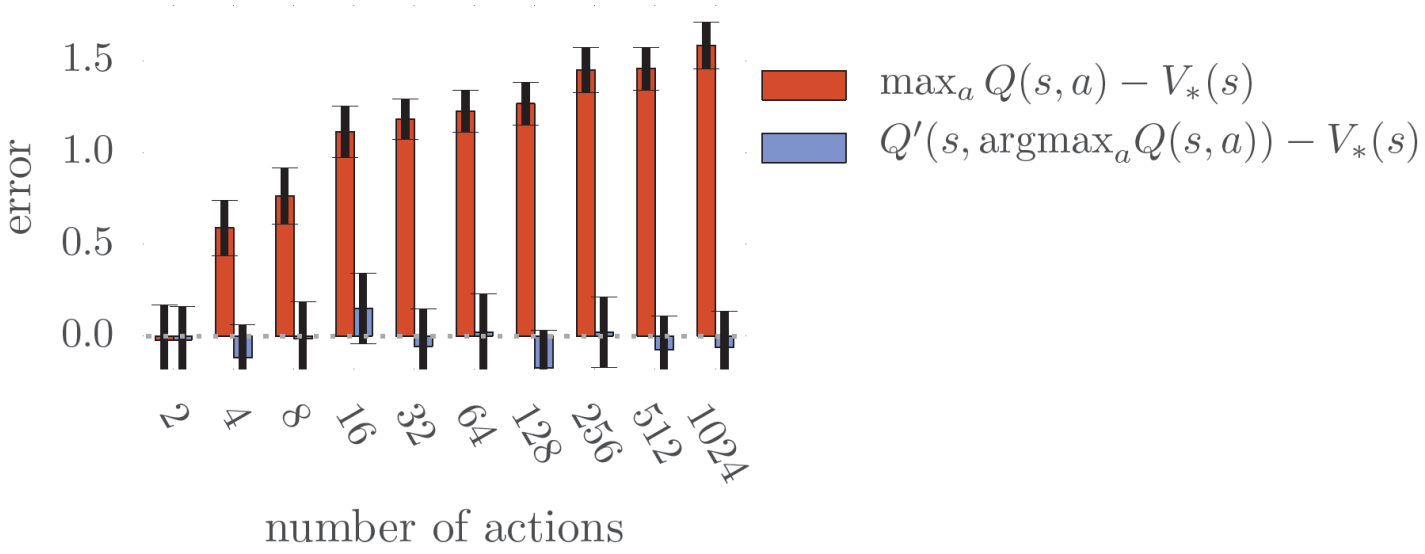
$$\max_{a' \in A} Q_{\theta'}(s_{t+1}, a') = Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_{\theta'}(s_{t+1}, a'))$$

此处选择的a'可能是由于Q函数错误地过高估计导致

"Double Reinforcement Learning with Double Q-Learning"，van Hasselt et al. (2016)
"Double Q-learning"，van Hasselt.(2010)

Q-learning中的过估计

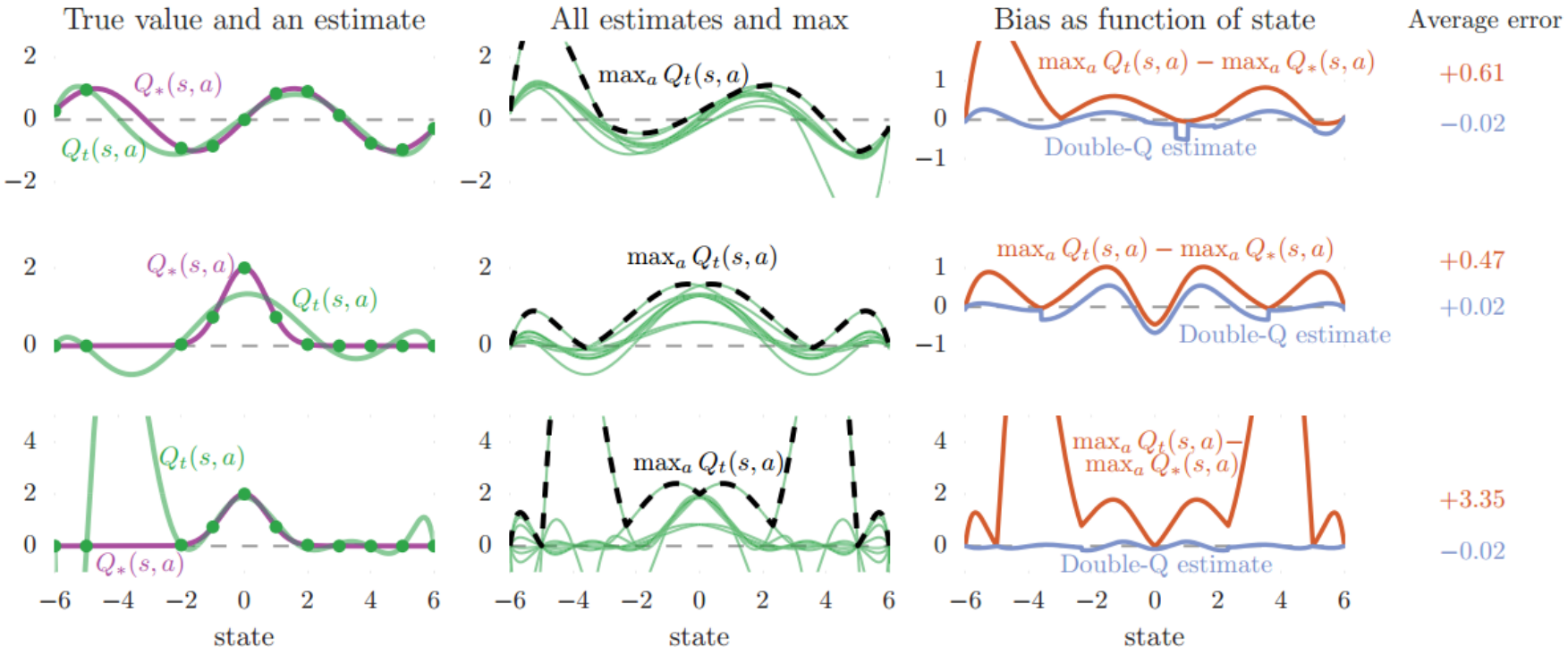
- Q函数的过高估计程度随着候选行动数量增大变得更严重



- 其中$Q_t(s,a) - V_*(s)$设为在[-1,1]区间均匀分布
- Q'函数是另一组独立训练的价值函数

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)
"Double Q-learning", van Hasselt.(2010)

Q-learning中过估计的例子



- 设置：x轴为状态，10个候选行动；紫线是真实价值函数，绿点是训练数据点，绿线是拟合的价值函数
- 中间列展示10个行动的$Q_t(s,a)$估计，在取max后，与真实$Q_*(s,a)$差距太大

"Double Reinforcement Learning with Double Q-Learning"，van Hasselt et al. (2016)

Double DQN

- 使用不同的网络来估值和决策

DQN $y_t = r_t + \gamma Q_\theta(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$

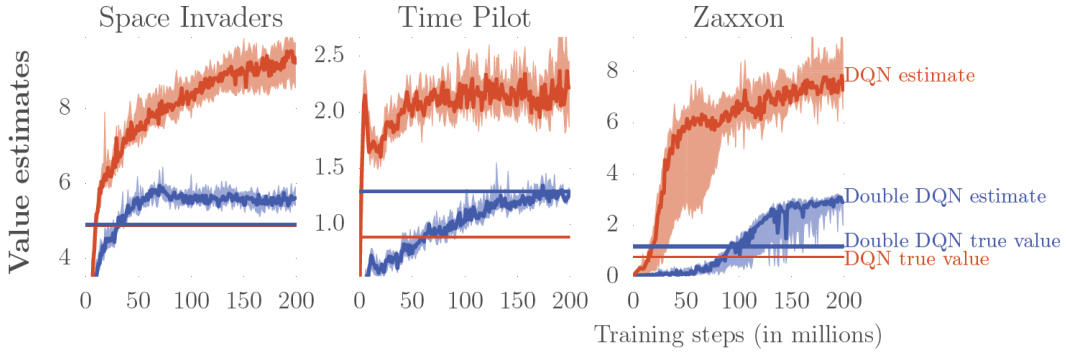
Double DQN $y_t = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a'))$

裁判　　运动员

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

在 Atari 环境中的实验结果

- 价值估计误差

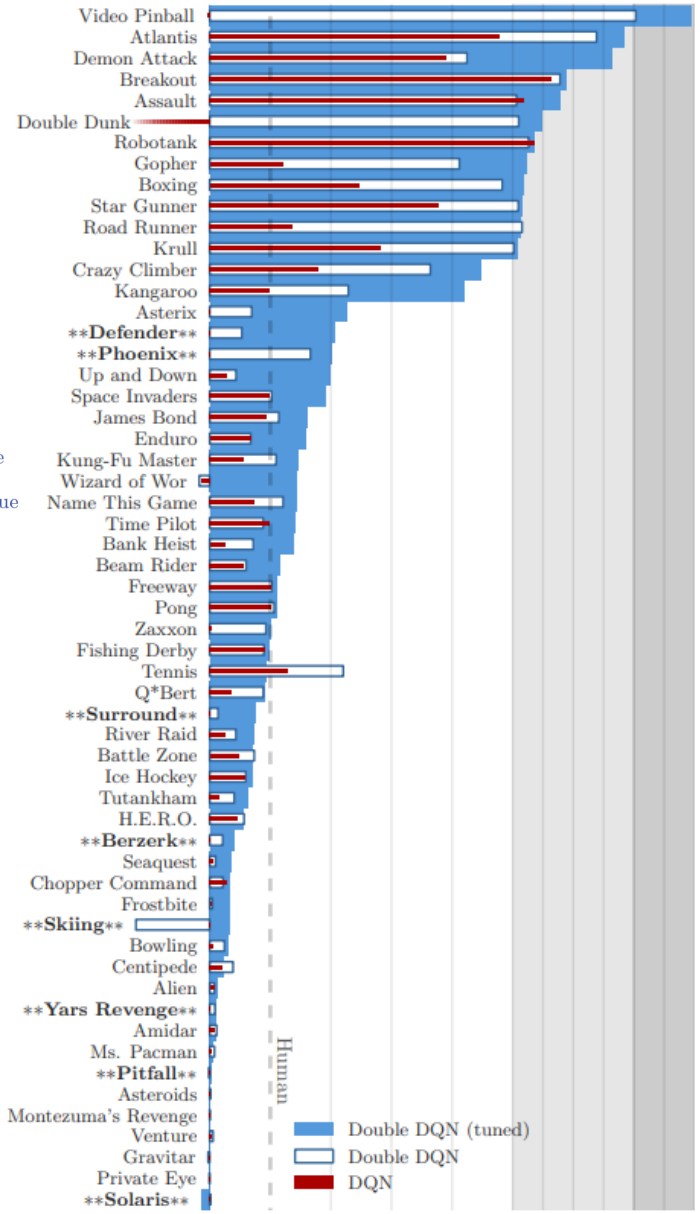


- Atari Performance

	no ops		human starts		
	DQN	DDQN	DQN	DDQN	DDQN (tuned)
Median	93%	**115%**	47%	88%	**117%**
Mean	241%	**330%**	122%	273%	**475%**

normalized performance

$$= \frac{\text{DQN score} - \text{random play score}}{\text{human score} - \text{random play score}}$$



"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

Dueling DQN

假设动作值函数服从某个分布：

$$Q(s,a) \sim \mathcal{N}(\mu,\sigma)$$

显然：$V(s) = \mathbb{E}[Q(s,a)] = \mu$

同样有：$Q(s,a) = \mu + \varepsilon(s,a)$

偏移量

问题

如何描述 $\varepsilon(s,a)$? → $\varepsilon(s,a) = Q(s,a) - V(s)$ → 也称为Advantage函数

Dueling DQN

Advantage 函数

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

$$Q^{\pi}(s,a) = \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$$

$$V^{\pi}(s) = \mathbb{E}_{a\sim\pi(s)}[Q^{\pi}(s,a)]$$

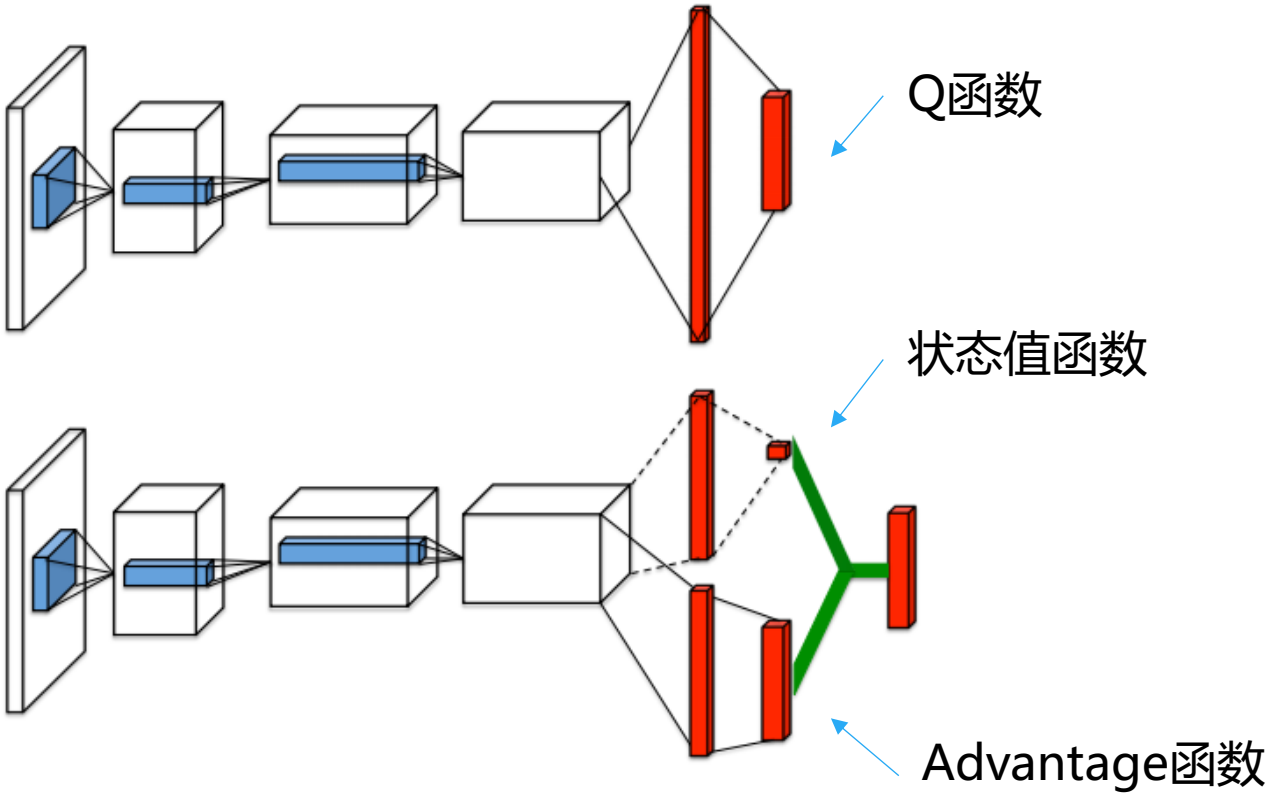
不同的Advantage聚合形式

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \max_{a'\in|A|} A(s,a';\theta,\alpha)\right)$$

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a';\theta,\alpha)\right)$$

"Dueling Network Architectures for Deep Reinforcement Learning", Wang et al. (2016)

网络结构



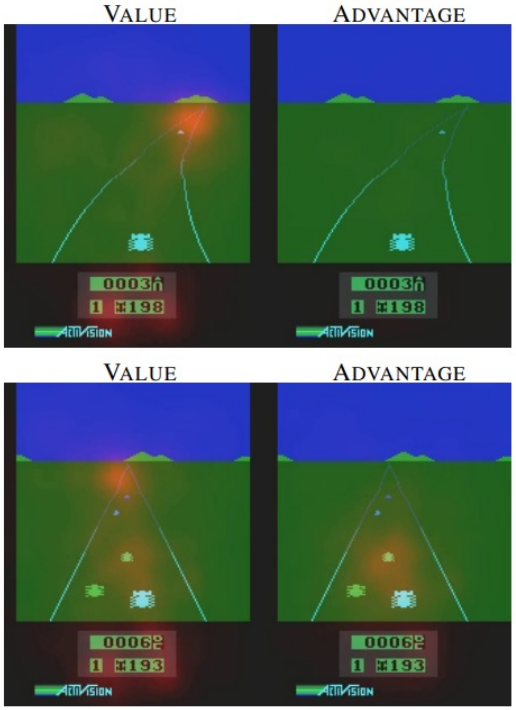
"Dueling Network Architectures for Deep Reinforcement Learning", Wang et al. (2016)

优点

- 处理与动作关联较小的状态
- 状态值函数的学习较为有效：一个状态值函数对应多个Advantage函数

显著区域
(saliency maps)

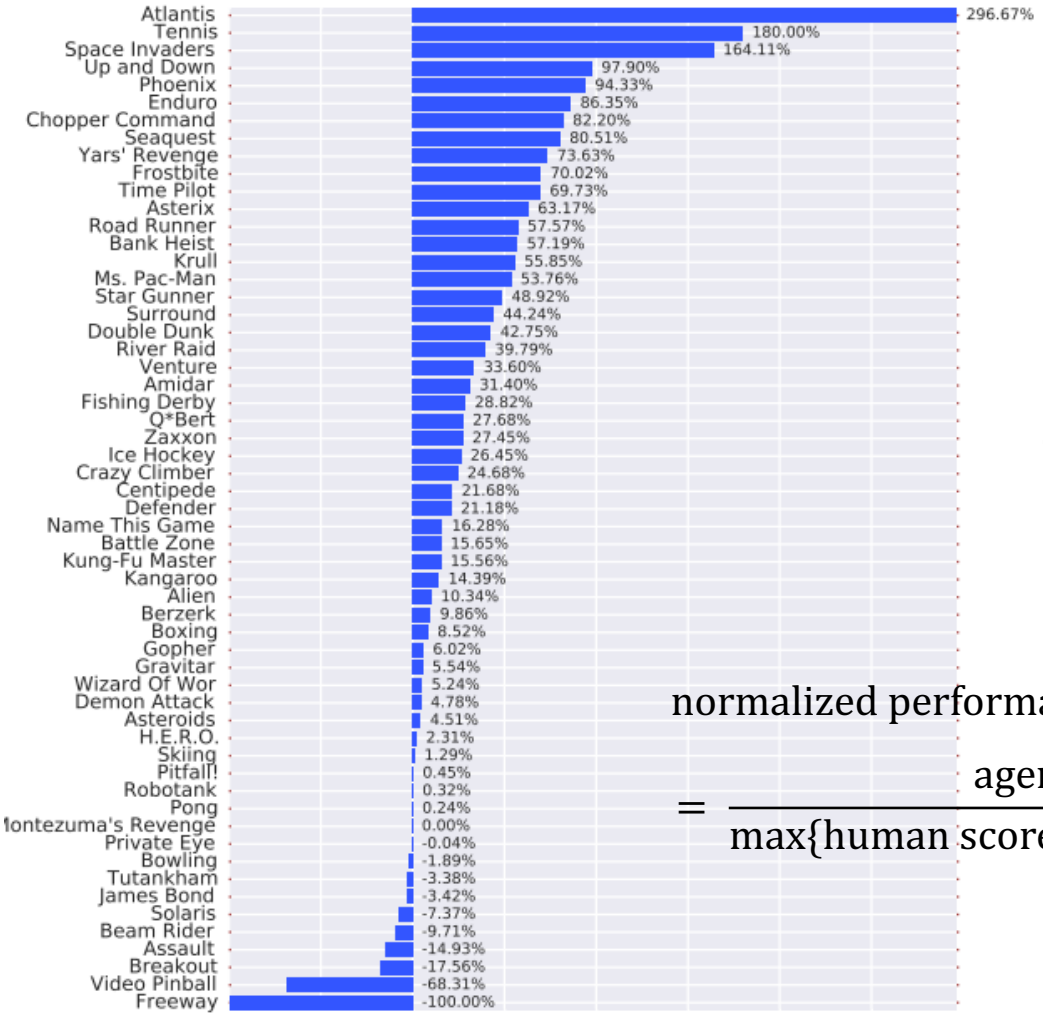
状态价值函数　Advantage函数



可以在不考虑动作的影响下判断出该状态的好坏。

可以强调动作的重要性：advantage学会只在agent面前有车的时候会加强注意力

在 Atari 环境中的实验结果I

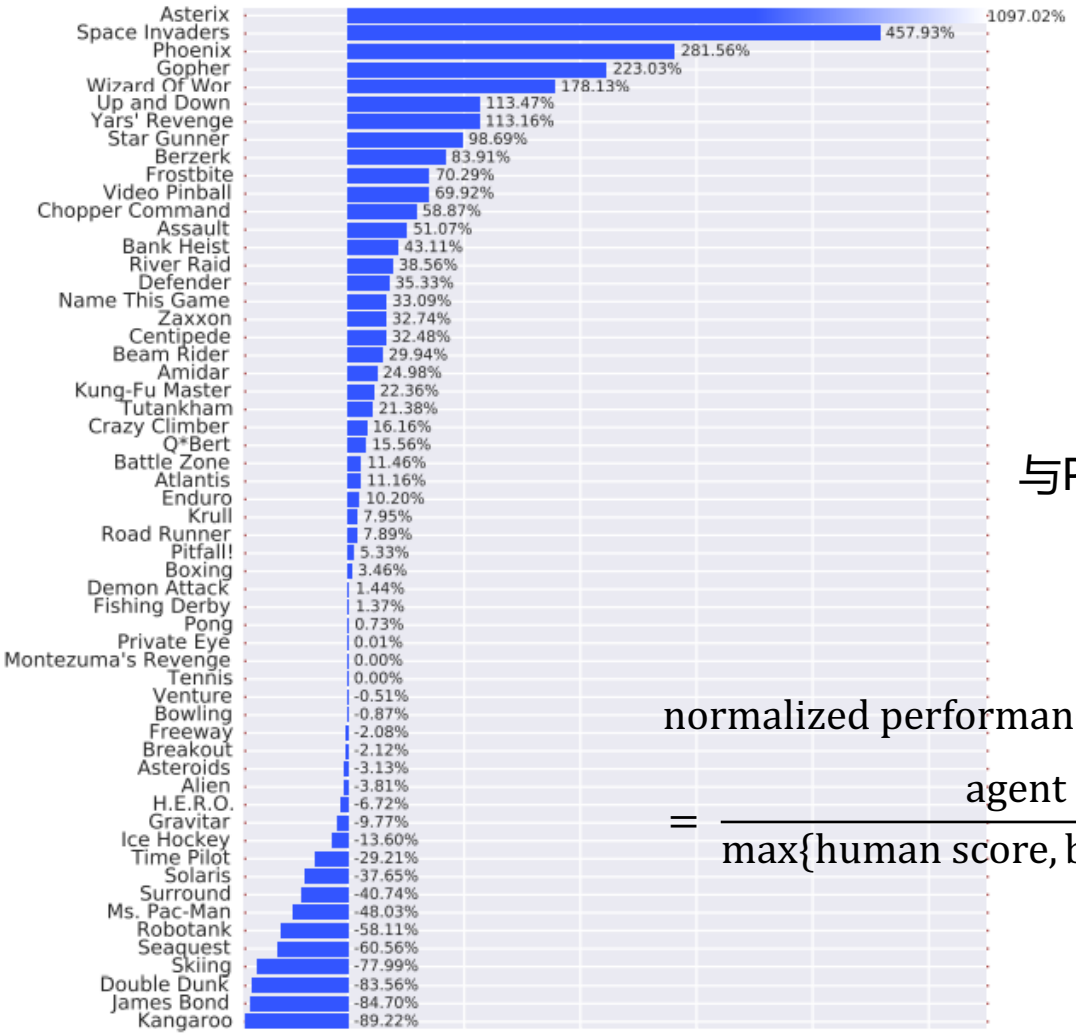


与DQN对比

$$\text{normalized performance} = \frac{\text{agent score} - \text{baseline score}}{\max\{\text{human score}, \text{baseline score}\} - \text{random play score}}$$

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

在 Atari 环境中的实验结果II



与Prioritized Double DQN对比

$$\text{normalized performance} = \frac{\text{agent score} - \text{baseline score}}{\max\{\text{human score}, \text{baseline score}\} - \text{random play score}}$$

"Double Reinforcement Learning with Double Q-Learning", van Hasselt et al. (2016)

课程大纲

强化学习基础

1. 强化学习技术概览
2. 马尔可夫决策过程
3. 动态规划
4. 值函数估计
5. 无模型控制方法
6. 参数化值函数
7. 策略梯度
8. 深度强化学习 – 价值方法
9. **深度强化学习 – 策略方法**

复习：策略梯度定理

- 策略梯度定理把似然比的推导过程泛化到多步马尔可夫决策过程
  - 用长期的价值函数 $Q^{\pi_\theta}(s,a)$代替前面的瞬时奖励 r_{sa}
- 策略梯度定理涉及
  - 起始状态目标函数 J_1 ，平均奖励目标函数 J_{avR} ，和平均价值目标函数 J_{avV}
- 定理
  - 对任意可微的策略 $\pi_\theta(a|s)$ ，任意策略的目标函数 $J = J_1, J_{avR}, J_{avV}$ ，其策略梯度是

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

详细证明过程请参考Rich Sutton’s Reinforcement Learning: An Introduction (2nd Edition)第13章

策略网络的梯度

- 对于随机策略，一般采样到每一个行动的概率由Softmax实现

$$\pi_\theta(a|s) = \frac{e^{f_\theta(s,a)}}{\sum_{a'} e^{f_\theta(s,a')}}$$

  - 其中$f_\theta(s,a)$是对状态-行动队的打分函数，由θ参数化，这可以通过一个神经网络来实现

- 其log形式的梯度为

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \frac{1}{\sum_{a'} e^{f_\theta(s,a')}} \sum_{a''} e^{f_\theta(s,a'')} \frac{\partial f_\theta(s,a'')}{\partial \theta}$$

$$= \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)} \left[\frac{\partial f_\theta(s,a')}{\partial \theta} \right]$$

策略网络的梯度

- 其log梯度形式

$$\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} = \frac{\partial f_\theta(s,a)}{\partial \theta} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\frac{\partial f_\theta(s,a')}{\partial \theta}\right]$$

- 策略网络的梯度为

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} Q^{\pi_\theta}(s,a)\right]$$

$$= \mathbb{E}_{\pi_\theta}\left[\left(\underbrace{\frac{\partial f_\theta(s,a)}{\partial \theta}}_{\text{反向梯度传播}} - \mathbb{E}_{a' \sim \pi_\theta(a'|s)}\left[\underbrace{\frac{\partial f_\theta(s,a')}{\partial \theta}}_{\text{反向梯度传播}}\right]\right) Q^{\pi_\theta}(s,a)\right]$$

策略梯度和Q-learning的对比

- Q 学习算法学习一个由 θ 作为参数的函数 $Q_\theta(s,a)$
  - 优化目标为最小化TD error

$$J(\theta) = \mathbb{E}_{\pi'}\left[\frac{1}{2}\left(r_t + \gamma \max_{a'} Q_{\theta'}(s_{t+1}, a') - Q_\theta(s_t, a_t)\right)^2\right]$$

  - 更新方程

$$\theta \leftarrow \theta - \alpha \frac{\partial J(\theta)}{\partial \theta}$$

$$= \theta + \alpha \mathbb{E}_{\pi'}\left[\left(r_t + \gamma \max_{a'} Q_{\theta'}(s_{t+1}, a') - Q_\theta(s_t, a_t)\right)\frac{\partial Q_\theta(s,a)}{\partial \theta}\right]$$

- 策略梯度学习一个由 θ 作为参数的策略$\pi_\theta(a|s)$
  - 优化目标直接为策略的价值（比Q学习更加直接）

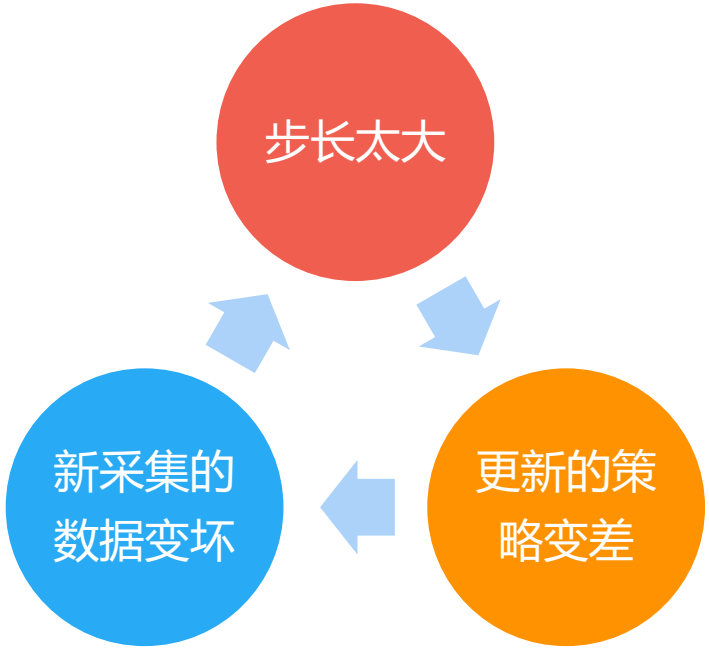
$$\max_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[A^{\pi_\theta}(s,a)]$$

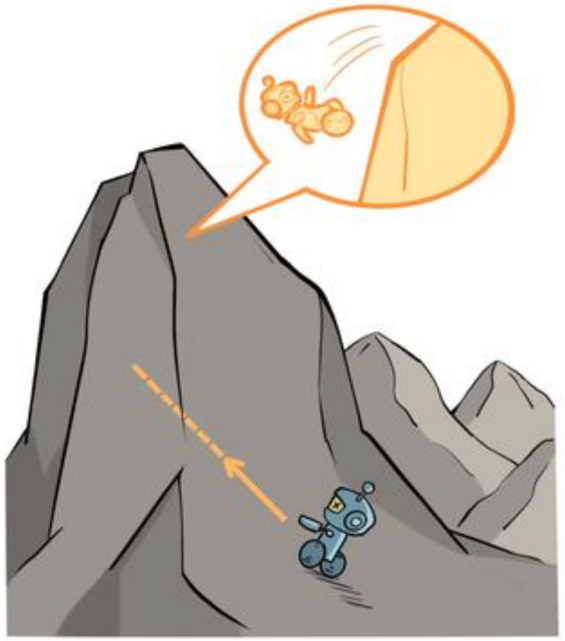
  - 更新方程

$$\theta \leftarrow \theta + \alpha \frac{\partial J(\theta)}{\partial \theta} = \theta + \alpha \mathbb{E}_{\pi_\theta}\left[\frac{\partial \log \pi_\theta(a|s)}{\partial \theta} A^{\pi_\theta}(s,a)\right]$$

策略梯度的缺点

- 适合的步长在策略梯度中难以确定
  - 采集到的数据的分布会随策略的更新而变化。
  - 较差的步长产生的影响大。





"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS" , John Schulman. (2016)

可信区域策略优化 Trust Region Policy Optimization

TRPO策略梯度的优化目标

- 优化目标的两种形式
  - 第一种：$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t \gamma^t r(s_t, a_t)]$
  - 因为 $V^{\pi_\theta}(s) = \mathbb{E}_{a \sim \pi_\theta(s)}[Q^{\pi_\theta}(s,a)] = \mathbb{E}_{a \sim \pi_\theta(s)}\left[\mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_{s_k=s, a_k=a} \sum_{t=k}^{\infty} \gamma^{t-k} r(s_t, a_t)]\right]$。
  - 所以优化目标的第二种形式是： $J(\theta) = \mathbb{E}_{s_0 \sim p_\theta(s_0)}[V^{\pi_\theta}(s_0)]$

相关定义

- τ：轨迹
- s_0：初始状态
- s_t ， a_t ，$r(s_t, a_t)$： t 时刻的状态，动作和奖励
- π_θ：使用的策略
- θ：表示策略所使用的参数
- Q^{π_θ} 和 V^{π_θ}：策略 π_θ 下的 Q 值与状态值函数

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS" ，John Schulman. (2016)

优化目标的优化量

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t \gamma^t r(s_t, a_t)]$$
$$J(\theta) = \mathbb{E}_{s_0 \sim p_\theta(s_0)}[V^{\pi_\theta}(s_0)]$$

$$J(\theta') - J(\theta) = J(\theta') - \mathbb{E}_{s_0 \sim p(s_0)}[V^{\pi_\theta}(s_0)]$$
$$= J(\theta') - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}[V^{\pi_\theta}(s_0)]$$

初始状态的分布与θ无关

$$= J(\theta') - \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t V^{\pi_\theta}(s_t) - \sum_{t=1}^{\infty} \gamma^t V^{\pi_\theta}(s_t)\right]$$
$$= J(\theta') + \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t))\right]$$

$J(\theta')$的定义

$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right] + \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (\gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t))\right]$$
$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}\left[\sum_{t=0}^{\infty} \gamma^t (r(s_t, a_t) + \gamma V^{\pi_\theta}(s_{t+1}) - V^{\pi_\theta}(s_t))\right]$$

$Q^{\pi_\theta}(s_t, a_t)$

$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)}[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t)]$$

不方便采样

$A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS" , John Schulman. (2016)

使用重要性采样

- 使用重要性采样（Importance Sampling）

$$A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - \mathrm{V}^{\pi_\theta}(s_t)$$

$$J(\theta') - J(\theta)$$

$$= \mathbb{E}_{\tau \sim p_{\theta'}(\tau)} \left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_\theta}(s_t, a_t) \right]$$

$$= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)} [\mathbb{E}_{a_t \sim \pi_{\theta'}(a_t|s_t)} [\gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

$$= \sum_t \mathbb{E}_{s_t \sim p_{\theta'}(s_t)} [\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)} [\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

仍然是 $p_{\theta'}$
（近似操作）

重要性采样

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS"，John Schulman. (2016)

忽略状态分布的差异

- 当策略更新前后的变化较小时，可以令 $p_\theta(s_t) \approx p_{\theta'}(s_t)$。
  - 假设使用确定性策略，当 $\pi_{\theta'}(s_t) \neq \pi_\theta\ (s_t)$ 的概率小于ϵ时
  - 或者假设使用随机策略，当$a' \sim \pi_{\theta'}(\cdot\,|s_t) \neq a \sim \pi_\theta(\cdot\,|s_t)$的概率小于$\epsilon$时
  - $p_{\theta'}(s_t) = (1-\epsilon)^t p_\theta(s_t) + (1-(1-\epsilon)^t) p_{mistake}(s_t)$
  - $|p_{\theta'}(s_t) - p_\theta(s_t)| = (1-(1-\epsilon)^t)|p_{mistake}(s_t) - p_\theta(s_t)| \leq 2(1-(1-\epsilon)^t) \leq 2\epsilon t$

$(1-\epsilon)^t \geq 1-\epsilon t \ for\ \epsilon \in [0,1]$

$$J(\theta') - J(\theta) \approx \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS" , John Schulman. (2016)

TRPO约束策略的变化

- 使用KL散度约束策略更新的幅度

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

$$\text{such that } \mathbb{E}_{s_t \sim p_\theta(s_t)}[D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t))] \leq \epsilon$$

- 可以使用基于泰勒展开和共轭梯度的求解方法（见动手学强化学习11.3节）
- 实际多使用constraint violate as penalty

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t, a_t)]] - \lambda(D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) - \epsilon)$$

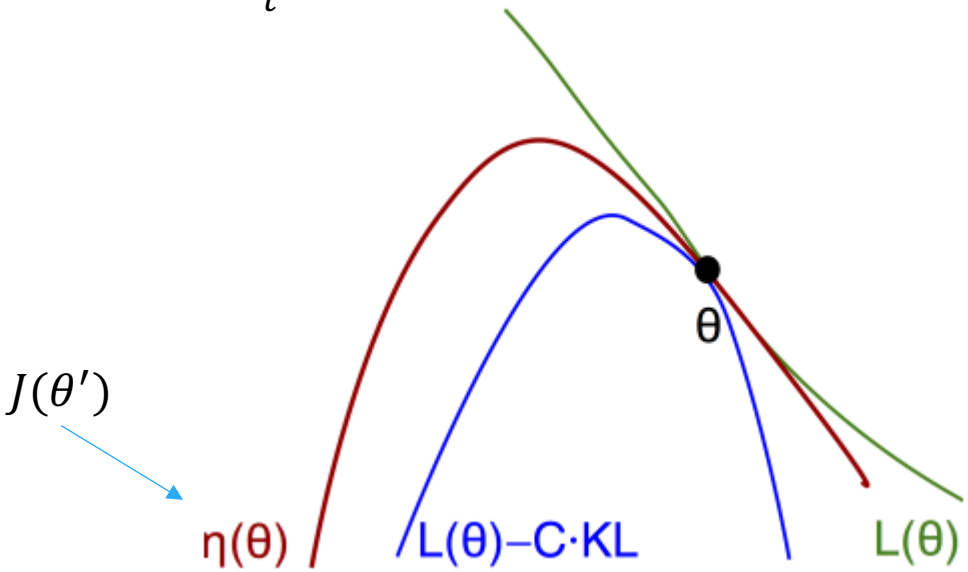
1. 优化上式，更新θ'
2. 更新 $\lambda \leftarrow \lambda + \alpha(D_{KL}(\pi_{\theta'}(a_t|s_t) \parallel \pi_\theta(a_t|s_t)) - \epsilon)$

"OPTIMIZING EXPECTATIONS: FROM DEEP REINFORCEMENT LEARNING TO STOCHASTIC COMPUTATION GRAPHS"，John Schulman. (2016)
http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-9.pdf

TRPO策略改进的单调性保证

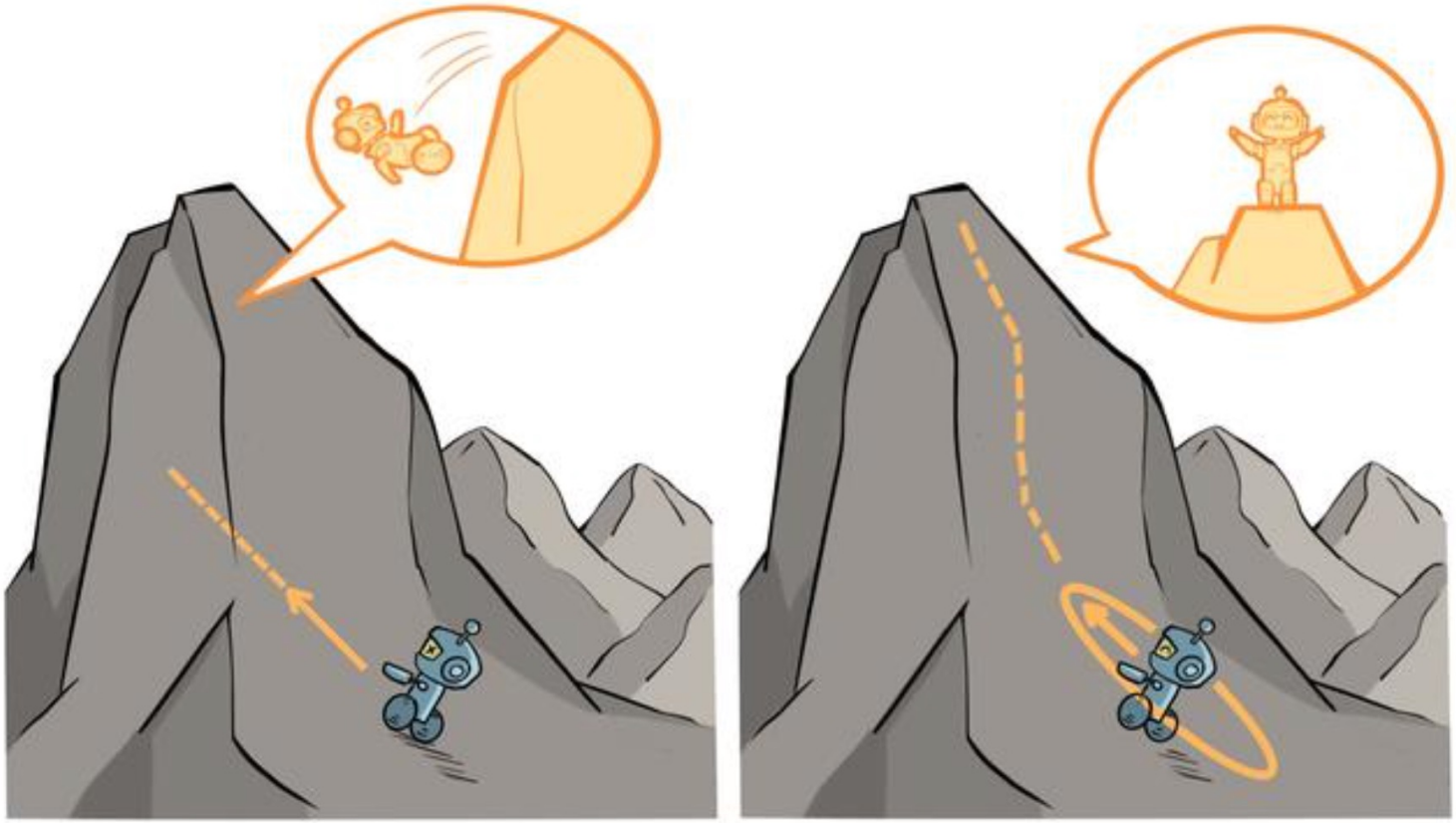
$$J(\theta') \geq L_\theta(\theta') - C \cdot D_{KL}^{max}(\theta, \theta'), where\ C = \frac{4\epsilon\gamma}{(1-\gamma)^2}, \epsilon = \max_{s,a} |A_\pi(s,a)|$$

$$L_\theta(\theta') = J(\theta) + \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} \gamma^t A^{\pi_\theta}(s_t, a_t)]]$$



"Advanced Policy Gradient Methods: Natural Gradient, TRPO, and More " , John Schulman. (2017)

TRPO的原理



Gradient Ascent　　Optimization in Trust Region

回顾TRPO

- TRPO使用KL散度约束策略更新的幅度

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t, a_t)]]$$

$$\text{such that } D_{KL}(\pi_{\theta'}(a_t|s_t) \| \pi_\theta(a_t|s_t)) \leq \epsilon$$

  - 使用constraint violate as penalty

$$\theta' \leftarrow \arg\max_{\theta'} \sum_t \mathbb{E}_{s_t \sim p_\theta(s_t)}[\mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t)}[\frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)}\gamma^t A^{\pi_\theta}(s_t, a_t)]] - \lambda(D_{KL}(\pi_{\theta'}(a_t|s_t) \| \pi_\theta(a_t|s_t)) - \epsilon)$$

1. 优化上式，更新θ'
2. 更新 $\lambda \leftarrow \lambda + \alpha(D_{KL}(\pi_{\theta'}(a_t|s_t) \| \pi_\theta(a_t|s_t)) - \epsilon)$

TRPO的不足

- 重要性比例带来的大方差
- 求解约束优化问题的困难

近端策略优化
PPO: Proximal Policy Optimization

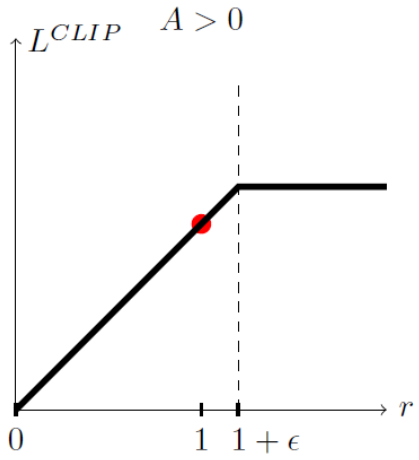
PPO在TRPO基础上的改进

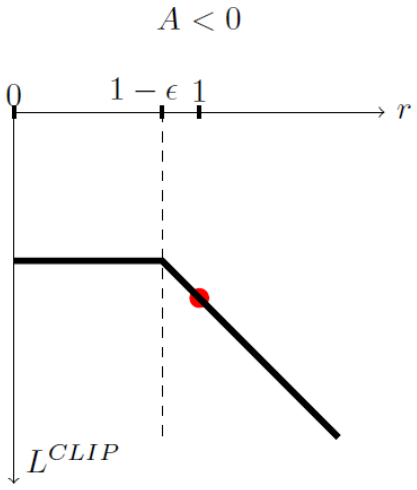
1. 截断式优化目标

conservative policy iteration

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$





构建下界

$$L^{CLIP}(\theta) \leq L^{CPI}(\theta)$$

在$r = 1$附近相等

$$L^{CLIP}(\theta) = L^{CPI}(\theta)$$

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO: Proximal Policy Optimization

PPO在TRPO基础上的改进

1. 截断式优化目标

conservative policy iteration

$$L^{CPI}(\theta) = \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}\hat{A}_t\right] = \hat{\mathbb{E}}_t\left[r_t(\theta)\hat{A}_t\right]$$

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

2. 优势函数$\hat{A}_t$选用多步时序差分

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t} V(s_T)$$

- 在每次迭代中，可以选择并行N个actor收集T步经验数据
- 计算每步的$\hat{A}_t$和$L^{CLIP}(\theta)$，构成mini-batch
- 更新参数θ，并更新$\theta_{\text{old}} \leftarrow \theta$

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO: Proximal Policy Optimization

PPO在TRPO基础上的改进

3. 自适应的KL惩罚项参数

$$L^{KLPEN}(\theta) = \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t - \beta \text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t)|\pi_\theta(\cdot|s_t)] \right]$$

动态调整β方法

- 计算KL值 $d = \hat{\mathbb{E}}_t \left[\text{KL}\left[\pi_{\theta_{\text{old}}}(\cdot|s_t)\middle|\pi_\theta(\cdot|s_t)\right] \right]$
  a) 如果$d < d_{\text{targ}}/1.5$，更新$\beta \leftarrow \beta/2$
  b) 如果$d > d_{\text{targ}} \times 1.5$，更新$\beta \leftarrow \beta \times 2$

注：这里1.5和2是经验参数，算法效能和它们并不是很敏感

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

PPO实验对比

No clipping or penalty: $L_t(\theta) = r_t(\theta)\hat{A}_t$

Clipping: $L_t(\theta) = \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$

KL penalty (fixed or adaptive) $L_t(\theta) = r_t(\theta)\hat{A}_t - \beta\,\text{KL}[\pi_{\theta_{\text{old}}}, \pi_\theta]$

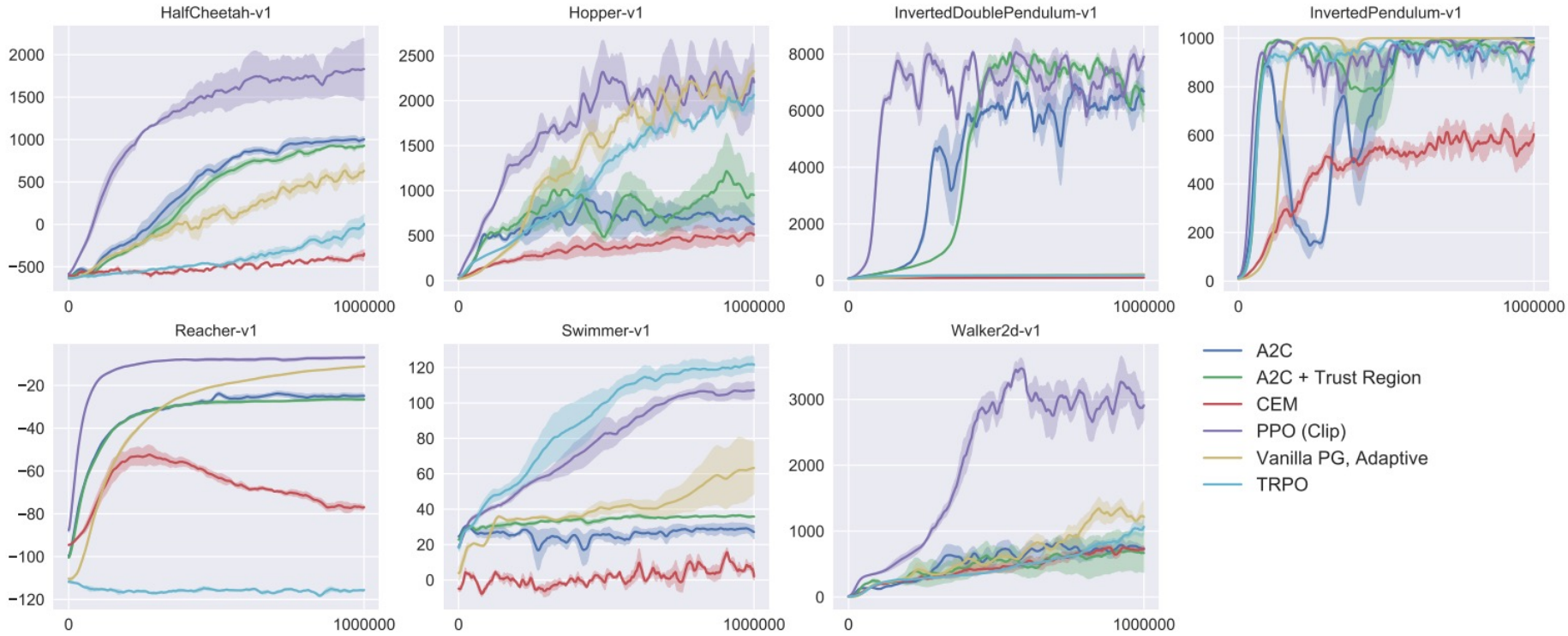
- 7个连续控制的环境
- 3个random seed
- 每个算法跑100个episode，跑21遍，做平均值计算
- 最佳score归一化为1

algorithm	avg. normalized score
No clipping or penalty	-0.39
Clipping, $\epsilon = 0.1$	0.76
Clipping, $\epsilon = 0.2$	**0.82**
Clipping, $\epsilon = 0.3$	0.70
Adaptive KL $d_{\text{targ}} = 0.003$	0.68
Adaptive KL $d_{\text{targ}} = 0.01$	0.74
Adaptive KL $d_{\text{targ}} = 0.03$	0.71
Fixed KL, $\beta = 0.3$	0.62
Fixed KL, $\beta = 1.$	0.71
Fixed KL, $\beta = 3.$	0.72
Fixed KL, $\beta = 10.$	0.69

Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

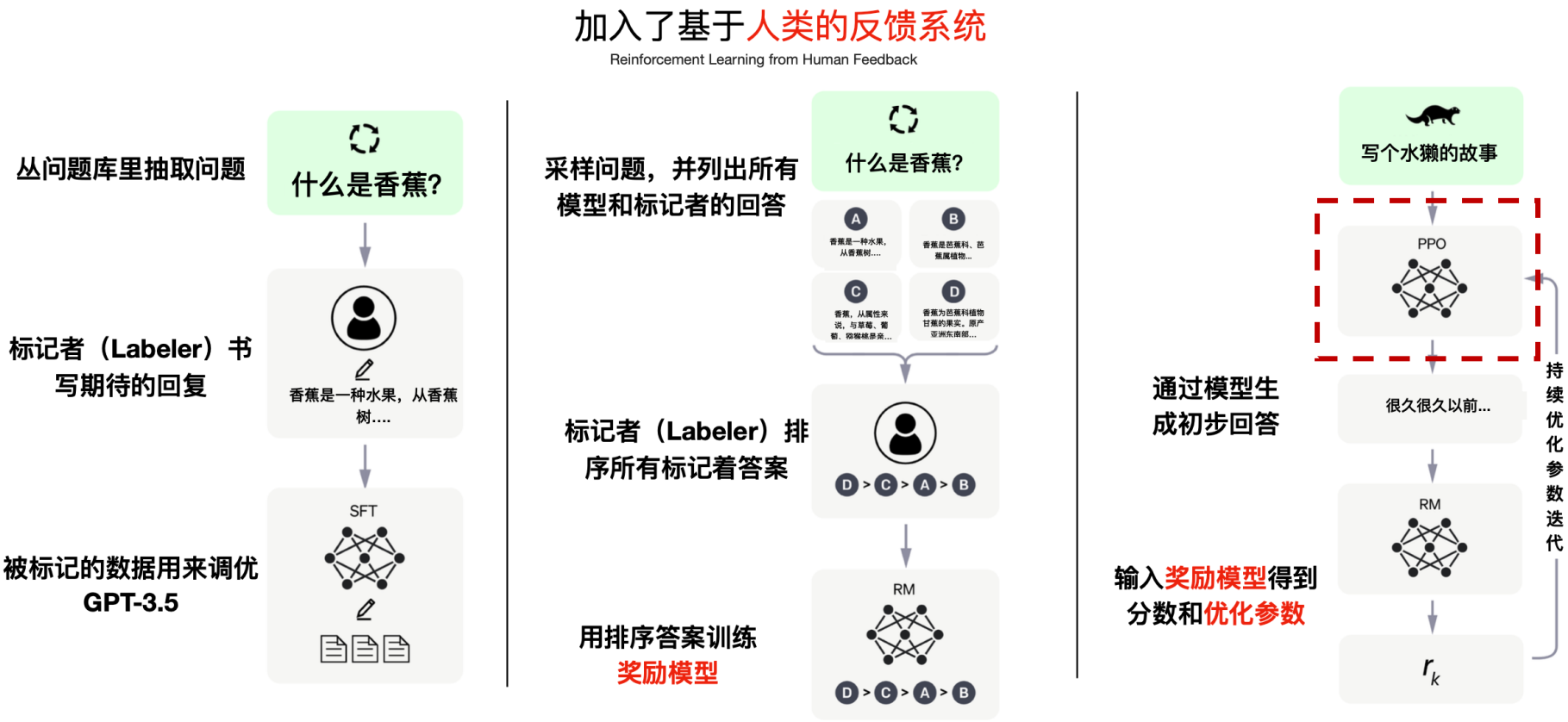
PPO实验对比

- 在MuJoCo实验环境中的对比结果（训练100万步）



Proximal Policy Optimization Algorithms, John Schulman, et al. (2017)

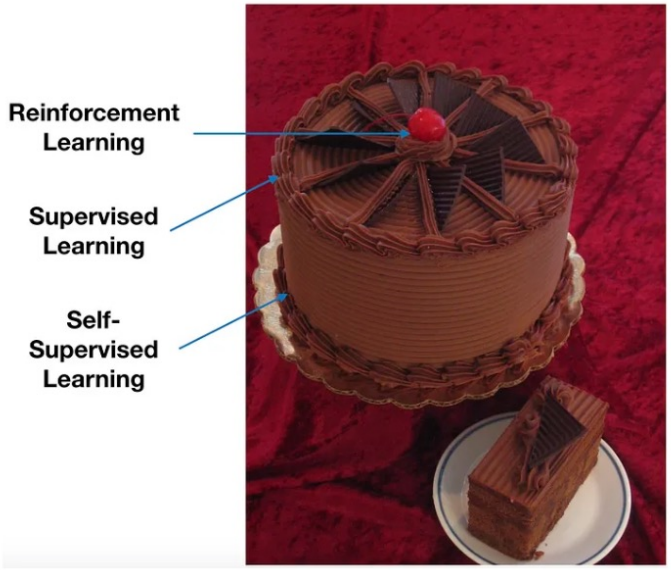
PPO在ChatGPT中的使用



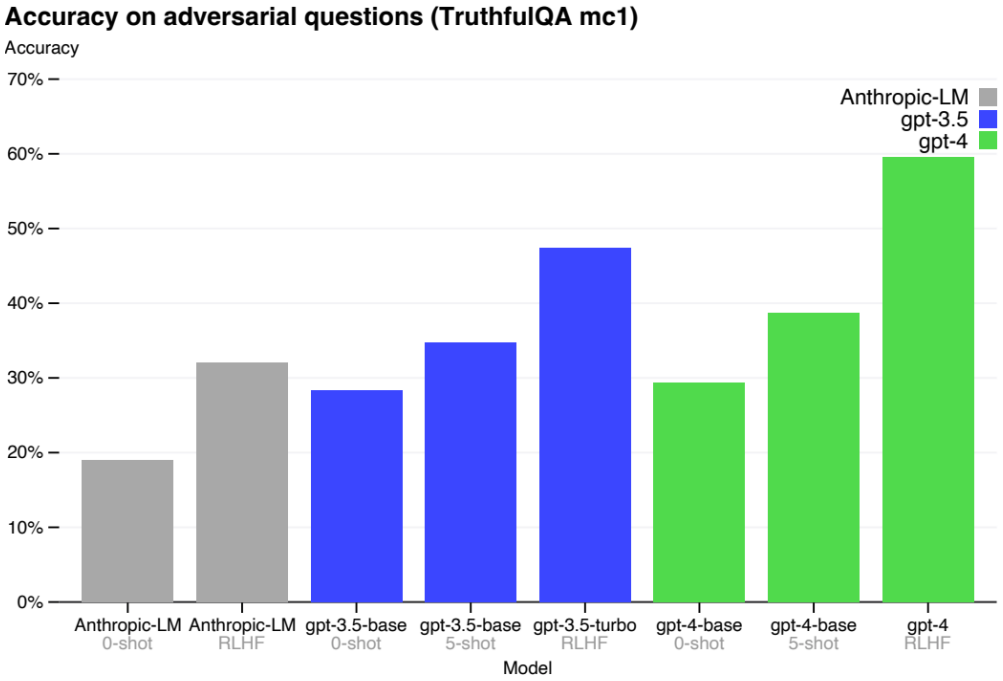
PPO在GPT4中的使用

大家之前以为强化学习的作用

"If intelligence is a cake, the bulk of the cake is self-supervised learning, the icing on the cake is supervised learning, and the cherry on the cake is reinforcement learning (RL)." — Yann LeCun head of Facebook AI



大家现在看到的强化学习的作用



from GPT-4 Technical Report

PPO在O1中的使用


September 12, 2024
Learning to Reason with LLMs
We are introducing OpenAI o1, a new large language model trained with reinforcement learning to perform complex reasoning. o1 thinks before it answers —it can produce a long internal chain of thought before responding to the user.
Contributions

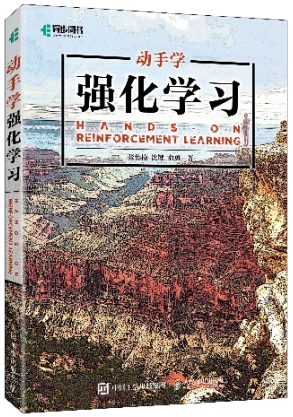
课程总结

强化学习基础

1. **强化学习**：决策与预测的区别、动态数据分布
2. **马尔可夫决策过程**：马尔可夫性质、占用度量
3. **动态规划**：策略迭代、价值迭代、策略提升
4. **值函数估计**：蒙特卡洛方法、时序差分方法
5. **无模型控制方法**：SARSA、Q-learning、同异策略
6. **参数化值函数**：Q函数梯度回传法则
7. **策略梯度**：策略梯度、Actor Critic
8. **深度强化学习 – 价值方法**：DQN及其变种
9. **深度强化学习 – 策略方法**：TRPO、PPO

坚定理想信念
持续强化学习

THANK YOU



欢迎关注《动手学强化学习》

所有学习材料请见：
https://wnzhang.net/teaching/sjtu-rl-2024/
https://hrl.boyuai.com/
https://space.bilibili.com/3546754433681656