机器学习（2024 秋季学期）

十六、强化学习

大纲

- 什么是强化学习？
- K-摇臂赌博机
- 有模型学习
- 免模型学习
- 值函数近似
- 一些方向

什么是强化学习?

- 从大了来说：操控智能体完成指定任务；
  可以是一系列任务，也可以是单一任务；
  通常需要做一系列动作；
  “审时度势”。

- 从小了来说：瓜农种西瓜

种下瓜苗后：(为简便，仅考虑浇水和不浇水两个动作，不考虑施肥、除草等)

$$\text{缺水} \xrightarrow{\text{浇水}} \text{健康} \xrightarrow{\text{浇水}} \text{溢水} \xrightarrow{\text{不浇水}} \text{缺水} \xrightarrow{\text{浇水}} \text{健康} \quad \text{收获好西瓜}$$

什么是强化学习?

- 例子：瓜农种西瓜
  - 多步决策过程
  - 过程中包含状态、动作、反馈 (奖赏) 等
  - 需多次种瓜，在过程中不断摸索，才能总结出较好的种瓜策略

种下瓜苗后：(为简便，仅考虑浇水和不浇水两个动作，不考虑施肥、除草等)

缺水 —浇水→ 健康 —浇水→ 溢水 —不浇水→ 缺水 —浇水→ 健康　收获好西瓜

抽象该过程：强化学习 (reinforcement learning)

马尔科夫决策过程

- 强化学习常用马尔可夫决策过程 (Markov Decision Process) 描述
  - 机器所处的环境 E
    - 例如在种西瓜任务中，环境是西瓜生长的自然世界
  - 状态空间 $\{X: x \in X\}$ 是机器感知到的环境的描述
    - 瓜苗长势的描述
  - 智能体能采取的行为空间 A
  - 潜在的状态转移(概率)函数 $P: X \times A \times X \to \mathbb{R}$
    - 瓜苗当前状态缺水，选择动作浇水，有一定概率恢复健康，也有一定概率无法恢复
  - 潜在的奖赏(reward)函数 $R: X \times A \times X \to \mathbb{R}$ (或 $R: X \times X \to \mathbb{R}$)
    - 瓜苗健康对应奖赏+1，瓜苗凋零对应奖赏-10浇水，施肥等
  - 策略(policy) $\pi: X \to A$ (或 $\pi: X \times A \to \mathbb{R}$)
    - 根据瓜苗状态是缺水时，返回动作浇水

什么是强化学习?

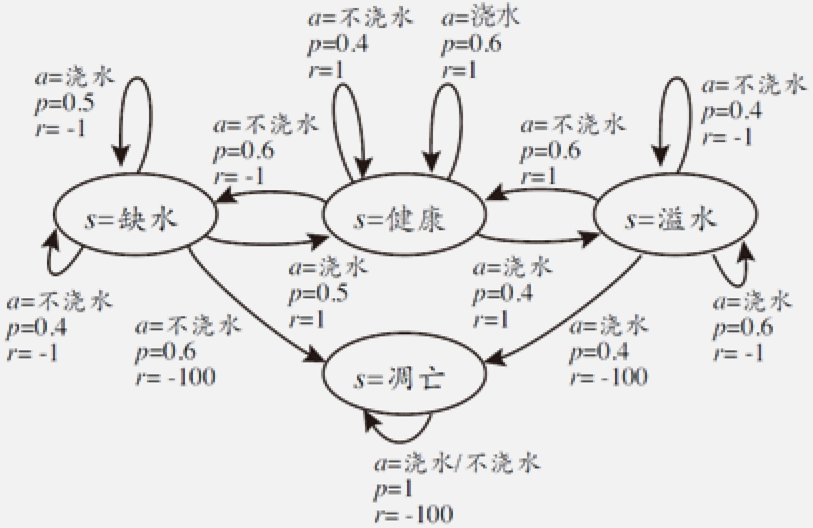
- 强化学习对应了四元组

$$E = <X, A, P, R>$$

- 强化学习的目标
  - 机器通过在环境中不断尝试从而学到一个策略 π，使得长期执行该策略后得到的期望累积奖赏最大

T 步期望累积奖赏： $\mathbb{E}[\frac{1}{T}\sum_{t=1}^{T} r_t]$

γ 期望折扣累积奖赏： $\mathbb{E}[\sum_{t=0}^{+\infty} \gamma^t r_{t+1}]$



$$a = \pi(x) \qquad < x_0, a_0, r_1, x_1, a_1, r_2, \ldots, x_{T-1}, a_{T-1}, r_T, x_T >$$

最大化 $\mathbb{E}[\frac{1}{T}\sum_{t=1}^{T} r_t]$

强化学习问题基本设置

- 强化学习 vs. 监督学习
  - 监督学习：给有标记样本

$$\text{示例} \xrightarrow{\text{分类器/回归器 } f} \text{标记}$$

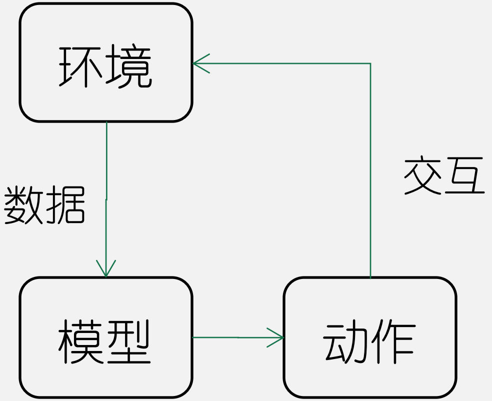
  - 强化学习：没有有标记样本，通过执行动作之后反馈的奖赏来学习

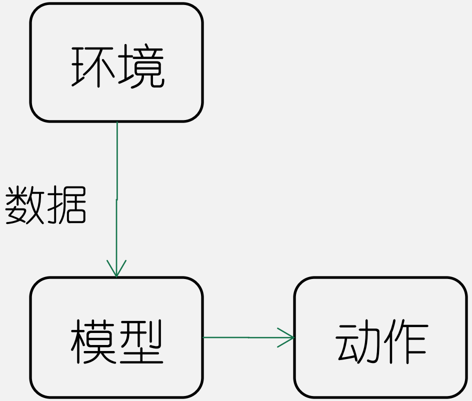
$$\text{状态} \xrightarrow{\text{策略 } \pi} \text{动作}$$

强化学习在某种意义上可以认为是具有“延迟标记信息”的监督学习

强化学习问题基本设置

- 强化学习 vs. 监督学习





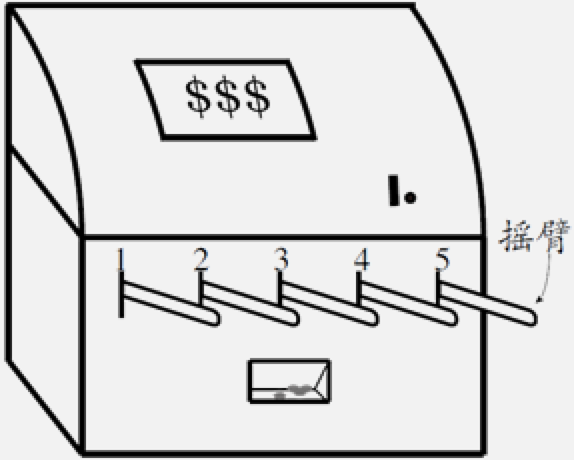
强化学习的样本可来自于与环境的交互过程。

大纲

- 什么是强化学习？
- K-摇臂赌博机
- 有模型学习
- 免模型学习
- 值函数近似
- 一些方向

K-摇臂赌博机

- K-摇臂赌博机 (K-Armed Bandit)
  - 只有一个状态，K个动作
  - 每个摇臂的奖赏服从某个期望未知的分布
  - 执行有限次动作
  - 最大化累积奖赏
- 强化学习面临的主要困难：探索-利用窘境 (Exploration-Exploitation dilemma)
  - 探索：估计不同摇臂的优劣 (奖赏期望的大小)
  - 利用：选择当前最优的摇臂
- 在探索与利用之间进行折中
  - ϵ - 贪心
  - Softmax



K-摇臂赌博机

- ϵ - 贪心
  - 以 ϵ 的概率探索：均匀随机选择一个摇臂
  - 以 $1-\epsilon$ 的概率利用：选择当前平均奖赏最高的摇臂

- Softmax：基于当前已知的摇臂平均奖赏来对探索与利用折中
  - 若某个摇臂当前的平均奖赏越大，则它被选择的概率越高
  - 概率分配使用Boltzmann分布：

$$P(k)=\frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(i)}{\tau}}}$$

其中， $Q(i)$ 记录当前摇臂的平均奖赏

- 两种算法都有一个折中参数(ϵ,τ)，算法性能孰好孰坏取决于具体应用问题

大纲

- 什么是强化学习?
- K-摇臂赌博机
- 有模型学习
- 免模型学习
- 值函数近似
- 一些方向

有模型学习

- 有模型学习 (model-based learning)：$E =< X, A, P, R >$
  - X, A, P, R 均已知
  - 方便起见，假设状态空间和动作空间均有限
- 强化学习的目标：找到使累积奖赏最大的策略 π
- 策略评估：使用某策略所带来的累积奖赏

状态值函数：从状态 x 出发，使用策略 π 所带来的累积奖赏

$$\begin{cases} V_T^\pi(x) = \mathbb{E}_\pi\left[\frac{1}{T}\sum_{t=1}^{T} r_t | x_0 = x\right], & T \text{ 步期望累积奖赏；} \\ V_\gamma^\pi(x) = \mathbb{E}_\pi\left[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x\right], & \gamma \text{ 期望折扣累积奖赏.} \end{cases}$$

状态-动作值函数：从状态 x 出发，执行动作 a 后再使用策略 π 所带来的累积奖赏

$$\begin{cases} Q_T^\pi(x, a) = \mathbb{E}_\pi\left[\frac{1}{T}\sum_{t=1}^{T} r_t | x_0 = x, a_0 = a\right]; \\ Q_\gamma^\pi(x, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{+\infty} \gamma^t r_{t+1} | x_0 = x, a_0 = a\right]. \end{cases}$$

有模型学习

- 给定 π，值函数的计算：值函数具有简单的递归形式
  - T 步累积奖赏：

$$V_T^\pi(x) = \mathbb{E}_\pi\left[\frac{1}{T}\sum_{t=1}^{T} r_t | x_0 = x\right]$$

$$= \mathbb{E}_\pi\left[\frac{1}{T}r_1 + \frac{T-1}{T}\frac{1}{T-1}\sum_{t=2}^{T} r_t | x_0 = x\right]$$

(全概率公式)

$$= \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^a\left(\frac{1}{T}R_{x\to x'}^a + \frac{T-1}{T}\mathbb{E}_\pi\left[\frac{1}{T-1}\sum_{t=1}^{T-1} r_t | x_0 = x'\right]\right)$$

$$= \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^a\left(\frac{1}{T}R_{x\to x'}^a + \frac{T-1}{T}V_{T-1}^\pi(x')\right).$$

Bellman等式

  - 折扣累积奖赏：

$$V_\gamma^\pi(x) = \sum_{a\in A}\pi(x,a)\sum_{x'\in X} P_{x\to x'}^a\left(R_{x\to x'}^a + \gamma V_\gamma^\pi(x')\right).$$

有模型学习

- 给定 π ，状态-动作值函数的计算：通过值函数来表示

$$\begin{cases} Q_T^{\pi}(x,a) = \sum_{x' \in X} P_{x \to x'}^a \left(\frac{1}{T} R_{x \to x'}^a + \frac{T-1}{T} V_{T-1}^{\pi}(x') \right); \\ Q_{\gamma}^{\pi}(x,a) = \sum_{x' \in X} P_{x \to x'}^a \left(R_{x \to x'}^a + \gamma V_{\gamma}^{\pi}(x') \right). \end{cases}$$

- 最优策略，最优值函数，最优状态-动作值函数
  - 最优策略：最大化累积奖赏 $\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{x \in X} V^{\pi}(x).$
  - 最优值函数： $\forall x \in X : V^*(x) = V^{\pi^*}(x).$
  - 最优状态-动作值函数

有模型学习

- 策略评估：评估一个策略的值函数：

$$V^{\pi}(x) = R\big(x, \pi(x)\big) + \gamma \sum_{x'} P_{x \to x'} V^{\pi}(x')$$

- 求解方法：$V_t^{\pi}(x) = R\big(x, \pi(x)\big) + \gamma \sum_{x'} P_{x \to x'} V_{t-1}(x')$
  - 通过这种方式得到的值函数收敛到正确的值函数；

$$\begin{aligned}
\|V_t^{\pi}(x) - V^{\pi}(x)\|_{\infty} &= \max_{x} \gamma \sum_{x'} P_{x \to x'} |V_{t-1}^{\pi}(x') - V^{\pi}(x')| \\
&\le \gamma \sum_{x'} P_{x \to x'} \max_{x'} |V_{t-1}^{\pi}(x') - V^{\pi}(x')| \\
&= \gamma \max_{x'} |V_{t-1}^{\pi}(x') - V^{\pi}(x')| \\
&= \gamma \|V_{t-1}^{\pi}(x) - V^{\pi}(x)\|_{\infty}
\end{aligned}$$

有模型学习

- 策略改进：将非最优策略改进为最优策略
  - 最优值函数/最优策略满足：

$$\begin{cases} V_T^*(x) = \max_{a\in A} \sum_{x'\in X} P_{x\to x'}^a \left(\frac{1}{T} R_{x\to x'}^a + \frac{T-1}{T} V_{T-1}^*(x')\right); \\ V_\gamma^*(x) = \max_{a\in A} \sum_{x'\in X} P_{x\to x'}^a \left(R_{x\to x'}^a + \gamma V_\gamma^*(x')\right). \end{cases}$$

最优Bellman等式

$$V^*(x) = \max_{a\in A} Q^{\pi^*}(x,a).$$

  - 非最优策略的改进方式：将策略选择的动作改为当前最优的动作

$$\pi'(x) = \underset{a\in A}{\arg\max}\, Q^{\pi}(x,a).$$

有模型学习

- 策略迭代 (policy iteration)：求解最优策略的方法
  - 随机策略作为初始策略
  - 策略评估+策略改进+策略评估+策略改进+……
  - 直到策略收敛
- 策略迭代算法的缺点
  - 每次改进策略后都要重新评估策略，计算开销较大

输入: MDP 四元组 $E = \langle X, A, P, R \rangle$;
累积奖赏参数 T.

过程:

$\forall x \in X: V(x) = 0,\ \pi(x,a) = \frac{1}{|A(x)|}$;
loop
    for $t = 1, 2, \ldots$ **do**
        $\forall x \in X: V'(x) = \sum_{a \in A} \pi(x,a) \sum_{x' \in X} P^a_{x \to x'} \left(\frac{1}{t} R^a_{x \to x'} + \frac{t-1}{t} V(x') \right)$;
        if $t = T + 1$ **then**
            break
        else
            $V = V'$
        end if
    end for
    $\forall x \in X: \pi'(x) = \arg\max_{a \in A} Q(x,a)$;
    if $\forall x: \pi'(x) = \pi(x)$ **then**
        break
    else
        $\pi = \pi'$
    end if
end loop

输出: 最优策略 π

图 16.8 基于 T 步累积奖赏的策略迭代算法

有模型学习

- 有模型学习小结
  - 强化学习任务可归结为基于动态规划的寻优问题
  - 与监督学习不同，在有模型学习时，我们不关注策略的泛化能力，而是通过规划的方式为每一个状态找到最好的动作
- 问题：如果模型未知呢？

大纲

- 什么是强化学习？
- K-摇臂赌博机
- 有模型学习
- 免模型学习
- 值函数近似
- 一些方向

免模型学习

- 免模型学习 (model-free learning)：更加符合实际情况
  - 转移概率，奖赏函数未知
  - 甚至环境中的状态数目也未知
  - 假定状态空间有限
- 免模型学习所面临的困难
  - 策略无法评估
  - 无法通过值函数计算状态-动作值函数
  - 机器只能从一个起始状态开始探索环境
- 解决困难的办法
  - 多次采样
  - 直接估计每一对状态-动作的值函数
  - 在探索过程中逐渐发现各个状态

免模型学习

- 蒙特卡罗强化学习：采样轨迹，用样本均值近似期望
  - 策略评估：蒙特卡罗法
    - 从某状态出发，执行某策略
    - 对轨迹中出现的每对状态-动作，记录其后的奖赏之和
    - 采样多条轨迹，每个状态-动作对的累积奖赏取平均
  - 策略改进：换入当前最优动作

一条轨迹：

$$< x_0, a_0, r_1, x_1, a_1, r_2, \ldots, x_{T-1}, a_{T-1}, r_T, x_T >$$

- 蒙特卡罗强化学习可能遇到的问题：轨迹的单一性

- 解决问题的办法

$$\pi^{\epsilon}(x) = \begin{cases} \pi(x), & \text{以概率 } 1-\epsilon; \\ A \text{ 中以均匀概率选取的动作}, & \text{以概率 } \epsilon. \end{cases}$$

  - ϵ -贪心法
    - 同策略：被评估与被改进的是同一个策略
    - 异策略：被评估与被改进的是不同的策略 (用重要性采样技术)

免模型学习

同策略蒙特卡罗强化学习算法

输入: 环境 E;
动作空间 A;
起始状态 x_0;
策略执行步数 T.

过程:

1: $Q(x,a)=0,\ \mathrm{count}(x,a)=0,\ \pi(x,a)=\frac{1}{|A(x)|}$;
2: **for** $s=1,2,\dots$ **do**
3: 在 E 中执行策略 π 产生轨迹
$< x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T >$;
4: **for** $t=0,1,\dots,T-1$ **do**
5: $R=\frac{1}{T-t}\sum_{i=t+1}^{T} r_i$;
6: $Q(x_t,a_t)=\frac{Q(x_t,a_t)\times\mathrm{count}(x_t,a_t)+R}{\mathrm{count}(x_t,a_t)+1}$;
7: $\mathrm{count}(x_t,a_t)=\mathrm{count}(x_t,a_t)+1$
8: **end for**
9: 对所有已见状态 x :

$$\pi(x)=\begin{cases}\arg\max_{a'} Q(x,a'), & \text{以概率 } 1-\epsilon;\\ \text{以均匀概率从 } A \text{ 中选取动作}, & \text{以概率 } \epsilon\ .\end{cases}$$

10: **end for**

输出: 策略 π

图 16.10 同策略蒙特卡罗强化学习算法

免模型学习

异策略蒙特卡罗强化学习算法

输入： 环境 E;
动作空间 A;
起始状态 x_0;
策略执行步数 T.

过程：

$Q(x,a)=0,\ \mathrm{count}(x,a)=0,\ \pi(x,a)=\frac{1}{|A(x)|}$;
for $s=1,2,\ldots$ **do**
  在 E 中执行 π 的 ϵ-贪心策略产生轨迹
  $< x_0, a_0, r_1, x_1, a_1, r_2, \ldots, x_{T-1}, a_{T-1}, r_T, x_T >$;
  $p_i = \begin{cases} 1-\epsilon+\epsilon/|A|, & a_i=\pi(x); \\ \epsilon/|A|, & a_i \neq \pi(x), \end{cases}$
  for $t=0,1,\ldots,T-1$ **do**
    $R=\frac{1}{T-t}\sum_{i=t+1}^{T}(r_i \times \prod_{j=i}^{T-1}\frac{1}{p_j})$;
    $Q(x_t,a_t)=\frac{Q(x_t,a_t)\times \mathrm{count}(x_t,a_t)+R}{\mathrm{count}(x_t,a_t)+1}$;
    $\mathrm{count}(x_t,a_t)=\mathrm{count}(x_t,a_t)+1$
  end for
  $\pi(x)=\arg\max_{a'} Q(x,a')$
end for

输出： 策略 π

图 16.11 异策略蒙特卡罗强化学习算法

免模型学习

- 蒙特卡罗强化学习的缺点：低效
  - 求平均时以“批处理式”进行
  - 在一个完整的采样轨迹完成后才对状态-动作值函数进行更新
- 克服缺点的办法：时序差分 (Temporal Difference，TD) 学习
- 时序差分学习
  - 增量式地进行状态-动作值函数更新
  - ϵ -贪心法
    - 同策略：Sarsa算法
    - 异策略：Q-学习 (Q-learning)

$$Q_t^{\pi}(x,a) = \frac{1}{t}\sum_{i=1}^{t} r_i \longrightarrow Q_{t+1}^{\pi}(x,a) = Q_t^{\pi}(x,a) + \frac{1}{t+1}\left(r_{t+1} - Q_t^{\pi}(x,a)\right)$$

免模型学习

Sarsa算法

用于更新 Q 的下一时刻动作 a' 即为下一时刻真正执行的动作

输入： 环境 E;
动作空间 A;
起始状态 x_0;
奖赏折扣 γ;
更新步长 α.

过程：

$Q(x,a)=0,\ \pi(x,a)=\frac{1}{|A(x)|}$;
$x=x_0,\ a=\pi(x)$;
for $t=1,2,\ldots$ **do**
  $r,x'=$ 在 E 中执行动作 a 产生的奖赏与转移的状态;
  $a'=\pi^{\epsilon}(x')$;
  $Q(x,a)=Q(x,a)+\alpha\big(r+\gamma Q(x',a')-Q(x,a)\big)$;
  $\pi(x)=\arg\max_{a''} Q(x,a'')$;
  $x=x',a=a'$
end for

输出： 策略 π

图 16.12 Sarsa 算法

免模型学习

- Q-学习算法

输入： 环境 E；
动作空间 A；
起始状态 x_0；
奖赏折扣 γ；
更新步长 α.

过程：

$Q(x,a)=0,\ \pi(x,a)=\frac{1}{|A(x)|}$;
$x=x_0$;
for $t=1,2,\ldots$ **do**
  $r,x'=$ 在 E 中执行动作 $\pi^{\epsilon}(x)$ 产生的奖赏与转移的状态;
  $a'=\pi(x')$;
  $Q(x,a)=Q(x,a)+\alpha\big(r+\gamma Q(x',a')-Q(x,a)\big)$;
  $\pi(x)=\arg\max_{a''}Q(x,a'')$;
  $x=x',\ a=a'$
end for

输出： 策略 π

图 16.13　Q-学习算法

用于更新 Q 的下一时刻动作 a' 在下一时刻不一定被执行

免模型学习

策略梯度方法

- 目标：$J(\theta) = \mathrm{E}[R(x,a)] = \sum_x d(x) \sum_a \pi_\theta(x) R(s,a)$

- 目标函数的梯度为：

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \sum_x d(x) \sum_a \nabla_\theta \pi_\theta(x) R(s,a) \\
&= \textstyle\sum_x d(x) \sum_a \pi_\theta(x) \nabla_\theta \log \pi_\theta(x) R(s,a) \\
&= \mathrm{E}[\nabla_\theta \log \pi_\theta(x) R(s,a)]
\end{aligned}
$$

- 此时我们可以通过采样来估计目标函数的梯度，通过更新policy的参数来提升期望reward。

大纲

- 什么是强化学习?
- K-摇臂赌博机
- 有模型学习
- 免模型学习
- 值函数近似
- 一些方向

值函数近似

- 问题：前面都假定状态空间是离散(有限)的，若状态空间是连续(无限)的，该怎么办？

- 连续状态空间所面临的困难
  - 值函数不再是关于状态的“表格值函数”（tabular value function)

- 解决困难的办法：值函数近似
  - 为简便起见，假定状态空间 $X = \mathbb{R}^n$
  - 为简便起见，首先考虑线性近似
  - 假定行为空间有限

值函数近似

值函数近似

- 将值函数表达为状态的线性函数

$$V_{\boldsymbol{\theta}}(\boldsymbol{x}) = \underbrace{\boldsymbol{\theta}}_{\text{参数向量}}^{\top}\underbrace{\boldsymbol{x}}_{\text{状态向量}}$$

- 用最小二乘误差来度量学到的值函数与真实的值函数 V^{π} 之间的近似程度

$$\varepsilon_{\boldsymbol{\theta}} = \mathbb{E}_{\boldsymbol{x}\sim\pi}\left[\left(V^{\pi}(\boldsymbol{x}) - V_{\boldsymbol{\theta}}(\boldsymbol{x})\right)^2\right].$$

- 用梯度下降法更新参数向量，求解优化问题

值函数近似

- 状态-动作值函数的线性近似

$$Q_{\boldsymbol{\theta}}(\boldsymbol{x}, a) = \boldsymbol{\theta}^{\top}(\boldsymbol{x}, \boldsymbol{a})$$

$$\boldsymbol{a} = (0, \ldots, 1, \ldots, 0)$$

- 非线性值函数近似
  - 核方法
  - 神经网络

大纲

- 什么是强化学习？
- K-摇臂赌博机
- 有模型学习
- 免模型学习
- 值函数近似
- 一些方向

模仿学习

- 强化学习任务中多步决策的搜索空间巨大，基于累积奖赏来学习很多步之前的合适决策非常困难

- 缓解方法：直接模仿人类专家的状态-动作对来学习策略
  - 相当于告诉机器在什么状态下应该选择什么动作
  - 引入了监督信息来学习策略

$$\boxed{\text{状态}} \xrightarrow{\text{策略 } \pi} \boxed{\text{动作}}$$

- 直接模仿学习

模仿学习

直接模仿学习

- 利用专家的决策轨迹，构造数据集 D：状态作为特征，动作作为标记
- 利用数据集 D，使用分类/回归算法即可学得策略
- 将学得的策略作为初始策略
- 策略改进，从而获得更好的策略

人类专家决策轨迹数据：

$$\{\tau_1, \tau_2, \ldots, \tau_m\},$$

$$\tau_i = < s_1^i, a_1^i, s_2^i, a_2^i, \ldots, s_{n_i+1}^i >$$

构造出的“有标记”数据集：

$$D = \{(s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n)\}$$

模仿学习

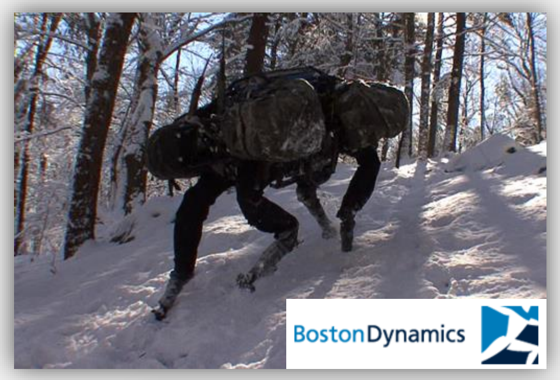
- 强化学习任务中，设计合理的符合应用场景的奖赏函数往往相当困难
- 缓解方法：从人类专家提供的范例数据中反推出奖赏函数
- 逆强化学习 (inverse reinforcement learning)
  - 基本思想：寻找某种奖赏函数使得范例数据是最优的，然后即可使用这个奖赏函数来训练强化学习策略

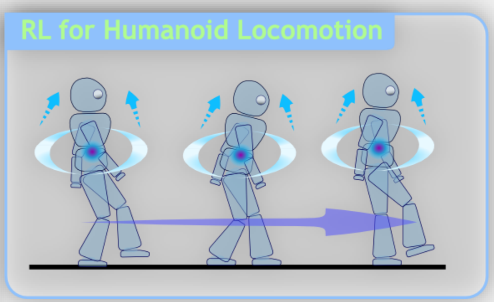
迭代式逆强化学习算法

```
输入：环境 E;
      状态空间 X;
      动作空间 A;
      范例轨迹数据集 D = {τ_1, τ_2, ..., τ_m}.
过程：
  x̄* = 从范例轨迹中算出状态加权和的均值向量;
  π = 随机策略;
  for t = 1, 2, ... do
      x̄_t^π = 从 π 的采样轨迹算出状态加权和的均值向量;
      求解 w* = arg max_w min_{i=1}^t w^T (x̄* − x̄_i^π)   s.t. ||w|| ≤ 1;
      π = 在环境 ⟨X, A, R(x) = w*^T x⟩ 中求解最优策略;
  end for
输出：奖赏函数 R(x) = w*^T x 与策略 π
```

图 16.15　迭代式逆强化学习算法

强化学习的应用


BostonDynamics


RL for Humanoid Locomotion


HI-SCORE
10000
00
READY!


Backgammon
AI
Noname
© 2003-2009 Odesys

小结

- 强化学习：多步决策过程
- 有模型学习
  - 基于动态规划的寻优
- 如何处理环境中的未知因素
  - 蒙特卡罗强化学习
  - 时序差分学习
- 如何处理连续状态空间
  - 值函数近似
- 如何提速强化学习过程
  - 直接模仿学习
  - 逆强化学习