

## 强化学习

机器处于环境 E 中，状态空间为  $X$ ，其中每个状态  $x \in X$  是机器感受环境的描述；



若某个动作  $a \in A$  作用于当前状态  $x$  上，则潜在的转移函数  $P$  将使得环境从当前状态按某种概率转移到另一个状态，同时根据潜在的奖赏  $R$  反馈给机器一个奖励。

$$\text{四元组: } E = \langle X, A, P, R \rangle$$

$P: X \times A \times A \mapsto R$  指定 状态转移概率。

$R: X \times A \times X \mapsto R$  指定 奖赏

若  $R$  仅与 状态转移有关者,  $R: X \times X \mapsto R$ .

机器要做的就是通过在环境中不断尝试而学到一个策略  $\pi$ ，根据当前状态输入 动作  $a = \pi(x)$

$$\begin{cases} \text{确定性: } \pi: X \mapsto A \\ \text{随机性: } \pi(x, a), \quad \sum_a \pi(x, a) = 1 \end{cases}$$

$$\begin{cases} \text{学习的目的: 最大化 累计奖赏} \\ \text{1 步骤奖赏} \quad E \left[ \frac{1}{T} \sum_{t=1}^T r_t \right] \\ \text{Y 步加权奖赏} \quad E \left[ \sum_{t=0}^{T-1} \gamma^t r_{t+1} \right] \end{cases}$$

### 1. 奖赏与利用

最大简单步奖赏: 若每个动作对应的奖赏是一个确定值, 那么尝试一遍所有动作便能找出奖赏最大的动作  
概率分布尝试一次并不能精确获得平均奖赏值。

权概率: 仅有某些臂的期望奖赏, 将所有机会平均分配, 以每个摇臂平均吐币率作为准确期望的近似估计

估计概率分布

权利用: 仅为执行奖赏最大的动作, 摆下目前为止平均奖赏最大的摇臂 (多个最优, 随机选一)  
摇臂当前最佳奖赏

(1) ε-贪心: ε 的概率, 均匀概率 跟随选择手数变化

$\begin{cases} \text{1-ε 的概率, 选择当前平均奖赏最高的摇臂.} \\ \text{ε 的概率, 探索更多奖赏, 选择 \epsilon} \end{cases}$

$$\begin{aligned} \text{平均奖赏: } Q(k) &= \frac{1}{n} \sum_{i=1}^n v_i & \xrightarrow{\text{减少记录}} & Q(k) = \frac{1}{n} (n-1) Q_{n-1}(k) + v_n \\ &= Q_{n-1}(k) + \frac{1}{n} (v_n - Q_{n-1}(k)) \end{aligned}$$

根据概率分布, 调整  $\epsilon$   $\begin{cases} \text{高: 需要更多探索, 降低 \epsilon} \\ \text{低: 选择 \epsilon} \end{cases}$

使用中:  $\epsilon$  可随尝试次数增加而逐渐减少

```

输入: 摆臂数 K;
      奖赏函数 R;
      尝试次数 T;
      探索概率 \epsilon.

过程:
1: r = 0;
2: \forall i = 1, 2, \dots, K : Q(i) = 0, count(i) = 0;
3: for t = 1, 2, \dots, T do
4:   if rand() < \epsilon then
5:     k = 从 1, 2, \dots, K 中以均匀分布随机选取
6:   else
7:     k = arg max_i Q(i)
8:   end if
9:   v = R(k);
10:  r = r + v;
11:  Q(k) = \frac{Q(k) \times count(k) + v}{count(k) + 1};
12:  count(k) = count(k) + 1;
13: end for
输出: 累积奖赏 r
  
```

(2) Softmax: 某些摇臂的平均奖赏明显高于其它, 则它们选取概率明显更高。

$$\begin{aligned} \text{摇臂概率分布基于 Boltzmann 分布: } P(k) &= \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^S e^{\frac{Q(i)}{\tau}}} & \text{其中 } Q(i) \text{ 为前摇臂平均奖赏} \\ & \tau 越小则平均奖赏高的摇臂概率越高, \tau \rightarrow 0 \text{ 时 贪婪利用} \\ & \tau \rightarrow \infty \text{ 时 随机探索} \end{aligned}$$

窮盡状态空间、窮盡动作空间:

将每个状态上的动作连起来形成决策树图，但是有考虑不可达决策过程。

```

输入: 摆臂数 K;
      奖赏函数 R;
      尝试次数 T;
      温度参数 \tau.

过程:
1: r = 0;
2: \forall i = 1, 2, \dots, K : Q(i) = 0, count(i) = 0;
3: for t = 1, 2, \dots, T do
4:   k = 从 1, 2, \dots, K 中根据式(16.4)随机选取
5:   v = R(k);
6:   r = r + v;
7:   Q(k) = \frac{Q(k) \times count(k) + v}{count(k) + 1};
8:   count(k) = count(k) + 1;
9: end for
输出: 累积奖赏 r
  
```

## 2. 有模型学习

$E = \langle X, A, P, R \rangle$  均已知, 此时对任意状态  $x, x'$  和动作  $a$ , 在  $t$  状态下执行动作  $a$  转移到  $x'$  的概率  $P_{x \rightarrow x'}^a$  已知.

该奖励带来的奖赏  $R_{x \rightarrow x'}^a$  也是已知的.

$V_t^*(x)$ : 从状态  $x$ , 使用策略  $\pi$  带来的累积奖赏.  $\rightarrow$  状态值函数

$Q_t^*(x, a)$ : 从状态  $x$ , 执行动作  $a$  再使用策略  $\pi$  的奖赏  $\rightarrow$  状态-动作值函数.

$$\begin{cases} V_t^*(x) = E_x [\frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x], & T \text{ 步累积奖赏} \\ V_t^*(x) = E_x [\sum_{t=0}^{+\infty} r_t^+ \mid x_0 = x], & T \text{ 折扣累积奖赏} \end{cases}$$

$$\begin{cases} Q_t^*(x, a) = E_x [\frac{1}{T} \sum_{t=1}^T r_t \mid x_0 = x, a_0 = a] \\ Q_t^*(x, a) = E_x [\sum_{t=0}^{+\infty} r_t^+ \mid x_0 = x, a_0 = a] \end{cases}$$

输入: MDP 四元组  $E = \langle X, A, P, R \rangle$ ;  
被评估的策略  $\pi$ ;

累积奖赏参数  $T$ .

过程:

```

1:  $\forall x \in X : V(x) = 0;$ 
2: for  $t = 1, 2, \dots$  do
3:    $\forall x \in X : V'(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a (\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V(x'));$ 
4:   if  $t = T + 1$  then
5:     break
6:   else
7:      $V = V'$ 
8:   end if
9: end for
输出: 状态值函数  $V$ 
```

MDP 具有马尔可夫性质, 系统下一时刻的状态仅由当前状态的决策决定.

$$\begin{aligned} V_t^*(x) &= E_x \left[ \frac{1}{T} r_t + \frac{T-1}{T} \sum_{t=2}^T r_t \mid x_0 = x \right] \\ &= \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a \left( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{t-1}^*(x') \right) \end{aligned}$$

$$\text{同理: } V_t^*(x) = \sum_{a \in A} \pi(x, a) \sum_{x' \in X} P_{x \rightarrow x'}^a (R_{x \rightarrow x'}^a + r V_t^*(x'))$$

$$\text{停止准则: } \max_{x \in X} |V(x) - V(x')| < \theta$$

对  $V_t^*(x)$  作以下变动

$$\begin{cases} Q_t^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a ( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{t-1}^*(x') ) \\ Q_t^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a ( R_{x \rightarrow x'}^a + r V_t^*(x') ). \end{cases}$$

$$\begin{cases} V_t^*(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a ( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_{t-1}^*(x') ) \\ V_t^*(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a ( R_{x \rightarrow x'}^a + r V_t^*(x') ) \end{cases}$$

下面进行策略改进. 目标  $\pi^* = \arg \max_{\pi} \sum_{x \in X} V_t^*(x)$

$$\Rightarrow V_t^*(x) = \max_{a \in A} Q_t^*(x, a)$$

$$\Rightarrow \begin{cases} Q_t^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a ( \frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} \max_{a' \in A} Q_{t-1}^*(x', a') ) \\ Q_t^*(x, a) = \sum_{x' \in X} P_{x \rightarrow x'}^a ( R_{x \rightarrow x'}^a + r \max_{a' \in A} Q_t^*(x', a') ) \end{cases}$$

最优 Bellman 等式: 将策略选择的动作改变为当前最优的动作  $\pi^*$ .  $Q_t^*(x, \pi^*(x)) \geq V_t^*(x)$

值函数对策略的每一点改进都是单调递增的.  $\pi^{t+1} = \arg \max_{a \in A} Q_t^*(x, a)$  直到  $\pi^*$  与  $\pi^{t+1}$  一致, 不再变化即停止.

输入: MDP 四元组  $E = \langle X, A, P, R \rangle$ ;  
累积奖赏参数  $T$ .  
过程:  
1:  $\forall x \in X : V(x) = 0, \pi(x, a) = \frac{1}{|A(x)|};$   
2: loop  
3: for  $t = 1, 2, \dots$  do  
4:  $\forall x \in X : V'(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V(x'));$   
5: if  $t = T + 1$  then  
6: break  
7: else  
8:  $V = V'$   
9: end if  
10: end for  
11:  $\forall x \in X : \pi'(x) = \arg \max_{a \in A} Q(x, a);$   
12: if  $\forall x : \pi'(x) = \pi(x)$  then  
13: break  
14: else  
15:  $\pi = \pi'$   
16: end if  
17: end loop  
输出: 最优策略  $\pi$

策略迭代法在每次改进后需进行重新评估

改进价值函数:

$$\begin{cases} V_t(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V_t(x')) \\ V_t(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a ( R_{x \rightarrow x'}^a + r V_t(x') ) \end{cases}$$

图 16.8 基于  $T$  步累积奖赏的策略迭代算法

输入: MDP 四元组  $E = \langle X, A, P, R \rangle$ ;  
累积奖赏参数  $T$ ;  
收敛阈值  $\theta$ .  
过程:  
1:  $\forall x \in X : V(x) = 0;$   
2: for  $t = 1, 2, \dots$  do  
3:  $\forall x \in X : V'(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a (\frac{1}{T} R_{x \rightarrow x'}^a + \frac{T-1}{T} V(x'));$   
4: if  $\max_{x \in X} |V(x) - V'(x)| < \theta$  then  
5: break  
6: else  
7:  $V = V'$   
8: end if  
9: end for  
输出: 策略  $\pi(x) = \arg \max_{a \in A} Q(x, a)$

图 16.9 基于  $T$  步累积奖赏的值迭代算法

$$\forall x \in X : V(x) = \max_{a \in A} \sum_{x' \in X} P_{x \rightarrow x'}^a ( R_{x \rightarrow x'}^a + r V(x') )$$

### 3. 免模型学习：学习算法不依赖于环境建模。

#### (1) 蒙特卡罗 强化学习

策略评估：多次采样，求取平均累积奖赏作为期望累积奖赏近似。

采样有限性 使其更适用于 T 步蒙特卡罗。

在模型未知的情况下，从起始状态出发，使用某种策略进行采样，执行该策略 T 步并获得轨迹

$\langle x_0, a_0, r_1, x_1, a_1, r_2, \dots, x_{T-1}, a_{T-1}, r_T, x_T \rangle$ 。对轨迹中出现的每一对状态-动作，记录其后的奖赏之和，作为该状态-动作对的一次累积奖赏采样值。多次采样得到多条轨迹后，将每个状态-动作对的累积奖赏采样值进行平均，得到状态-动作值函数的估计。

为了得到多条不同的采样轨迹：采用  $\epsilon$ -贪心法

$$\left\{ \begin{array}{l} \epsilon: \text{从所有动作中随机挑选一个} \\ 1-\epsilon: \text{当前最优动作} \end{array} \right.$$

其中 最优动作被选中的概率为  $1 - \epsilon + \frac{\epsilon}{|A|}$ ，每个非最优动作被选中的概率是  $\frac{\epsilon}{|A|}$

用策略：评估，最蓝同策略。

改进  $\Rightarrow$  策略评估：引入  $\epsilon$ -贪心  
策略改进：改进原始策略。

增量计算：每采样出一条轨迹，对值函数进行更新。

```
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, count(x, a) = 0, π(x, a) = 1/|A(x)|;
2: for s = 1, 2, ..., 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, ..., x_{T-1}, a_{T-1}, r_T, x_T >;
4:   for t = 0, 1, ..., T-1 do
5:     R = 1/(T-t) * sum_{i=t+1}^T r_i;
6:     Q(x_t, a_t) = Q(x_t, a_t) * count(x_t, a_t) + R;
7:     count(x_t, a_t) = count(x_t, a_t) + 1;
8:   end for
9:   对所有已见状态 x:
      π(x) = arg max_{a'} Q(x, a'),           以概率 1 - ε;
      π(x) = {均匀概率从 A 中选取动作}, 以概率 ε.
10: end for
输出: 策略 π
```

图 16.10 同策略蒙特卡罗强化学习算法

改进  $\Rightarrow$  策略评估：引入  $\epsilon$ -贪心  
策略改进：改进原始策略。

函数子在分布 P 下的期望  $E_P f = \int_x p(x) f(x) dx$   
 从 P 上采样估计 f 的期望  $\hat{E}_P f = \frac{1}{m} \sum_{i=1}^m f(x_i)$   
 引入另一个分布  $E_{P'} f = \int_{x'} q(x') \frac{p(x)}{q(x)} f(x') dx'$   
 值函数  $\frac{p(x)}{q(x)}$  f(x) 在 P' 下的期望，在 Q 上采样  $\{x'_1, x'_2, \dots, x'_m\}$   
 $\hat{E}_{P'} f = \frac{1}{m} \sum_{i=1}^m \frac{p(x'_i)}{q(x'_i)} f(x'_i)$

用元：

$$Q(x, a) = \frac{1}{m} \sum_{i=1}^m \frac{p_i(x)}{p_i^{(x)}} R_i$$
 其中  $R_i$  为第 i 条轨迹 从开始至结束的奖励。  
 $p_i^{(x)}$  和  $p_i^{(x')}$  分别表示两个策略产生第 i 条轨迹的概率。

```
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, count(x, a) = 0, π(x, a) = 1/|A(x)|;
2: for s = 1, 2, ..., do
3:   在 E 中执行 π 的  $\epsilon$ -贪心策略产生轨迹
      < x_0, a_0, r_1, x_1, a_1, r_2, ..., x_{T-1}, a_{T-1}, r_T, x_T >;
4:   p_i = {1 - ε + ε/|A|, a_i = π(x_i); ε/|A|, a_i ≠ π(x_i)};
5:   for t = 0, 1, ..., T-1 do
6:     R = 1/(T-t) * sum_{i=t+1}^T r_i;
7:     Q(x_t, a_t) = Q(x_t, a_t) * count(x_t, a_t) + R;
8:     count(x_t, a_t) = count(x_t, a_t) + 1;
9:   end for
10:  π(x) = arg max_{a'} Q(x, a')
11: end for
输出: 策略 π
```

图 16.11 异策略蒙特卡罗强化学习算法

蒙特卡罗 强化学习法：重算一个轨迹后再更新

$\epsilon$ -贪心，softmax：基于动态规划的策略估计和值迭代，执行一步即更新值函数

顺序差分学习：结合蒙特卡罗与动态规划方法。

蒙特卡罗：多次尝试后求平均作为期望累积奖励的近似，但未平均使用批处理

$\Rightarrow$  增量式：已知状态-动作对  $(x, a)$ ，基于  $t+1$  采样的值函数  $Q_t^{\pi}(x, a) = \frac{1}{t} \sum_{i=1}^t r_i$

$$t+1 \text{ 个采样 } r_{t+1}, Q_{t+1}^{\pi}(x, a) = Q_t^{\pi}(x, a) + \frac{1}{t+1} (r_{t+1} - Q_t^{\pi}(x, a))$$

每步累积量加  
可替换为  $a_{t+1}$ ， $\alpha$  为较小的正数

和为 1，更新步长以避开之后的累积量越重要。

从  $\gamma$  折扣累积量  $Q_{t+1}^{\pi}(x, a) = Q_t^{\pi}(x, a) + \alpha (R_{x \rightarrow x'}^a + \gamma Q_t^{\pi}(x', a') - Q_t^{\pi}(x, a))$

其中  $x'$  是前一次在状态  $x$  执行动作  $a$  后的状态， $a'$  是策略  $\pi$  在  $x'$  上选择的动作。

Sarsa：同蒙特卡罗法，更新值函数需知道

前一步状态	其中评估和执行均为 $\epsilon$ -贪心策略。
前一步动作	
增量值	
当前状态	
将要执行的动作	

Q-学习：将 Sarsa 重新为异策略算法。  
评估：原始  
执行： $\epsilon$ -贪心

输入：环境  $E$ ；  
动作空间  $A$ ；  
起始状态  $x_0$ ；  
奖赏折扣  $\gamma$ ；  
更新步长  $\alpha$ 。  
过程：  
1:  $Q(x, a) = 0, \pi(x, a) = \frac{1}{|A(x)|}$ ;  
2:  $x = x_0, a = \pi(x)$ ;  
3: for  $t = 1, 2, \dots$  do  
4:    $r, x' =$  在  $E$  中执行动作  $a$  产生的奖赏与转移的状态;  
5:    $a' = \pi'(x')$ ;  
6:    $Q(x, a) = Q(x, a) + \alpha(r + \gamma Q(x', a') - Q(x, a))$ ;  
7:    $\pi(x) = \arg \max_{a''} Q(x, a'')$ ;  
8:    $x = x', a = a'$ ;  
9: end for  
输出：策略  $\pi$

图 16.12 Sarsa 算法

输入：环境  $E$ ；  
动作空间  $A$ ；  
起始状态  $x_0$ ；  
奖赏折扣  $\gamma$ ；  
更新步长  $\alpha$ 。  
过程：  
1:  $Q(x, a) = 0, \pi(x, a) = \frac{1}{|A(x)|}$ ;  
2:  $x = x_0$ ;  
3: for  $t = 1, 2, \dots$  do  
4:    $r, x' =$  在  $E$  中执行动作  $a = \pi^e(x)$  产生的奖赏与转移的状态;  
5:    $a' = \pi(x')$ ;  
6:    $Q(x, a) = Q(x, a) + \alpha(r + \gamma Q(x', a') - Q(x, a))$ ;  
7:    $\pi(x) = \arg \max_{a''} Q(x, a'')$ ;  
8:    $x = x'$ ;  
9: end for  
输出：策略  $\pi$

图 16.13 Q-学习算法

之前的假设：学习在有限状态空间进行。每个状态可用一个编号指代

值函数可表示为一个数组，索引  $i$  对应函数值为数组  $i$  的值，且更新一个状态上的值不会影响其它状态上的值。

现实世界：状态连贯且无穷多。

假设状态空间  $x \in \mathbb{R}^n$  ( $n$  维实数空间)，值函数表达为状态的线性函数。

$V_\theta(x) = \theta^\top x$  其中  $x$  为状态向量， $\theta$  为参数向量，这种方法又称为值函数近似。

近似程度  $J_\theta = E_{\pi, x} [(V_\theta(x) - V_\theta(x))^2]$  其中  $E_{\pi, x}$  表示由策略  $\pi$  采样得状态的期望。

梯度下降 最小化误差  $-\frac{\partial J_\theta}{\partial \theta} = E_{\pi, x} [2(V_\theta(x) - V_\theta(x)) \frac{\partial V_\theta(x)}{\partial \theta}]$

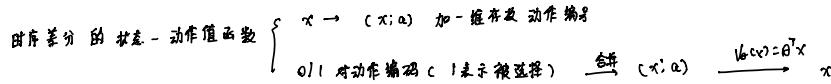
$$= E_{\pi, x} [2(V_\theta(x) - V_\theta(x)) x]$$

$$\theta = \theta + \alpha (r + \gamma V_\theta(x) - V_\theta(x)) x$$

$$\Rightarrow \theta = \theta + \alpha (r + \gamma V_\theta(x) - V_\theta(x)) x$$

$$= \theta + \alpha (r + \gamma \theta^\top x - \theta^\top x) x$$

由于无法知道  $V$ ，利用时序差分  $V^T(x) = r + \gamma V^T(x')$  估计。



基于线性值函数近似来替代 Sarsa 算法中的值函数，即可得到图 16.14 的线性值函数近似 Sarsa 算法。类似地可得到线性值函数近似 Q-学习算法。显然，可以容易地用其他学习方法来代替式(16.32)中的线性学习器，例如通过引入核方法实现非线性值函数近似。

```

输入: 环境 E;
       动作空间 A;
       起始状态  $x_0$ ;
       奖赏折扣  $\gamma$ ;
       更新步长  $\alpha$ .
过程:
1:  $\theta = 0$ ;
2:  $x = x_0$ ,  $a = \pi(x) = \arg \max_{a'} \theta^T(x; a')$ ;
3: for  $t = 1, 2, \dots$  do
4:    $r, x' =$  在  $E$  中执行动作  $a$  产生的奖赏与转移的状态;
5:    $a' = \pi'(x')$ ;
6:    $\theta = \theta + \alpha(r + \gamma \theta^T(x'; a') - \theta^T(x; a))(x; a)$ ;
7:    $\pi(x) = \arg \max_{a'} \theta^T(x; a')$ ;
8:    $x = x'$ ,  $a = a'$ ;
9: end for
输出: 策略  $\pi$ 

```

图 16.14 线性值函数近似 Sarsa 算法

#### 4. 模仿学习：从范例中学习

① 直接模仿：已知人类专家决策轨迹  $\{T_1, T_2, \dots, T_n\}$

每条轨迹包含状态和动作序列  $T_i = \langle s_i^1, a_i^1, s_i^2, a_i^2, \dots, s_{n_i+1}^i \rangle$ ，其中  $n_i$  为第  $i$  条轨迹中的转移次数。

新的数据集： $D = \{(s_1, a_1), (s_2, a_2), \dots, (s_{n_i+1}, a_{n_i+1})\}$

状态作为特征，动作作为标记，可用分类或回归

作为初始策略，再用强化学习方法进行改进。

#### ② 迭代强化学习：从范例数据反推策略函数。

已知状态空间  $X$ ，动作空间  $A$ ，决策轨迹数据集  $\{T_1, T_2, \dots, T_n\}$

假设机器结果与范例一致， $\Leftrightarrow$  在某个奖赏函数中找最优策略，轨迹与范例一致

$\Rightarrow$  找出奖赏函数使范例最优，用它来训练策略。

$$R(x) = w^T x, \text{ 假设元的奖励模型为 } \rho^* = E \left[ \sum_{t=0}^{+\infty} \gamma^t R(x_t) | x \right] = E \left[ \sum_{t=0}^{+\infty} \gamma^t w^T x_t | x \right]$$

$$\begin{aligned}
 &= w^T E \left[ \sum_{t=0}^{+\infty} \gamma^t x_t | x \right] \\
 &\quad \underbrace{\qquad\qquad\qquad}_{\text{状态向量加权和}} \\
 &= w^T \bar{x}
 \end{aligned}$$

显然，我们难以获得所有策略，一个较好的办法是从随机策略开始，迭代地求解更好的奖赏函数，基于奖赏函数获得更好的策略，直至最终获得最符合范例轨迹数据集的奖赏函数和策略，如图 16.15 算法所示。注意在求解更好的奖赏函数时，需将式(16.39)中对所有策略求最小改为对之前学得的策略求最小。

```

输入: 环境 E;
       状态空间 X;
       动作空间 A;
       范例轨迹数据集  $D = \{\tau_1, \tau_2, \dots, \tau_m\}$ .
过程:
1:  $\bar{x}^*$  = 从范例轨迹中算出状态加权和的均值向量;
2:  $\pi =$  随机策略;
3: for  $t = 1, 2, \dots$  do
4:    $\bar{x}_t^*$  = 从  $\pi$  的采样轨迹算出状态加权和的均值向量;
5:   求解  $w^* = \arg \max_w \min_{t=1}^T w^T (\bar{x}^* - \bar{x}_t^*)$  s.t.  $\|w\| \leq 1$ ;
6:    $\pi =$  在环境  $(X, A, R(x) = w^T x)$  中求解最优策略;
7: end for
输出: 奖赏函数  $R(x) = w^{*T} x$  与策略  $\pi$ 

```

图 16.15 迭代式逆强化学习算法

② 使用蒙特卡罗方法通过样本近似期望

范例数据看成最优策略的样本

$\bar{x}^*$ ：每条范例轨迹上的状态加权和再平均

$$W^* = \arg \max_w \min_{\pi} w^T (\bar{x}^* - \bar{x})$$

s.t.  $\|w\| \leq 1$