强化学习

目　录

返回目录

图　片

表　格

算　法

返回目录

要 点

返回目录

返回目录

1 导论

特征 智能体与环境交互（采样），在不断尝试中学习策略，使收益最大化。

- 试错探索：不会获知应采取的行动，通过尝试获得。
- 延迟收益：一个动作的收益可能无法短期体现，而是长期浮现。
- 环境不确定性：当前动作不但会影响当前收益，还会影响后续环境，进而影响后续收益。
- 影响未知性：无法预测动作的影响，需要与环境频繁交互。
- 试探（开拓动作空间）与开发/贪心（根据经验获得收益）折中。

其他优化方法

- 凸优化：状态空间较小，线性规划。
- 最优控制：已知模型，解析回报函数，动态规划。
- 进化算法：控制策略简单。
- 机器学习
  - 有监督学习：有标签，注重推断与泛化能力。
  - 无监督学习：无标签，寻找数据隐含结构。

分类

1. 模型依赖性
   - 有模型：规划。
   - 无模型：试错。
2. 策略更新方法
   - 值函数：求解值函数重构策略。
   - 直接策略搜索：搜索策略空间。
   - Actor-Critic方法：类似于策略迭代，同时逼近值函数和策略。

3. 回报函数是否已知
    - 正向：从回报到策略。
    - 逆向：从专家示例到回报。
4. 任务体量：分层强化学习、元强化学习、多智能体强化学习、迁移学习等

发展

值函数→直接策略搜索→深度强化学习。详细历史见12.1。

与深度学习结合，与专业知识结合，理论分析型增强，与认知科学结合，体量增大，与贝叶斯结合。

2 马尔可夫决策过程与贝尔曼方程

2.1 马尔可夫决策过程（Markov decision process，MDP）

2.1.1 要素

[1]

- 状态（state，S）：强化学习依赖的概念。
- 动作（action，A）：智能体做出的选择。
- 奖励/收益（reward，R）：短期学习目标，环境给予智能体的信号。
- 策略（policy，π）：在特定状态下，动作集的分布$\pi(a|s) = p[A_t = a|S_t = s]$。
- 回报（return，G）：长期收益累计，可能含有折扣，需综合评估。
- 折扣因子（γ）。
- 值函数（value function，V）：一定状态下预估的期望回报。
- 行为/动作值函数（Q）：一定状态-动作对下预估的期望回报。
- 环境模型（P）：模拟环境的反应。

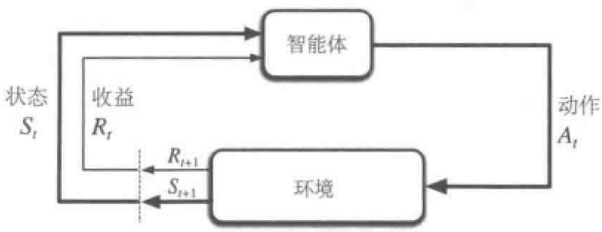


图 1: 马尔可夫决策过程

2.1.2 状态、动作与收益

序贯交互轨迹（TRAJECTORY） $\tau = S_0, A_0, R_1, S_1, A_1, R_2, \ldots$

随机变量s', r服从离散概率分布$p(s', r|s, a) \doteq \Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$，即$S_t, R_t$所有可能组合的概率和为1。

马尔可夫性 [2] 即“无记忆性”，指未来状态仅依赖于当前状态，而独立于过去状态，S_t, R_t只依赖于S_{t-1}, A_{t-1}。

状态转移

当前状态和动作下，转移到某状态的概率，包括该状态下各可能收益情况：

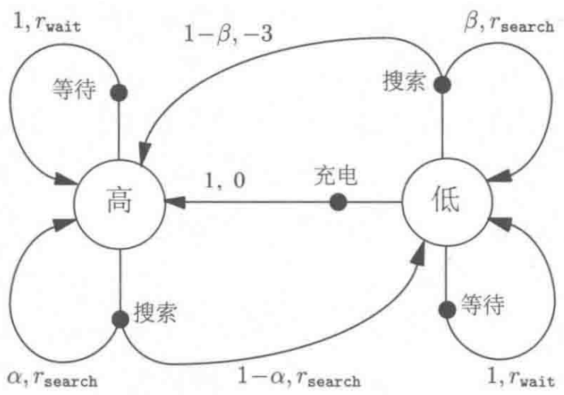
$$p(s'|s, a) \doteq \Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in R} p(s', r|s, a)$$

以下给出了有无指定未来状态的两种期望收益：

$$r(s, a) \doteq E[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r|s, a)$$

$$r(s, a, s') \doteq E[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in R} r \frac{p(s', r|s, a)}{p(s'|s, a)}$$

返回目录



(a) 状态转移图（节点不能重复）

s	a	s'	$p(s' \mid s, a)$	$r(s, a, s')$
high	search	high	α	$r_{\texttt{search}}$
high	search	low	$1-\alpha$	$r_{\texttt{search}}$
low	search	high	$1-\beta$	-3
low	search	low	β	$r_{\texttt{search}}$
high	wait	high	1	$r_{\texttt{wait}}$
high	wait	low	0	$r_{\texttt{wait}}$
low	wait	high	0	$r_{\texttt{wait}}$
low	wait	low	1	$r_{\texttt{wait}}$
low	recharge	high	1	0
low	recharge	low	0	0

(b) 状态转移表

图 2: 回收机器人状态转移

2.1.3 策略

贪婪策略 $\pi(a|s) = \mathrm{argmax}_a q(a)$。

探索-利用平衡策略

- ϵ-greedy策略 [3] ：

$$a = \begin{cases} \mathrm{argmax}_a q(a) & ,p = 1-\epsilon \\ \mathrm{random}(a) & ,p = \epsilon \end{cases} \Rightarrow \pi(a|s) = \begin{cases} 1-\epsilon+\frac{\epsilon}{|A|} & ,a = \mathrm{argmax}_a q(a) \\ \frac{\epsilon}{|A|} & ,\mathrm{otherwise} \end{cases}$$

  靠近贪心策略，但所有动作概率不为零。在实际使用时，需要注意多最优情况。

- UCB（upper confidence bound）策略：

$$\pi(a|s) = Q(a) + c\sqrt{\frac{\ln N}{n(a)}}$$

  其中，c控制探索强度，N是当前轮数，$n(a)$是a被选次数。

  可以自适应平衡探索与利用。

- 玻尔兹曼分布（Boltzmann）：

$$\pi(a|s) = \frac{e^{Q(a)/\tau}}{\sum_{a'} e^{Q(a)/\tau}}$$

返回目录

其中τ是温度参数，控制随机性程度，趋于0时接近贪心策略，趋于∞时接近均匀随机选择。

可以动态调整探索强度。

- 高斯策略：

$$\pi = \mu + \epsilon, \epsilon \sim N(0, \sigma^2)$$

增量式更新 [4] 将轮次更新的量化为递推关系，减少空间复杂度，如运行均值：

$$Q_{n+1} = \frac{1}{n}\sum_{i=1}^{n} R_i = Q_n + \frac{1}{n}(R_n - Q_n)$$

2.1.4 回报与折扣

[5]

- 幕（episode）：一次交互序列。
- 分幕式任务：具有分幕重复特性，下一幕开始状态与上一幕终结状态无关。

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T$$

其中T是最终时刻，由其划分非终结状态集S和所有状态集S^+。

- 持续性任务：持续不断发生，不能自然分幕，最终时刻趋于无穷。

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \leqslant \frac{1}{1-\gamma} maxR_t$$

其中，折扣率$\gamma \in [0, 1]$越大代表长期收益越重要。

- 统一表示：有限项终止后，状态持续转移回自己，相当于无限项。

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

2.1.5 值函数

[6]

返回目录

值函数

$$
\begin{aligned}
v_\pi(s) &\doteq E_\pi[G_t|S_t=s] = E_\pi[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s], s\in S \\
&= E_\pi[R_{t+1}+\gamma G_{t+1}|S_t=s](\text{后继递推关系}) \\
&= \sum_a \pi(a|s)\sum_{s'}\sum_r p(s',r|s,a)\{r+\gamma E_\pi[G_{t+1}|S_{t+1}=s']\}(\text{全概率条件期望展开}) \\
&= \underbrace{\sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)[r+\gamma v_\pi(s')]}_{\text{贝尔曼方程}}
\end{aligned}
$$

行为值函数

$$
\begin{aligned}
q_\pi(s,a) &\doteq E_\pi[G_t|S_t=s,A_t=a] = E_\pi[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s,A_t=a], s\in S \\
&= R(a|s)+\gamma\sum_{s'\in S}P(a|ss')\sum_{a'\in A}\pi(a'|s')q_\pi(s'|a')
\end{aligned}
$$

其与值函数有转化关系：

$$
v_\pi(s) = \sum_{a\in A}\pi(a|s)q_\pi(s,a)
$$

回溯算法

后继状态的价值信息回传给当前状态。

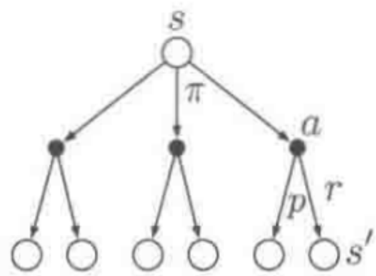


图 3: DP回溯图（节点可以重复）

最优值函数

$\forall s \in S, q_\pi(s, \pi'(s)) = v_{\pi'}(s) \geqslant v_\pi(s)$，则称$\pi'$优于或等于$\pi$。值函数定义了策略的偏序关系，最优策略存在且可能不唯一，它们共享最优值函数：

$$v^*(s) \doteq \max_\pi v_\pi(s)$$
$$q^*(s, a) \doteq \max_\pi q_\pi(s, a)$$

值函数最优和策略最优等价。

2.1.6 构建要点

- 确定动作、状态、收益（不含先验知识，不为达到子目标而舍弃最终目标）。
- 奖励与惩罚：相对的，可以全奖励或全惩罚。
- 同一问题可能有多层次MDP。
- 不同状态的可行动作设置：利用先验知识，人为排除愚蠢动作。

2.2 贝尔曼方程

7

2.2.1 贝尔曼方程

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')]$$

可化简为$v = r_\pi + \gamma P_\pi v'$，其说明一个状态依赖其他状态值。

2.2.2 贝尔曼最优方程

方程组中方程数对应状态数，如环境模型P已知，并具有马尔可夫性，则可求解。但一般难以满足，且计算资源有限，求近似解。

形式

- 同一状态，最优动作：转移收益一定，递推最优值函数

$$
\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\
&= \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\
&= \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')]
\end{aligned}
$$

- 统一状态-动作对，最优下一状态-动作对：

$$
\begin{aligned}
q_*(s,a) &= E[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a] \\
&= \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')]
\end{aligned}
$$

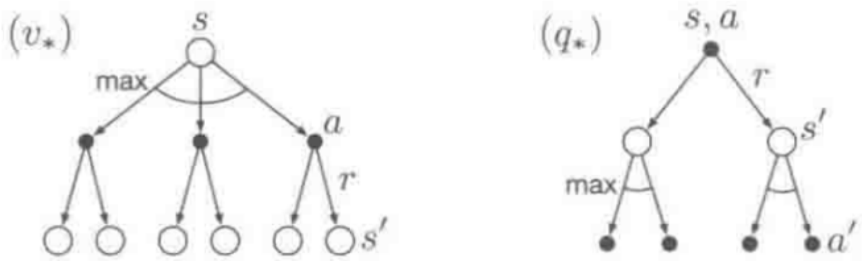


图 4: DP回溯图的两种形式（最优）

描述方式

- 元素：$v(s) = \max_\pi \sum_{s \in S} \pi(a|s) q(s,a)$。
- 矩阵向量：$v = \max_\pi (r_\pi + \gamma P_\pi v)$。

求解　伸缩映射性，见12.2。

贪婪最优策略　最优策略下，各状态价值一定等于其下最优动作的期望回报，可使用贪心策略求取（证明：凸组合最大值为最大一项）。

3 动态规划（DYNAMIC PROGRAMMING，DP）：期望更新

使用值函数结构化组织最优策略搜索，将贝尔曼方程转化成近似逼近理想值函数的递归更新公式，即将多阶段决策问题转化为多个单阶段决策问题。

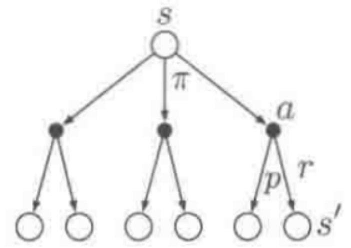


图 5: DP回溯图：显示一步的所有转移

3.1 策略迭代

[8] 反复进行策略评估和策略迭代，得到改进的值函数估计和策略，最后收敛到最优，收敛较快。

策略评估（PE） 计算v_{π_k}

- 直接求解：$v_{\pi_k} = (I - \gamma P_{\pi_k})^{-1} r_{\pi_k}$。
- 迭代求解：$v_{\pi_k}^{(j+1)} = r_{\pi_k} + \gamma P_{\pi_k} v_{\pi_k}^{(j)}, j = 0, 1, 2, \ldots$。

1. 期望更新：基于后继可能状态的期望值。
2. 截断策略评估：不需要完全收敛。

策略改进（PI） 根据原策略的值函数，利用贪心方法构造新策略，其一定不差于原策略。对于确定性策略和随机策略都成立。

$$\pi_{k+1} = \mathrm{argmax}_{\pi}(r_\pi + \gamma P_\pi v_{\pi_k})$$

算法 1：策略迭代

1: 参数：阈值$\theta > 0$确定估计精度
2: 初始化：$\forall s \in S$，任意初始化$v(s) \in R, \pi(s) \in A(s)$
3: 循环

返回目录

算法 1：策略迭代

循环 ▷ 策略评估
  $\Delta \leftarrow 0$
  对于 $\forall s \in S$ 执行
    $v \leftarrow v_{\pi_k}(s)$
    $v_{\pi_k}^{(j+1)}(s) \leftarrow \sum_a \pi_k(a|s)[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v_{\pi_k}^{(j)}(s')]$
    $\Delta \leftarrow \max(\Delta, |v - v_{\pi_k}^{(j+1)}(s)|)$
直到 $\Delta < \theta$
策略稳定←true ▷ 策略改进
对于 $\forall s \in S$ 执行
  $a_{old} \leftarrow \pi(s)$
  对于 $\forall a \in A(s)$ 执行
    $q_{\pi_k}(s,a) \leftarrow \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a) v_{\pi_k}(s')$
  $\pi(s) \leftarrow \text{argmax}_a q_{\pi_k}(s,a)$
  如果 $a_{old} \neq \pi(s)$ 那么
    策略稳定←false
直到 策略稳定

3.2 值迭代

9 只进行一次策略评估遍历，对每个状态更新一次，结合策略改进和极端策略评估。更新公式如下：

$$v_{k+1}(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$$

策略更新（PU） $\pi_{k+1} = \text{argmax}_\pi(r_\pi + \gamma P_\pi v_k)$，贪婪选取$a_k^*(s) = \text{argmax}_a q_k(a,s)$。

价值更新（VU） $v_{k+1} = r_{\pi_{k+1}} + \gamma P_{\pi_{k+1}} v_k = \max_a q_k(a,s)$。

算法 2：值迭代

参数：阈值$\theta > 0$确定估计精度
初始化：$\forall s \in S^+$，任意初始化$v(s)$，其中$v(\text{终止}) = 0$

算法 2：值迭代

循环
  $\Delta \leftarrow 0$
  对于 $\forall s \in S$ 执行
    $v \leftarrow v_k(s)$
    对于 $\forall a \in A(s)$ 执行
      $q_k(s,a) \leftarrow \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$
    $a_k^*(s) \leftarrow \mathrm{argmax}_a q_k(s,a)$
    $v_{k+1}(s) \leftarrow \max_a q_k(s,a)$
    若$a = a_k^*$且$\pi_{k+1}(a|s) = 0$，则令$\pi_{k+1}(a|s) = 1$
    $\Delta \leftarrow \max(\Delta, |v_{k+1}(s) - v|)$
直到 $\Delta < \theta$
return 策略 $\pi(s) = \mathrm{argmax}_a \sum_{s',r} p(s',r|s,a)[r + \gamma v(s')]$

3.3 对比

- 值迭代：只需要维护值函数，保证收敛到全局最优策略，尤其适用于复杂策略空间，但每次迭代需遍历所有动作，计算成本较高。
- 策略迭代：需要同时维护值函数和策略，通常比值迭代更快收敛，尤其在策略空间较小时，但依赖初始策略质量，可能陷入局部最优。

3.4 其他内容

异步动态规划 使用任意可用状态值，以任意顺序更新，避免遍历更新，减小计算量。

广义策略迭代（GPI） 策略评估和策略改进以更细粒度进行交替，可视为竞争与合作。

动态规划的效率 动态规划的时间复杂度是动作与状态数量的多项式级，在面对维度灾难时，优于线性规划和直接搜索。

返回目录

4 蒙特卡洛（MONTE CARLO，MC）：采样更新

[10] 针对分幕式任务，不需要先验知识，即P，通过多幕采样数据获得经验代替值函数解决问题。

4.1 概念

核心需求 由于P的缺失，V是不够的，需要评估Q，即需要对每个状态-动作对进行评估。

行为值函数估计 给定的一幕中，指定状态的一次出现叫做对其的一次访问（visit），第一次出现为首次访问。可以不同程度地使用一幕数据。

- 首次访问（first visit）：$\hat{q}(s) = \frac{G_{11}(s,a)+G_{21}(s,a)+...}{N(s,a)}$。
- 每次访问（every visit）：$\hat{q}(s) = \frac{G_{11}(s,a)+G_{12}(s,a)+\cdots+G_{21}(s,a)+...}{N(s,a)}$。

$N(s)$是s的访问次数，$N(s) \to \infty, \hat{q}(s,a) \to q_{\pi}(s,a)$。

图 6: MC回溯图：显示一幕所有采样到的转移

幕长 靠近目标的状态比远离目标的状态更早具有非零值，幕长应足够长，无需无限长。

优势

- 不需要P。
- 对每个状态的估计是独立的，可聚焦于状态子集，无需考虑其他状态，此时效率很高。
- 可从实际经历和模拟经历中学习。
- 无马尔可夫性时性能损失较小。

4.2 on-policy（同轨）

[11] 采样并改进相同策略，为平衡探索和开发，采用ϵ-greedy策略。

返回目录

试探性出发（ES） 为采样部分无法正常获得的状态-动作对，可设定所有对都有概率作为起始。满足充分探索的理论要求，但实际中很难实现。

算法 3：MC-On-policy（首次访问）

```
参数：ϵ > 0
初始化：∀s ∈ S, a ∈ A(s)，任意初始化Q(s, a) ∈ R，初始化Returns(s, a)为空列表，
ϵ-greedy初始化策略π
循环
    根据π生成一幕序列S_0, A_0, R_1, S_1, A_1, R_2, . . . , S_{T−1}, A_{T−1}, R_T
    G ← 0
    对于 t = T − 1, T − 2, . . . , 0 执行
        G ← γG + R_{t+1}
        如果 S_t在此幕中首次出现 那么
            将G加入Returns(S_t, A_t)
            Q(S_t, A_t) ← average[Returns(S_t, A_t)]
            A* ← argmax_a Q(S_t, a)
            ϵ-greedy策略选取π(a|S_t)
```

4.3 off-policy（离轨）

[12] 采样与改进不同策略，前者称为行为策略（Behavior Policy）b（保证对所有可能动作的采样），后者称为目标策略（Target Policy）π，可视为特殊的离轨。

重要度采样（IMPORTANCE SAMPLING） [13]

计算回报时，对轨迹在目标策略和行为策略中出现的相对概率进行加权：

$$\rho_{t:T-1} = \Pi_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} \text{（约去相同的转移概率）}$$

- 普通重要度采样：$V(s) \doteq \frac{\sum_{t\in\tau(s)} \rho_{t:T(t)-1} G_t}{|\tau(s)|}$，无偏但无界。
- 加权重要度采样：$V(s) \doteq \frac{\sum_{t\in\tau(s)} \rho_{t:T(t)-1} G_t}{\sum_{t\in\tau(s)} \rho_{t:T(t)-1}}$，有偏但偏差值渐近收敛。

减小方差的方法：

返回目录

- 折扣敏感：把折扣率γ视作幕终止的概率，得到第n步终止的无折扣部分回报$\sum_{i=1}^{n} R_{t+i}$，即平价部分回报。全回报$G_t = \sum_{i=1}^{T-t} \gamma^{i-1} R_{t+i}$可视为各平价部分回报的加权和，即该步截止得到的回报与概率之积的和。适用于普通型和加权型。

- 每次决策型：$E[\rho_{t:T-1} G_t] = E[\tilde{G}_t] = E[\sum_{i=1}^{T-t} \gamma^{i-1} \rho_{t:t+i-1} R_{t+i}]$。适用于普通型。

增量式更新

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}[G_n - V_n]$$
$$C_{n+1} \doteq C_n + W_{n+1}$$

其中，W_i是随机权重，C_i是其累加和。

算法 4：MC-Off-policy（每次访问）

初始化：$\forall s \in S, a \in A(s)$，任意初始化$Q(s,a) \in R, C(s,a) = 0$，初始化$\pi(s) = \mathrm{argmax}_a Q(s,a)$ ▷ 目标策略为贪心策略
循环
  根据b生成一幕序列$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$ ▷ 行为策略为ϵ-greedy策略
  $G \leftarrow 0, W \leftarrow 1$
  对于 $t = T-1, T-2, \ldots, 0$ 执行
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$
    $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$
    $\pi(S_t) \leftarrow \mathrm{argmax}_a Q(S_t, a)$
    如果 $A_t \neq \pi(S_t)$ 那么
      break ▷ 如果不是最优动作则退出内层循环
    $W \leftarrow W \cdot \frac{1}{b(A_t|S_t)}$ ▷ 更新重要度采样权重

潜在问题：贪心行为普遍时，只会从幕尾学习；贪心行为不普遍时，学习速度较慢。

4.4 对比

- on-policy通常具有更高的稳定性，但可能需要更多样本才能收敛，因为每次策略更新后都需要新的数据。

- off-policy虽然可能更快找到好的解，但由于使用了不同的行为策略，学习过程可能不太稳定。

5 时序差分（TEMPORAL DIFFERENCE，TD）：采样更新

TD可直接从与环境的互动中获取信息，不需要P，同时运用自举思想，可基于已得到的其他状态估计来更新当前$v(s)$，相当于结合了DP和MC的优点。

5.1 TD(0)

14 TD(0)的更新公式为：

$$V_{t+1}(S_t) = V_t(S_t) + \alpha_t(S_t)[R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)]$$

- TD误差$\delta_t = R_{t+1} + \gamma V_t(S_{t+1}) - V_t(S_t)$。
- TD目标$R_{t+1} + \gamma V_t(S_{t+1})$

MC误差可写成TD误差之和$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k$，其在步长较小时成立。

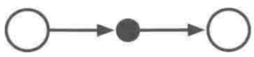


图 7: TD回溯图

优势

- 不需要P, R。
- 更新快：MC须等到幕尾确定增量，更新G_t；而TD只需等到下一时刻，更新TD目标。
- 只评估当前动作，与后续动作无关。

算法 5：TD(0)

输入：待评估策略π
参数：步长$\alpha \in (0, 1]$
初始化：$\forall s \in S^+$，任意初始化$V(s), V(终止状态) = 0$

返回目录

算法 5：TD(0)

对于 每一幕 执行
　初始化S
　当 S不是终止状态 执行
　　$A \leftarrow \pi(S)$
　　执行动作A，观察R, S'
　　$V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
　　$S \leftarrow S'$

随机游走　在随机任务实践中，TD(0)的收敛速度要比常量αMC快。这是因为前者的最优性与预测回报更相关，找出的是完全符合马尔可夫过程模型的最大似然估计参数，收敛到确定性等价估计；而后者只在有限方面最优，找出的是最小化训练集均方误差的估计。

批量更新　值函数根据增量和改变，在处理整批数据后才更新。

5.2 Sarsa（on-policy-TD）

[15] Sarsa（State-Action-Reward-State-Action）是TD算法的行为值函数版本：

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t[R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1}) - Q_t(S_t, A_t)]$$

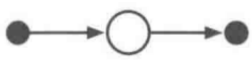


图 8: Sarsa回溯图

算法 6：Sarsa（on-policy-TD）

参数：步长$\alpha \in (0, 1], \epsilon > 0$
初始化：$\forall s \in S^+$，任意初始化$Q(s, a), Q(终止状态, \cdot) = 0$
对于 每一幕 执行
　初始化S
　使用从Q得到的ϵ-greedy策略，在S处选择A
　当 S不是终止状态 执行
　　执行动作A，观察R, S'

算法 6：Sarsa（on-policy-TD）

使用从Q得到的ϵ-greedy策略，在S'处选择A'

$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$

$S \leftarrow S', A \leftarrow A'$

期望SARSA [16]

$$Q_{t+1}(S_t,A_t) = Q_t(S_t,A_t) + \alpha_t[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1})Q_t(S_{t+1},a) - Q_t(S_t,A_t)]$$

期望Sarsa相较Sarsa，虽然计算复杂，但是消除了随机选择带来的方差。α的选择对二者有一定影响，尤其在长期稳态性能上。生成策略可以基于相同或不同策略，即离轨或在轨是可变的。基于此，Q-learning可视为期望Sarsa的一个特例。

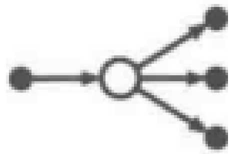


图 9: 期望Sarsa回溯图

5.3 Q-learning（off-policy-TD）

[17] Q-learning旨在求解行为值贝尔曼最优方程，直接逼近$q^*(s,a)$。

$$Q_{t+1}(S_t,A_t) = Q_t(S_t,A_t) + \alpha_t[R_{t+1} + \gamma \max_a Q_t(S_{t+1},a) - Q_t(S_t,A_t)]$$

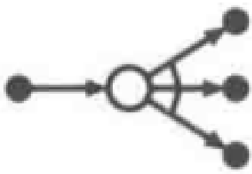


图 10: Q-learning回溯图

算法 7：Q-learning（off-policy-TD）

参数：步长$\alpha \in (0,1]$，探索率$\epsilon > 0$

初始化：$\forall s \in S^+, a \in A(s)$，任意初始化$Q(s,a), Q(终止状态,\cdot) = 0$

对于 每一幕 执行

返回目录

算法 7：Q-learning（off-policy-TD）

    初始化S
    当 S不是终止状态 执行
        使用从Q得到的ϵ-greedy策略，在S处选择A
        执行动作A，观察R, S'
        $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$
        $S \leftarrow S'$

双Q-LEARNING [18]

$$Q_{1_{t+1}}(S_t, A_t) = Q_{1_t}(S_t, A_t) + \alpha_t\{R_{t+1} + \gamma Q_{2_t}[S_{t+1}, argmax_a Q_{1_t}(S_{t+1}, a)] - Q_{1_t}(S_t, A_t)\}$$

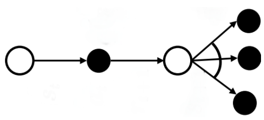


图 11: 双Q-learning回溯图

- 最大化偏差：贪心策略和柔性策略都在隐式估计最大值，会产生正偏差，致使回报值偏离，带来明显错误决策。
- 双学习：划分样本，学习两个独立的估计$Q_1(a), Q_2(a)$，确定动作$A* = argmax_a Q_1(a)$，再计算价值的估$Q_2(A*) = Q_2(argmax_a Q_1(a))$，后者是无偏的（可以交换再来一次）。需要双倍内存，但是计算量维持。
- 后位状态：利用先验知识，知晓动作后状态，并有后位值函数。在后位状态相同的时候可以迁移，减少计算量。

算法 8：双Q-learning

参数：步长$\alpha \in (0,1]$，探索率$\epsilon > 0$
初始化：$\forall s \in S^+, a \in A(s)$，任意初始化$Q_1(s,a), Q_2(s,a), Q_1(终止状态,\cdot) = Q_2(终止状态,\cdot) = 0$
对于 每一幕 执行
    初始化S
    当 S不是终止状态 执行

返回目录

算法 8：双Q-learning

基于Q_1+Q_2，使用ϵ-greedy策略在S处选择A
执行动作A，观察R,S'
如果 以0.5的概率 那么
    $Q_1(S,A)\leftarrow Q_1(S,A)+\alpha[R+\gamma Q_2(S',argmax_a Q_1(S',a))-Q_1(S,A)]$
否则
    $Q_2(S,A)\leftarrow Q_2(S,A)+\alpha[R+\gamma Q_1(S',argmax_a Q_2(S',a))-Q_2(S,A)]$
$S\leftarrow S'$

5.4 对比

Sarsa较为保守，在存在风险的任务中，会避开低回报动作；Q-learning较为乐观，更倾向于探索并找到最优解。在存在陷阱的任务中，Sarsa会比Q-learning取得更好的结果。

6 N步自举法

6.1 n-TD

[19] n-TD作为MC和TD的一般推广，在两种极端方法间找到了性能更好的平衡点。n-TD在n步后进行更新，截断得到n步回报。

$$G_{t:t+n}\doteq R_{t+1}+\gamma R_{t+2}+\cdots+\gamma^{n-1}R_{t+n}+\gamma^n V_{t+n-1}(S_{t+n})$$

其中$V_{t+n}(S_t)\doteq V_{t+n-1}(S_t)+\alpha[G_{t:t+n}-V_{t+n-1}(S_t)]$。

算法 9：n-TD

输入：待评估策略π
参数：步长$\alpha\in(0,1],n\in\mathbb{N}_+$
初始化：$\forall s\in S$，任意初始化$V(s)$
对于 每一幕 执行
    初始化S_0为非终止状态
    $T\leftarrow\infty$

返回目录

算法 9：n-TD

对于 $t = 0, 1, 2, \ldots$ 执行
　如果 $t < T$ 那么
　　根据$\pi(\cdot|S_t)$采取动作A_t
　　观察R_{t+1}, S_{t+1}
　　如果 S_{t+1}是终止状态 那么
　　　$T \leftarrow t+1$
　$\tau \leftarrow t-n+1$　　$\triangleright$ τ是正在更新的状态的时间
　如果 $\tau \geqslant 0$ 那么
　　$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
　　如果 $\tau+n < T$ 那么
　　　$G \leftarrow G + \gamma^n V(S_{\tau+n})$
　　$V(S_\tau) \leftarrow V(S_\tau) + \alpha[G - V(S_\tau)]$
　如果 $\tau = T-1$ 那么
　　break

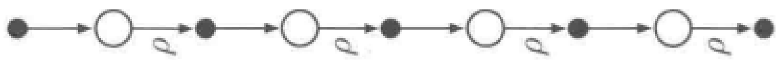
6.2 n-Sarsa

20 n-Sarsa统一了Sarsa和MC，其节点转移全部基于采样得到的单独路径：

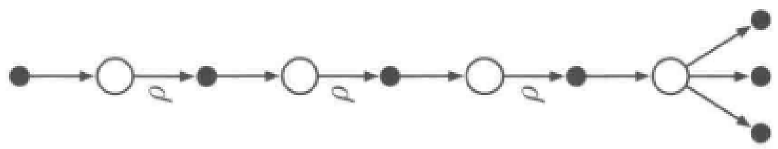
$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)]$$

n-期望Sarsa只对最后一个状态到动作的转移展开：

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n})$$



(a) n-Sarsa



(b) n-期望Sarsa

图 12: n-Sarsa回溯图

返回目录

算法 10：n-Sarsa

- 参数：步长$\alpha \in (0,1]$，探索率$\epsilon > 0$，步数$n \in N_+$
- 初始化：$\forall s \in S, a \in A$，任意初始化$Q(s,a)$，初始化π（如基于Q的ϵ-greedy策略）
- 对于 每一幕 执行
  - 初始化S_0为非终止状态
  - 根据$\pi(\cdot|S_0)$选取A_0
  - $T \leftarrow \infty$
  - 对于 $t = 0, 1, 2, \ldots$ 执行
    - 如果 $t < T$ 那么
      - 执行动作A_t，观察R_{t+1}, S_{t+1}
      - 如果 S_{t+1}是终止状态 那么
        - $T \leftarrow t + 1$
      - 否则
        - 根据$\pi(\cdot|S_{t+1})$选取A_{t+1}
    - $\tau \leftarrow t - n + 1$ ▷ τ是正在更新的状态的时间
    - 如果 $\tau \geqslant 0$ 那么
      - $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      - 如果 $\tau + n < T$ 那么
        - $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$
      - $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$
    - 如果 $\tau = T - 1$ 那么
      - break

针对离线n步时序差分学习有：

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\rho_{t:t+n-1}[G_{t:t+n} - V_{t+n-1}(S_t)]$$

其中，重要度采样率为目标策略和行为策略采取n个动作的相对概率：

$$\rho_{t:h} \doteq \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

算法 11：n-期望Sarsa-off-policy

输入：行为策略b，满足$b(a|s) > 0$
参数：步长$\alpha \in (0, 1]$，探索率$\epsilon > 0$，步数$n \in \mathbb{N}_+$
初始化：$\forall s \in S, a \in A$，任意初始化$Q(s, a)$，初始化目标策略π
对于 每一幕 执行
  初始化S_0为非终止状态
  根据$b(\cdot|S_0)$选取A_0
  $T \leftarrow \infty$
  对于 $t = 0, 1, 2, \ldots$ 执行
    如果 $t < T$ 那么
      执行动作A_t，观察R_{t+1}, S_{t+1}
      如果 S_{t+1}是终止状态 那么
        $T \leftarrow t + 1$
      否则
        根据 $b(\cdot|S_{t+1})$选取A_{t+1}
    $\tau \leftarrow t - n + 1$ ▷ τ是正在更新的状态的时间
    如果 $\tau \geqslant 0$ 那么
      $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ ▷ 重要性采样权重
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
      如果 $\tau + n < T$ 那么
        $G \leftarrow G + \gamma^n \sum_a \pi(a|S_{\tau+n}) Q(S_{\tau+n}, a)$ ▷ 期望Sarsa使用期望值
      $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho[G - Q(S_\tau, A_\tau)]$
    如果 $\tau = T - 1$ 那么
      break

6.3 n-树回溯

21

带控制变量的每次决策模型

返回目录

为保证不被选择的动作不会因$\rho_t=0$而回报为0，使方差较大，采取以下n步回报off-policy方法：

$$G_{t:h} \doteq \rho_t(R_{t+1}+\gamma G_{t+1:h})+(1-\rho_t)V_{h-1}(S_t)$$

其中$(1-\rho_t)V_{h-1}(S_t)$称为控制变量，其能保证$\rho_t=0$时估计值不收缩，但不会改变更新值的期望。

可写为以下递归形式：

$$\begin{aligned} G_{t:h} &\doteq R_{t+1}+\gamma[\rho_{t+1}G_{t+1:h}+\bar{V}_{h-1}(S_{t+1})-\rho_{t+1}Q_{h-1}(S_{t+1},A_{t+1})] \\ &= R_{t+1}+\gamma\rho_{t+1}[G_{t+1:h}-Q_{h-1}(S_{t+1},A_{t+1})]+\gamma\bar{V}_{h-1}(S_{t+1}) \end{aligned}$$

n-树回溯

off-policy因所学内容相关性小，比on-policy慢，一些方法可以缓解这一问题，比如不使用重要度采样的树回溯算法。相比于前面以沿途收益和底部节点估计价值为更新目标的算法，树回溯的更新源于整个树的行为值估计，即各叶子节点的行为值估计按出现概率加权。单步回溯树：

$$G_{t:t+1} \doteq R_{t+1}+\gamma\sum_a \pi(a|S_{t+1})Q_t(S_{t+1},a)$$

拓展到n-回溯树的递归形式，其对路径可能分支进行展开，不进行采样：

$$G_{t:t+n} \doteq R_{t+1}+\gamma\sum_{a\neq A_{t+1}}\pi(a|S_{t+1})Q_{t+n-1}(S_{t+1},a)+\gamma\pi(A_{t+1}|S_{t+1})G_{t+1:t+n}$$

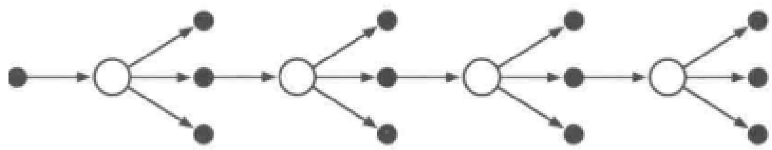


图 13: n-树回溯回溯图

算法 12：n-树回溯

参数：步长$\alpha\in(0,1], n\in\mathbb{N}_+$
初始化：$\forall s\in S, a\in A$，任意初始化$Q(s,a)$，初始化π
对于 每一幕 执行
    初始化S_0为非终止状态

算法 12：n-树回溯

根据S_0任意选取A_0
$T \leftarrow \infty$
对于 $t = 0, 1, 2, \ldots$ 执行
  如果 $t < T$ 那么
    执行动作A_t，观察R_{t+1}, S_{t+1}
    如果 S_{t+1}是终止状态 那么
      $T \leftarrow t+1$
    否则
      根据S_{t+1}选取A_{t+1}
  $\tau \leftarrow t - n + 1$ ▷ τ是正在更新的状态的时间
  如果 $\tau \geqslant 0$ 那么
    如果 $t+1 \geqslant T$ 那么
      $G \leftarrow R_T$
    否则
      $G \leftarrow R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$
    对于 $k = \min(t, T-1)$递减到$\tau+1$ 执行
      $G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a|S_k) Q(S_k, a) + \gamma \pi(A_k|S_k) G$
    $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$
  如果 $\tau = T-1$ 那么
    break

6.4 n-Q(σ)

[22] 结合采样的Sarsa和展开的树回溯，在每个状态由参数σ决定是采样还是展开，将两种线性情况组合起来：

$$G_{t:h} \doteq R_{t+1} + \gamma(\sigma_{t+1}\rho_{t+1} + (1-\sigma_{t+1})\pi(A_{t+1}|S_{t+1}))(G_{t+1:h} - Q_{h-1}(S_{t+1}, A_{t+1})) + \gamma \bar{V}_{h-1}(S_{t+1})$$

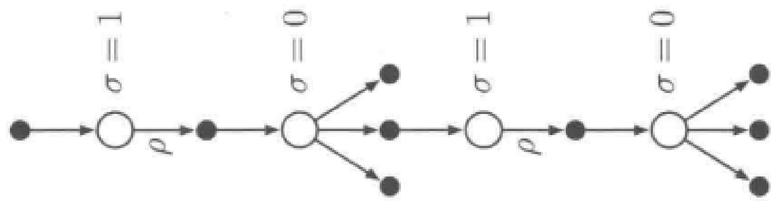


图 14: Q(sigma)回溯图

算法 13：n-Q(σ)-off-policy

输入：行为策略b，满足$b(a|s) > 0$
参数：步长$\alpha \in (0,1]$，探索率$\epsilon > 0$，步数$n \in \mathbb{N}_+$
初始化：$\forall s \in S, a \in A$，任意初始化$Q(s,a)$，初始化目标策略π
对于 每一幕 执行
  初始化S_0为非终止状态
  根据$b(\cdot|S_0)$选取A_0
  $T \leftarrow \infty$
  对于 $t = 0,1,2,\dots$ 执行
    如果 $t < T$ 那么
      执行动作A_t，观察R_{t+1}, S_{t+1}
      如果 S_{t+1}是终止状态 那么
        $T \leftarrow t+1$
      否则
        根据$b(\cdot|S_{t+1})$选取A_{t+1}
        选择σ_{t+1} ▷ 指示是采样还是展开
        $\rho_{t+1} \leftarrow \frac{\pi(A_{t+1}|S_{t+1})}{b(A_{t+1}|S_{t+1})}$ ▷ 重要性采样比率
    $\tau \leftarrow t-n+1$ ▷ τ是正在更新的状态的时间
    如果 $\tau \geqslant 0$ 那么
      $G \leftarrow 0$
      对于 $k = \min(t, T-1)$递减到$\tau+1$ 执行
        如果 $k = T$ 那么
          $G \leftarrow R_T$
        否则
          $\bar{V} \leftarrow \sum_a \pi(a|S_k)Q(S_k,a)$ ▷ 计算期望状态值

返回目录

算法 13：n-Q(σ)-off-policy

$G \leftarrow R_k + \gamma[\sigma_k\rho_k + (1-\sigma_k)\pi(A_k|S_k)][G - Q(S_k, A_k)] + \gamma\bar{V}$

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$

如果 $\tau = T - 1$ 那么

break

7 表格型方法总结对比

[23] 基于模型的方法（DP、启发式搜索）主要进行规划，无模型的方法（MC、TD）主要进行学习，二者的核心都是值函数的计算。

表格型方法介绍 见12.3

三个维度

- 更新
- 自举程度
- 同轨/离轨

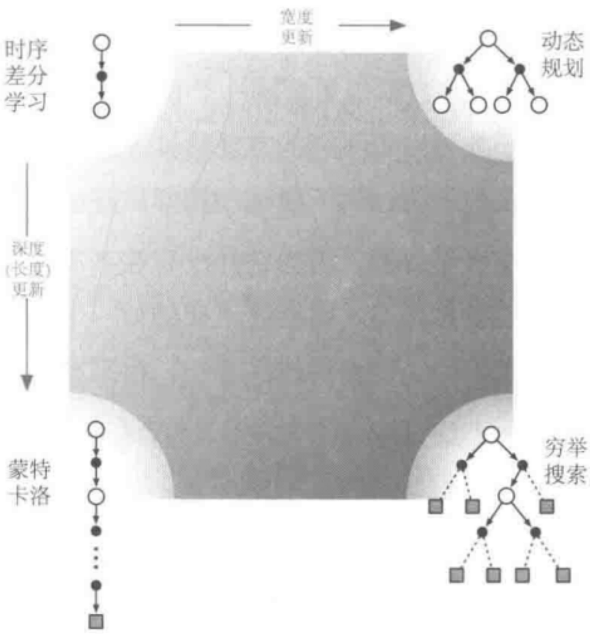


图 15: 表格型方法总结对比

更新 期望更新能产生更好的估计，但是需要更多的计算。

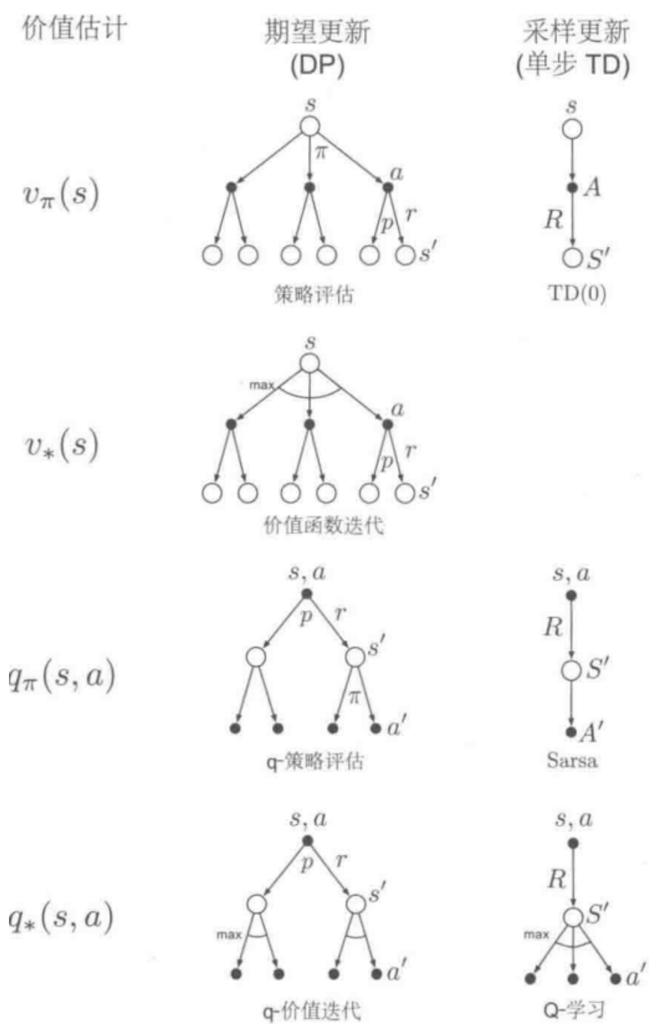
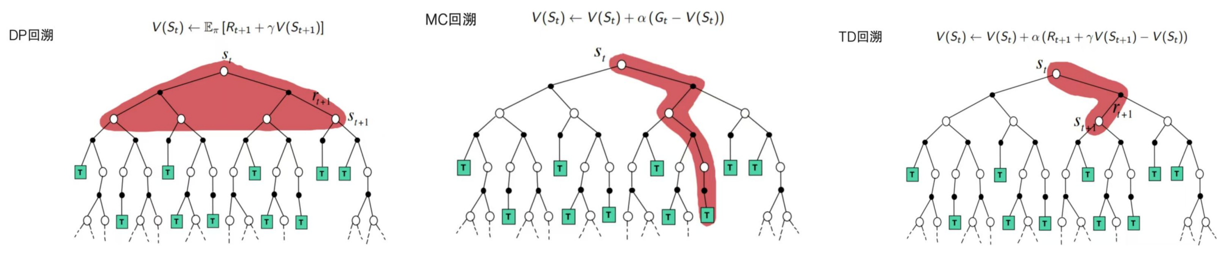


图 16: 更新对比



(a) DP (b) MC (c) TD

图 17: 表达式对比

表达式对比 统一格式：

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha_t(S_t, A_t)[\bar{q}_t - Q_t(S_t, A_t)]$$

返回目录

算法	$\bar{q}_t$的表达式
Sarsa	$\bar{q}_t = r_{t+1} + \gamma q_t(s_{t+1}, a_{t+1})$
n步Sarsa	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^n q_t(s_{t+n}, a_{t+n})$
期望Sarsa	$\bar{q}_t = r_{t+1} + \gamma \sum_a \pi_t(a\|s_{t+1}) q_t(s_{t+1}, a)$
Q - 学习算法	$\bar{q}_t = r_{t+1} + \gamma \max_a q_t(s_{t+1}, a)$
蒙特卡罗算法	$\bar{q}_t = r_{t+1} + \gamma r_{t+2} + \cdots$

图 18: 表达式对比

8 值函数近似

8.1 函数近似

$$\hat{v}(s, w) \approx v_\pi(s), w \in R^d, d \ll |S|$$

目标函数

$$J(\omega) = E[(v_\pi(S) - \hat{v}(S, \omega))^2]$$

在对状态按重要程度进行加权后，可得到均方值误差：

$$\overline{VE}(w) \doteq \sum_{s \in S} \mu(s)[v_\pi(s) - \hat{v}(s, w)]^2$$

一般无法保证最优，求解局部最优。

状态分布

- 均匀分布（各状态同等重要）：$J(\omega) = \frac{1}{|S|} \sum_{s \in S} [v_\pi(s) - \hat{v}(s, \omega)]^2$。
- 平稳分布（马氏过程长期行为）：$J(\omega) = \sum_{s \in S} d_\pi(s)[v_\pi(s) - \hat{v}(s, \omega)]^2$。

优势

- 具有一定泛化能力，适应部分观测问题。
- 曲线拟合：用少量参数储存状态，阶数越高越近似。

返回目录

8.2 随机梯度下降（SGD）

[24]

$$\omega_{k+1} = \omega_k - \alpha_k \nabla_\omega J(\omega_k)$$

其中，

$$\begin{aligned}
\nabla_\omega J(\omega) &= \nabla_\omega E[(v_\pi(S) - \hat{v}(S,\omega))^2] \\
&= E[\nabla_\omega (v_\pi(S) - \hat{v}(S,\omega))^2] (\text{有界可换求导与期望顺序}) \\
&= -2E[(v_\pi(S) - \hat{v}(S,\omega))\nabla_\omega \hat{v}(S,\omega)]
\end{aligned}$$

因此$\omega_{k+1} = \omega_k + \alpha(v_\pi(s_k) - \hat{v}(s_k,\omega_k))\nabla_\omega \hat{v}(s_k,\omega_k)$。

负梯度方向降速最快　梯度方向增长最快，负梯度方向下降最快。

步长α

近似方法

- $v_\pi(s_t)$：
  - 蒙特卡洛：g_t。
  - 时序差分：$r_{t+1} + \gamma \hat{v}(s_{t+1},\omega_t)$。
- $\hat{v}(S,\omega)$：
  - 线性参数：$\hat{v}(S,\omega) = \phi(S)^T\omega$，$\phi(S)$为特征函数。可采用多项式基函数、傅里叶基函数或径向基函数，表格法可视为其特殊情况。可以使用最小二乘法来减少迭代产生的计算量。
  - 非线性参数：神经网络，输入状态，网络参数为ω，输出$\hat{v}(S,\omega)$。
  - 核函数（见12.4）、高斯回归等非参数方法。

半梯度方法　只考虑w_t对估计值的影响，而忽略对目标的影响。在使用自举目标时，目标本身依赖于当前w，这使得它们有偏。

- 优势：学习速度较快，支持持续在线学习，无需等待幕结束。
- 局限：稳健性差，在非线性函数近似中可能不稳定。

返回目录

算法 14：梯度蒙特卡洛

输入：待评估π，可微函数$\hat{v}: S \times R^d \to R$
参数：步长$\alpha > 0$
初始化：任意初始化$w \in R^d$
循环 ▷ 对每一幕
  根据π生成一幕交互数据$S_0, A_0, R_1, S_1, A_1, \cdots, R_T, S_T$
  对于 $t = 0, 1, \cdots, T-1$ 执行
    $w \leftarrow w + \alpha[G_t - \hat{v}(S_t, w)]\nabla\hat{v}(S_t, w)$

算法 15：半梯度TD(0)

输入：待评估π，可微函数$\hat{v}: S^+ \times R^d \to R, \hat{v}(终止状态, \cdot) = 0$
参数：步长$\alpha > 0$
初始化：任意初始化$w \in R^d$
循环 ▷ 对每一幕
  初始化S
  对于 $t = 0, 1, \cdots, T-1$ 执行
    选取$A \sim \pi(\cdot|S)$并采取，观察R, S'
    $w \leftarrow w + \alpha[R + \gamma\hat{v}(S', w) - \hat{v}(S, w)]\nabla\hat{v}(S, w)$
    $S \leftarrow S'$
    如果 S' 为终止状态 那么
      break

8.3 DQN（Deep Q-Network）

[25] DQN用神经网络作为非线性函数近似器，最小化损失函数（贝尔曼最优性误差），适用于高维空间的状态和动作问题：

$$J(\omega) = E[(R + \gamma \max_{a'} \hat{q}(S', a', \omega^-) - \hat{q}(S, A, \omega))^2]$$

其中ω为主网络参数，ω^-为目标网络参数。

主要技术

- 两个网络：主网络$\hat{q}(S, A, \omega)$和目标网络$\hat{q}(S', a', \omega^-)$，后者参数阶段性从前者同步。

返回目录

- 防止过拟合：
  - 随机丢弃法（dropout）。
  - 批量归一化（batch normalization）。
  - 残差直连边。
- 更新：
  - 软更新：部分更新。
  - 硬更新：直接复制。

- 经验回放（Experience Replay）：存储经验到固定大小的回放缓冲区，训练时从中随机选取。可以打乱样本相关性，提升训练稳定性，可改进为优先经验回放。
- 帧堆叠：将图像作为神经网络输入时，堆叠多帧图像作为输入，并跳帧选取放入帧，增加时间信息。
- 奖励裁剪（Reward Clipping）：将奖励限制在特定范围内（甚至使用符号函数），避免大奖励幅度波动，提升训练稳定性，适用于奖励范围差异大的环境。

算法 16：DQN

初始化主网络参数ω和目标网络参数ω^-
初始化经验回放缓冲区$B=\{(s,a,r,s')\}$
初始化计数器$t \leftarrow 0$
循环
    如果 $t \mod C = 0$ 那么 ▷ 每隔C步更新目标网络（初始化一致）
        $\omega^- \leftarrow \omega$
    从B中均匀采样小批量样本$\{(s,a,r,s')\}$
    对于 每个样本(s,a,r,s') 执行
        如果 s' 是终止状态 那么
            $y \leftarrow r$
        否则
            $y \leftarrow r + \gamma \max_{a'} \hat{q}(s', a', \omega^-)$ ▷ 计算目标值
    使用小批量样本$\{(s,a,y)\}$更新主网络参数ω，最小化损失$(y-\hat{q}(s,a,\omega))^2$
    $t \leftarrow t+1$

DOUBLE-DQN 两个值函数逼近网络，一个选择动作，一个评估值函数。

返回目录

9 策略梯度（POLICY GRADIENT）

[26] 将策略参数化，在策略空间进行搜索。

9.1 概念

$$\pi(a|s,\theta)$$

目标 学习θ使$U(\theta)=\sum_\tau P(\tau,\theta)\underbrace{\sum_{t=0}^{H}R(s_t,u_t)}_{R(\tau)}$最大，可使用梯度上升算法。

似然率策略梯度

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_\tau P(\tau,\theta)R(\tau) \\
&= \sum_\tau \nabla_\theta P(\tau,\theta)R(\tau) \\
&= \sum_\tau P(\tau,\theta)\frac{\nabla_\theta P(\tau,\theta)}{P(\tau,\theta)}R(\tau) \\
&= \sum_\tau P(\tau,\theta)\nabla_\theta \log P(\tau,\theta)R(\tau)
\end{aligned}$$

利用经验平均后为：

$$\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{m}\sum_{i=1}^{m}\nabla_\theta \log P(\tau,\theta)R(\tau)$$

返回目录

其无偏但方差很大，其中

$$
\begin{aligned}
\nabla_\theta \log P(\tau^{(i)}, \theta) &= \nabla_\theta \log[\prod_{t=0}^{H} P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)}) \cdot \pi_\theta(u_t^{(i)}|s_t^{(i)})] \\
&= \nabla_\theta[\underbrace{\sum_{t=0}^{H} \log P(s_{t+1}^{(i)}|s_t^{(i)}, u_t^{(i)})}_{\text{与动力学无关}} + \underbrace{\sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)})]}_{\text{只与策略相关}} \\
&= \nabla_\theta[\sum_{t=0}^{H} \log \pi_\theta(u_t^{(i)}|s_t^{(i)})] \\
&= \sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})
\end{aligned}
$$

其中$\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})$是策略梯度，根据具体情况计算。

优势

- 可以逼近确定性策略。
- 可以逼近任意概率分布，不受$q(s, a)$限制。
- 策略是更简单的函数逼近，如PID控制。
- 策略参数化更容易加入先验知识。

9.2 REINFORCE（MC-policy gradient）

27

$$
\theta_{t+1} \doteq \theta_t + \alpha G_t \nabla \log \pi(a_t|s_t; \theta_t)
$$

减小方差的方法

1. 基线：

$$
\begin{aligned}
\nabla_\theta U(\theta) \approx \hat{g} &= \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}, \theta) R(\tau^{(i)}) \\
&= \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}, \theta)[R(\tau^{(i)}) - b] \text{添加无关常数}
\end{aligned}
$$

返回目录

其中$E[\nabla_\theta \log P(\tau,\theta)b]=0$，

为使$\nabla_\theta \log P(\tau^{(i)},\theta)[R(\tau^{(i)})-b]$的方差最小，取：

$$b=\frac{E_p[(\sum_{t=0}^{H}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)}))^2 R(\tau)]}{E_p[(\sum_{t=0}^{H}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)}))^2]}$$

2. 修改值函数：当前动作与过去回报无关，只与过去动作有关。

$$\begin{aligned}\nabla_\theta U(\theta) &\approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})(\sum_{k=t}^{H-1}(R(s_k^{(i)})-b))\\ &\approx \frac{1}{m}\sum_{i=1}^{m}\sum_{j=0}^{H-1}(\sum_{t=0}^{j}\nabla_\theta \log \pi_\theta(u_t^{(i)}|s_t^{(i)})(r_j-b_j))\end{aligned}$$

算法 17：REINFORCE

1: 输入：可微分的参数化策略$\pi(a|s,\theta)$
2: 参数：步长$\alpha>0$
3: 初始化：初始化策略参数$\theta\in R^{d'}$
4: 循环 ▷ 对每一幕循环
5: 按照$\pi(\cdot|\cdot,\theta)$生成一幕$S_0,A_0,R_1,\ldots,S_{T-1},A_{T-1},R_T$
6: 对于 $t=0,1,\ldots,T-1$ 执行
7: $G\leftarrow\sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$ ▷ (G_t)
8: $\theta\leftarrow\theta+\alpha\gamma^t G\nabla\ln\pi(A_t|S_t,\theta)$

10 ACTOR-CRITIC方法

11 策略搜索方法总结对比

$$\text{策略搜索方法}\begin{cases}\text{无模型}\begin{cases}\text{随机策略}\begin{cases}\text{策略梯度：TRPO}\\ \text{统计学习}\\ \text{路径积分}\end{cases}\\ \text{确定性策略：DDPG}\end{cases}\\ \text{基于模型：指导策略搜索(GPS)，传统控制器+监督学习}\end{cases}$$

12 附录

12.1 历史

1. 源于动物学习心理学的试错法：效应定律（Edward Thorndike），条件反射（巴普洛夫），快乐-痛苦系统（图灵），向“老师”学习到向“评论家”学习，自动学习机（M.L.Tsetlin），分类器系统（救火队算法和遗传算法）。
2. 最优控制：贝尔曼方程与马尔可夫决策过程（Richard Bellman），维度灾难。
3. 时序差分方法：次级强化物，广义强化（Klopf），与试错法结合（“行动器-评判器”结构，Sutton），与最优控制结合（Q-learning，Chris Watkins）。

返回正文1。

12.2 贝尔曼最优方程求解

收缩映射定理 若$f(x)$是收缩映射，则存在唯一一个不动点x^*满足$f(x^*)=x^*$。针对$x_{k+1}=f(x_k)$，在$x_k\to x^*, k\to\infty$的过程中，收敛速度成指数级增长。

- 存在性：$\|x_{k+1}-x_k\| = \|f(x_{k+1})-f(x_k)\| \leqslant \gamma\|x_k-x_{k-1}\| \leqslant \cdots \leqslant \gamma^k\|x_1-x_0\|$，由于$\gamma<1$，$\gamma^k\to 0$，所以$x_{k+1}-x_k\to 0$。同理可得$\|x_m-x_n\|\leqslant \frac{\gamma^n}{1-\gamma}\|x_1-x_0\|\to 0$。进而得到$\{x_k\}$是收敛数列，存在$\lim_{k\to\infty} x_k = x^*$。

- 唯一性：$\|f(x_k)-x_k\| = \|x_{k+1}-x_k\|$，其快速收敛到0，则在极限处有不动点$f(x^*)=x^*$。假设存在另一不动点，其必与该不动点相等。

- 指数级收敛：$\|x^*-x_n\| = \lim_{m\to\infty}\|x_m-x_n\| \leqslant \frac{\gamma^n}{1-\gamma}\|x_1-x_0\|\to 0$。

贝尔曼最优方程的伸缩映射性

$\forall v1,v2$，有贝尔曼最优方程$\pi_i^* \doteq \arg\max_\pi(r_\pi+\gamma P_\pi v_i)$，
故$f(v_i)=\max_\pi(r_\pi+\gamma P_\pi v_i) = r_{\pi_i^*}+\gamma P_{\pi_i^*}v_i \geqslant r_{\pi_j^*}+\gamma P_{\pi_j^*}v_i (i\neq j)$，
则

$$
\begin{aligned}
f(v_1)-f(v_2) &= r_{\pi_1^*}+\gamma P_{\pi_1^*}v_1-(r_{\pi_2^*}+\gamma P_{\pi_2^*}v_2)\\
&\leqslant r_{\pi_1^*}+\gamma P_{\pi_1^*}v_1-(r_{\pi_1^*}+\gamma P_{\pi_1^*}v_2)\\
&= \gamma P_{\pi_1^*}(v_1-v_2)
\end{aligned}
$$

同理有$f(v_2)-f(v_1)\leqslant \gamma P_{\pi_2^*}(v_2-v_1)$，
故$\gamma P_{\pi_2^*}(v_1-v_2)\leqslant f(v_1)-f(v_2)\leqslant \gamma P_{\pi_1^*}(v_1-v_2)$，
取边界极值z，有$|f(v_1)-f(v_2)|\leqslant z$，即$\|f(v_1)-f(v_2)\|_\infty\leqslant\|z\|_\infty$。
又有$\|z\|_\infty=\max_i|z_i|\leqslant\gamma\|v_1-v_2\|_\infty$，所以$\|f(v_1)-f(v_2)\|_\infty\leqslant\gamma\|v_1-v_2\|_\infty$。
故贝尔曼最优方程有伸缩映射性。

贝尔曼最优方程解的性质

- 唯一性：唯一解v^*能通过$v_{k+1}=f(v_k)=\max_{\pi\in\Pi}(r_\pi+\gamma P_\pi v_k)$迭代求解，其对应策略$\pi^*=\operatorname{argmax}_{\pi\in\Pi}(r_\pi+\gamma P_\pi v^*)$。

- 最优性（$v^*=v_{\pi^*}\geqslant v_\pi$）：由$v_\pi=r_\pi+\gamma P_\pi v_\pi$和$v^*=\max_\pi(r_\pi+\gamma P_\pi v^*)=r_{\pi^*}+\gamma P_{\pi^*}v^*\geqslant r_\pi+\gamma P_\pi v^*$，可得$v^*-v_\pi\geqslant(r_\pi+\gamma P_\pi v^*)-(r_\pi+\gamma P_\pi v_\pi)=\gamma P_\pi(v^*-v_\pi)$，即有$v^*-v_\pi\geqslant\gamma P_\pi(v^*-v_\pi)\geqslant\cdots\geqslant\gamma^n P_\pi^n(v^*-v_\pi)$，由于$\gamma<0$，$\forall p_{ij}\in P_\pi, p_{ij}\leqslant 1$，$\lim_{n\to\infty}\gamma^n P_\pi^n(v^*-v_\pi)$趋于o，所以$v^*\geqslant v_\pi$。

返回正文2.2.2。

12.3 表格型方法

12.3.1 模型和规划

模型

- 分布模型：生成所有可能的结果的描述与概率分布。
- 样本模型：从所有可能中生成一个确定的结果，其通过概率分布采样得到。
- 分布模型可以生成样本模型，但样本模型一般更容易获得。

规划

- 规划：以环境模型为输入，生成或改进与其进行交互的策略。
- 规划空间：
  - 状态空间规划：在状态空间搜索最优策略。
  - 方案空间规划：进化算法、偏序规划。
- 规划时间：
  - 后台规划：从环境模型生成模拟经验，改进策略或值函数。
    * 表格型方法
    * 近似方法
  - 决策时规划：使用模拟经验为当前状态选择动作

统一的状态空间规划算法

通过仿真经验的回溯操作计算值函数，将其作为改善策略的中间步骤。

$$\text{模型} \Longrightarrow \text{模拟经验} \overset{\text{回溯}}{\Longrightarrow} \text{值函数} \Longrightarrow \text{策略}$$

各算法的差异集中在回溯操作、执行操作顺序、回溯信息保留时长上。极小步长适于大尺度规划问题。

12.3.2 *Dyna-Q*

学习和规划由相同算法完成，真实经验用于学习，模拟经验用于规划。

返回目录

框架

- 间接强化学习：更充分地利用有限经验，获得更好的策略，减少与环境的交互作用。
- 直接强化学习：不受模型设计偏差影响。

算法 18：表格型Dyna-Q

初始化：$\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$，初始化$Q(s,a)$和$Model(s,a)$
循环
 $S \leftarrow$ 当前状态（非终止状态）
 基于(S,Q)选取A ▷ 例如使用ϵ-greedy策略
 执行动作A，观察R, S'
 $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$ ▷ 直接强化学习更新
 $Model(S,A) \leftarrow R, S'$
 对于 $i = 1, \dots, n$ 执行 ▷ 规划
  随机选择已观测过的S和其下采取过A
  $R, S' \leftarrow Model(S,A)$ ▷ 从模型获取预测
  $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma \max_a Q(S',a) - Q(S,A)]$ ▷ 规划更新

12.3.3 改进方法

模型错误 鼓励长期未出现动作，其模型可能不正确，规避在次优解收敛。

优先遍历 相比于均匀采样无长期收益的动作，集中更新有收益的动作，反向聚焦提供了相应的思路。关联前导动作和前导状态，在后续动作有收益时先更新前导动作价值，进行有效更新。按照价值改变多少对状态-动作对进行优先级排序，并由后至前反向传播出高影响序列。优先遍历为提高规划效率分配了计算量，但由于采用期望更新而在随机环境中有所局限。

算法 19：确定性环境下的优先级遍历

初始化：$\forall s \in \mathcal{S}, a \in \mathcal{A}(s)$，初始化$Q(s,a), Model(s,a)$，初始化优先级队列PQueue为空
循环
 $S \leftarrow$当前状态（非终止状态）
 基于(S,Q)选取A ▷ 例如使用ϵ-greedy策略

算法 19：确定性环境下的优先级遍历

执行动作A，观察R, S'
$\text{Model}(S, A) \leftarrow R, S'$
$P \leftarrow |R + \gamma \max_a Q(S', a) - Q(S, A)|$ ▷ 计算优先级
如果 $P > 0$ 那么
  将(S, A)以优先级P插入PQueue
对于 $i = 1, \ldots, n$ 执行 ▷ 进行n次规划更新
  如果 PQueue为空 那么
    break
  $(S, A) \leftarrow \text{PQueue}(0)$ ▷ 取出优先级最高的状态-动作对
  $R, S' \leftarrow \text{Model}(S, A)$ ▷ 从模型获取预测
  $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$ ▷ 规划更新
  对于 每个可达到S的状态-动作对$(\bar{S}, \bar{A})$ 执行 ▷ 反向传播更新
    $\bar{R}, \bar{S}' \leftarrow \text{Model}(\bar{S}, \bar{A})$
    如果 $\bar{S}' = S$ 那么
      $P \leftarrow |\bar{R} + \gamma \max_a Q(S, a) - Q(\bar{S}, \bar{A})|$
      如果 $P > 0$ 那么
        将$(\bar{S}, \bar{A})$以优先级P插入PQueue

轨迹采样　借助模拟生成经验回溯更新。on-policy轨迹采样对于大尺度问题有一定优势，能够跳过无关状态，获得最优部分策略。实时动态规划（RTDP）是on-policy轨迹采样值迭代版本，属于异步DP，可以在较少访问频率下为一些任务找到最优策略，并且产生轨迹所用的策略也会接近最优策略。

启发式搜索　聚焦于当前状态。

预演算法　作为MC的特例，通过平均多个起始于可能动作并遵循给定策略的模拟轨迹的回报来估计动作价值，可以改进预演策略性能。蒙特卡洛树搜索（MCTS）作为一种预演算法，通过累积蒙特卡洛模拟得到的值估计来不断将模拟导向高收益轨迹。其一次循环中包含选择、扩展、模拟、回溯四个步骤。

返回正文23。

12.4 核函数近似

核函数作为一种基于记忆的方法，通过计算特征向量的内积来衡量状态间相关性，适用于局部近似和高维状态空间问题。

- 基于记忆样本：使用RBF核的核函数以存储样本的状态为中心，每个特征对应一个样本状态。
- 非参数化：不需要学习参数。

与线性参数化的关系

- 等价性：任何线性参数化方法都可以被重塑为核函数。当状态由特征向量$x(s)$表示时，核函数$k(s,s')$可表示为特征向量的内积：

$$k(s,s') = x(s)^T x(s')$$

- 相同结果：如果使用相同的特征向量和训练数据，核函数与线性参数化会得到相同的近似结果。

优势 避免高维计算，高效处理高维特征。

返回正文24。

12.5 数学基础

概率空间(Ω, F, P)

- 性质
  - 非负性：$\forall A \in F, P(A) \geqslant 0$。
  - 规范性：$P(\Omega) = 1$。
  - 可列可加性：若$A_1, A_2, \ldots$互斥，则$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$。
- 运算
  - 补集：$P(A^c) = 1 - P(A)$。
  - 交集：$P(A \cap B) = P(A) + P(B) - P(A \cup B)$。

随机变量

- 离散型
  - 概率质量函数(PMF)：$P(X=x)=p(x)$，满足 $\sum_x p(x)=1$。
  - 期望：$E[X]=\sum_x x\cdot p(x)$。
- 连续型
  - 概率密度函数(PDF)：$f(x)\geqslant 0$，满足$\int_{-\infty}^{\infty} f(x)dx=1$。
  - 期望：$E[X]=\int_{-\infty}^{\infty} x\cdot f(x)dx$。
- 方差：$Var(X)=E[(X-E[X])^2]$。

条件概率与独立性

- 条件概率：$P(B|A)=\frac{P(A\cap B)}{P(A)}$当$P(A)>0$。
- 全概率公式：$P(B)=\sum_{A\subseteq F} P(B|A)P(A)$。
- 贝叶斯定理：$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$。
- 独立性： A,B独立 $\iff P(A\cap B)=P(A)P(B)$。
- 条件独立：$P(A,B|C)=P(A|C)P(B|C)$。

马尔可夫链与转移概率

- 马尔可夫性（无记忆性）：$P(S_{t+1}|S_t,S_{t-1},\ldots,S_0)=P(S_{t+1}|S_t)$。
- 转移矩阵：$P\in[0,1]^{S\times S}, P(s'|s)=\sum_a P(s'|s,a)P(a|s)$。

大数定律与中心极限定理

- 弱大数律：$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{p} E[X]$。
- 强大数律：$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{a.s.} E[X]$。
- 中心极限定理：$X_1,X_2,\ldots$独立同分布，均值为μ，方差为$\sigma^2<\infty$，则$\frac{1}{\sqrt{n}}\sum_{i=1}^{n}(X_i-\mu)\xrightarrow{d} N(0,\sigma^2)$。

返回目录

泛函分析

- 期望的线性：$E[aX + bY] = aE[X] + bE[Y]$。
- 协方差：$Cov(X, Y) = E[(X - \mu_X)(Y - \mu_Y)] = E[XY] - \mu_X \mu_Y$。
- 相关系数：$\rho(X, Y) = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$。

返回目录