

Reinforcement Learning

Q.

Regular MDP but twist: don't know T or R

T : model $T(s, a, s')$

R : reward function $R(s, a, s')$

→ Model-Based learning

Idea: Learn an Approximation model based on experiences

Solve value as if learned model correct

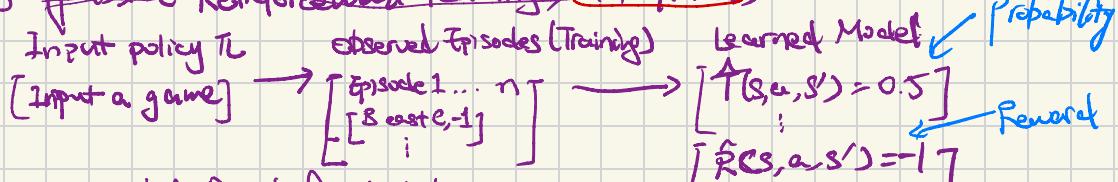
Step 1: Learn empirical MDP model

- o Count outcome s' for each s, a
- o Normalize to give an estimate of $\hat{T}(s, a, s')$
- o Discover each $\hat{R}(s, a, s')$ when experience (s, a, s')

Step 2: Solve learned MDP

- o For example, use value iteration, as before

Eg. ~~(Passive Reinforcement learning, not optimised)~~



Eg. — Model-Based & Model-Free

Goal: Compute expected age of students here

Known $P(A)$

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \dots$$

Without $P(A)$, instead collect samples $[a_1, a_2, \dots, a_N]$

Unknown $P(A)$: "Model Based"

Why does this work? Because eventually you learn the right model.

$$\hat{P}(a) = \frac{\text{num}(a)}{N}$$
$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Unknown $P(A)$: "Model Free"

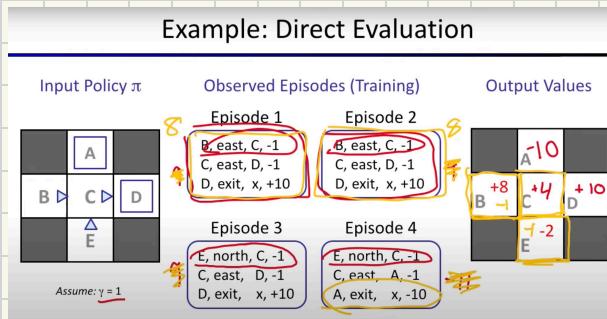
Why does this work? Because samples appear with the right frequencies.

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

→ Passive Learning : Direct Evaluation.

- Goal: Compute values for each state under π_L
- Idea: Average together observed sample values
 - Act according to π_L
 - Everytime visited a state, write down what the sum of discounted reward
 - Average those sample
- Has advantage of using interconnection between states.

Eg. Direct Evaluation:



→ Advantage of direct evaluation

- easy understand
- doesn't require T, R
- eventually computes correct average value, Using Sample transitions.

→ Disadvantage

- wastes information about state connection
- Each state must learn separately
- long time to learn.

→ Can't use Policy Evaluation

→ need T and R

→ Static Iterative Value Computing (Model-based)

- We want to improve our estimate of V by computing these averages:

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

$$V_0^{\pi}(s) = 0$$

- Idea: Take samples of outcomes s' (by doing the action!) and average

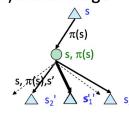
$$\text{sample}_1 = R(s, \pi(s), s'_1) + \gamma V_k^{\pi}(s'_1)$$

$$\text{sample}_2 = R(s, \pi(s), s'_2) + \gamma V_k^{\pi}(s'_2)$$

$$\dots$$

$$\text{sample}_n = R(s, \pi(s), s'_n) + \gamma V_k^{\pi}(s'_n)$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i \text{sample}_i$$



Almost! But we can't
rewind time to get sample
after sample from state s.

→ Temporal Difference learning

- Big idea: learn from every experience!

- Update $V(s)$ each time we experience a transition (s, a, s', r)
- More likely outcomes s' will contribute updates more often

- Temporal difference learning of values

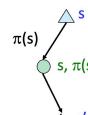
- Policy still fixed, still doing evaluation!

- Move values toward value of whatever successor occurs using exponential running average

Sample of $V(s)$: $\text{sample} = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to $V(s)$: $V^{\pi}(s) \leftarrow (1 - \alpha)V^{\pi}(s) + (\alpha)\text{sample}$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(\text{sample} - V^{\pi}(s))$



基本原理

TD学习通过观察环境反馈（即状态转移和奖励）来更新价值函数的估计。具体来说，当代理从状态 s 通过行为 a 转移到状态 s' 并接收到奖励 r 时，TD学习利用这一经验来更新状态 s 的价值估计 $V(s)$ ，更新公式通常表示为：

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$$

其中，

$\cdot V(s)$ 是状态 s 的当前价值估计。

$\cdot \alpha$ 是学习率，决定了新信息覆盖旧信息的速度。

$\cdot r$ 是从状态 s 转移到 s' 时获得的即时奖励。

$\cdot \gamma$ 是折扣因子，用于计算未来奖励的当前价值，表示了未来奖励的重要性。

$\cdot V(s')$ 是状态 s' 的价值估计。

$\cdot [r + \gamma V(s') - V(s)]$ 是时序差分误差，表示了估计值与实际观察到的奖励之间的差异。

TD学习的特征

• TDQ: 最基本的时序差分学习方法，根据当步的观察来更新价值估计。

• SARSA (状态-动作-奖励-状态-动作)：一种在策略上进行时序差分更新的方法，用于学习动作价值函数 $Q(s, a)$ ，它考虑了当步从动作 a 下一步态为 s' 时选择的动作。

• Q-Learning：另一种学习动作价值函数 $Q(s, a)$ 的方法，它在更新时采用了一种“贪心”策略，选择使得 $Q(s, a)$ 最大的 a' ，而不当做 \downarrow 实际选择的是什么动作。

Temporal Difference :

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

• 当前状态 s ：机器人在迷宫的起点，假设其价值估计 $V(s) = 5$ ，

• 下一个状态 s' ：机器人向第一步到达的新位置，价值估计 $V(s') = 10$ 。

• 即时奖励 r ：移动到下一个状态的奖励为 $+2$ 。

• 学习率 α ：选择 0.1 表示新信息的更新权重。

• 折扣因子 γ ：选择 0.9 ，表示对未来奖励的考虑程度。

更新过程

按照公式： $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$

我们可以将已知值代入：

$$V(s) \leftarrow 5 + 0.1[2 + 0.9 \times 10 - 5]$$

计算得到：

$$V(s) \leftarrow 5 + 0.1[2 + 9 - 5]$$

$$V(s) \leftarrow 5 + 0.1 \times 6$$

$$V(s) \leftarrow 5 + 0.6$$

$$V(s) \leftarrow 5.6$$

→ Exponential Moving Average

Exponential Moving Average

• Exponential moving average

- The running interpolation update: $\bar{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$
- Makes recent samples more important:

$$\bar{x}_n = \frac{x_n + (1 - \alpha) \cdot x_{n-1} + (1 - \alpha)^2 \cdot x_{n-2} + \dots}{1 + (1 - \alpha) + (1 - \alpha)^2 + \dots}$$

- Forgets about the past (distant past values were wrong anyway)

- Decreasing learning rate (alpha) can give converging averages

公式: $\hat{x}_n = (1 - \alpha) \cdot \bar{x}_{n-1} + \alpha \cdot x_n$

这里, \bar{x}_n 表示第 n 项的移动平均值, α 是平滑因子 (通常介于 0 和 1 之间), x_n 是第 n 项的实际值, \bar{x}_{n-1} 是第 $n - 1$ 项的移动平均值。

• 更新原理: 当新数据点 x_n 来临时, EMA 会按照比例 α 融入新的数据点, 同时保留原有的平均值的一部分。比例 α 决定了新数据对当前平均值的影响程度, 以及旧数据点在新平均值中的权重。

• 重视近期数据: 由于权重随时间呈指数级衰减, EMA 对最近的观察结果赋予了更高的权重, 使其对变化更加敏感。

• 忘记过去: 过去的观察值 (特别是较远的过去) 对当前平均值的影响逐渐减弱, 直至趋近于 0。这意味着随着时间推移, 过去的数据对当前估计的贡献越来越小。

• 收敛性: 当 α 较小时, EMA 会更平滑且对新数据的反应更慢, 这有助于平均值收敛到某个稳定的水平。随着 α 的减小, 新的观察值对移动平均值的贡献减小, 使移动平均值依赖更多的历史信息。

例子

假设我们正在跟踪一项指标 (比如温度或股票价格), 我们希望使用 EMA 来平滑短期波动, 并识别其长期趋势。

• 如果 α 设为 0.1, 新的数据点 x_n 只会占当前平均值的 10%, 而之前的平均值 \bar{x}_{n-1} 占 90%。

• 如果我们在第 n 次观测到值为 100, 而之前的 EMA 值 \bar{x}_{n-1} 为 90, 则更新后的 EMA 值 \bar{x}_n 为: $\bar{x}_n = (1 - 0.1) \cdot 90 + 0.1 \cdot 100 = 91$

这样, 新的 EMA 值就稍微增加, 反映了量 ↓ 测值的影响。

→ Q-Value Iteration

价值迭代 (Value Iteration)

价值迭代是一种求解马尔可夫决策过程 (MDP) 的动态规划方法, 其目标是找到每个状态的最优值函数 $V^*(s)$, 从而确定最优策略。它是一个迭代过程, 具体步骤如下:

1. 初始值: 开始时, 所有状态的值 $V_0(s)$ 初始化为 0 (任何初始估计)。
2. 迭代更新: 对于每一次迭代 k , 计算下一个深度的值 $V_{k+1}(s)$ 为所有动作 a 的最大期望总回报: $V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$ 这里, $T(s, a, s')$ 是状态转移概率, $R(s, a, s')$ 是即时奖励, γ 是折扣因子, $V_k(s')$ 是上一次迭代后状态 s' 的值。

Q值迭代 (Q-Value Iteration) Solve MDP and complete environment

虽然价值迭代可以确定最优值, 但 Q 值迭代提供了直接评估每个状态-动作对的优势, 从而更加直接地推导出最优策略。Q 值迭代的步骤如下:

1. 初始值: 所有状态-动作对的 Q 值 $Q_0(s, a)$ 初始化为 0。
2. 迭代更新: 对于每一次迭代 k , 更新状态-动作对的 Q 值 $Q_{k+1}(s, a)$ 为所有可能的后续状态 s' 的总回报的期望值:
$$Q_{k+1}(s, a) \leftarrow \max_{s'} T(s, a, s') [R(s, a, s') + \gamma \max_{a'} Q_k(s', a')] \quad \boxed{\text{Q-Value Iteration}}$$
 不同于价值迭代只需要考虑后续状态的值, Q 值迭代在考虑后续状态时, 还需要选取最大化 $Q_k(s', a')$ 的动作 a' 。

Q学习 (Q-Learning) Unknown MDP

Q 学习是基于样本的 Q 值迭代方法。它不需要环境的模型 (即转移概率 T 和奖励函数 R), 而是直接从代理与环境的交互中学习 Q 值。

学习过程

1. 从样本中学习: 当代理在状态 s 采取动作 a 后, 观察到下一个状态 s' 和即时奖励 r , 代理就接收到了一个样本 (s, a, s', r) 。

2. 更新 Q 值: 代理考虑旧的 Q 值估计 $Q(s, a)$ 和新的样本估计。新样本估计由即时奖励和折扣后的最大 Q 值的构成, 计算公式为:

$$Q(s, a) \leftarrow R(s, a, s') + \gamma \max_{a'} Q(s', a') \quad \boxed{\text{J - Q-update}}$$

3. 结合新旧估计: 然后, 代理结合旧的 Q 值和新样本估计来更新 Q 值, 更新公式为:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[\text{sample}]$$

其中, α 是学习率, 它决定了新样本在更新中的权重。

Q-learning:

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[\text{sample}]$$

New old new

→ Q-learning Properties

离策策略学习 (Off-Policy Learning)

在离策策略学习中, 代理可以根据从探索中获得的经验来学习最优策略, 即使这些经验来源于次优策略。这意味着, 代理在学习过程中可以尝试各种不同的动作, 包括那些不是当前最佳选择的动作, 而最终还是能够找到最佳策略。

注意事项 (Caveats)

- 足够的探索: 代理需要有足够的机会尝试各种动作, 以获得环境的全面理解。
- 学习率调整: 学习率 (α) 需要随着时间推移而逐渐减小, 确保学习过程最终能够稳定下来。如果学习率降低得太快, 代理可能没有足够的机会从新的探索中学习; 如果学习率降低得太慢, 学习过程可能会变得很震荡, 难以收敛。
- 动作选择: 在理论上, 只要这些条件得到满足, 最终代理如何选择动作 (探索策略) 并不会影响它学到 Q 值的能力。

Q-learning Converges to optimal Policy even if you're acting suboptimally.

$$\alpha = 1 \rightarrow \text{old} = 0,$$

→ How to Explore

如何探索?

- 探索策略：在强化学习中，代理必须在探索（尝试新的或不熟悉的动作）和利用（选择已知的最佳动作）之间找到平衡。探索是必要的，因为代理需要收集有关环境的足够信息，以确保找到最优策略。

强制探索的几种方案

- 最简单的方案：随机动作 (ϵ -greedy)
 - 在每个时间步，进行一次决策，如同抛一枚硬币。
 - 以较小的概率 ϵ 采取随机动作。
 - 以较大的概率 $1 - \epsilon$ 根据当前策略采取动作。

随机动作的问题

- 探索与学习结束：虽然通过随机选择动作可以确保探索整个动作空间，但这也可能导致代理在学习完成后仍然进行无目的的探索，也被称为“抖动”(thrashing)。
- 解决方案之一：随着时间的推移降低 ϵ 的值，从而减少随机动作的比例，使代理更多地利用其学到的策略。
- 另一种解决方案：实现更复杂的探索函数，这可能包括对环境的结构进行建模，或者采用上下文相关的探索策略。

(E) $\left\{ \begin{array}{l} \text{For new strategies} \\ \text{Avoid local Optima} \\ \text{learn Environment} \end{array} \right.$

In Short Summary

- 如果我们知道MDP：
 - 计算 V^* , Q^* , π^* 的精确值，其中 V^* 是最优值函数， Q^* 是最优的动作价值函数， π^* 是最优策略。
 - 评估一个固定的策略 π 。
- 如果我们不知道MDP：
 - 我们可以估计MDP，然后解决它。
 - 我们可以为一个固定的策略 π 估计 V 。
 - 我们可以在执行探索策略的同时，估计最优策略的 $Q(s, a)$ 。

技术：

- 离线MDP解决方案：
 - 价值迭代 (Value Iteration)：一种动态规划算法，通过迭代更新来计算每个状态的最优值函数。
 - 策略评估 (Policy Evaluation)：评估一个给定策略的价值函数，即计算在该策略下每个状态的预期回报。
- 强化学习 (Reinforcement Learning)：
 - 模型基RL (Model-based RL)：在强化学习中，如果我们对MDP有一个模型（例如，我们知道状态转移概率和奖励函数），我们可以使用这个模型来预测并计划最优动作。
 - 无模型：价值学习 (Model-free: Value learning)：在不直接知道MDP模型的情况下学习每个状态的价值。
 - 无模型：Q学习 (Model-free Q-learning) ↓：无需知道状态转移概率和奖励函数，直接从经验中学习动作的价值。

→ Exploration Functions

→ When to explore:

Random act: explore a fix amount

Better idea: Explore area whose badness is not yet established

探索函数

探索函数是一种计算给定状态-动作对应优势（或价值）的方法，它考虑到了以下两个因素：

• 价值估计 u : 代表了对某个状态-动作对的当前价值估计。

• 访问次数 n : 状态-动作对被探索的次数。

探索函数的一个示例是 $f(u, n) = u + k/n$ ，其中 u 是价值估计， n 是访问次数， k 是一个正的常数。这个函数随着某个状态-动作对的访问次数的增加而减小，鼓励代理探索那些不经常访问的状态-动作对。

Q值更新

幻灯片中展示了两种Q值更新的方法：

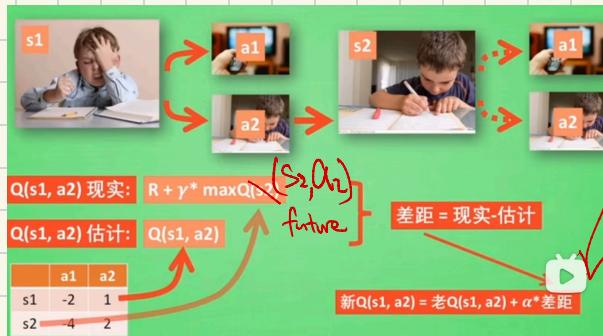
• 常规Q值更新：使用标准的Q学习公式来更新Q值 不涉及探索函数。

$$Q(s, a) \leftarrow R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

• 修改后的Q值更新：在更新过程中使用探索函数来调整Q值，以鼓励探索。

$$Q(s, a) \leftarrow R(s, a, s') + \gamma \max_{a'} f(Q(s', a'), N(s', a'))$$

这里， $R(s, a, s')$ 是从状态 s 执行动作 a 转移到状态 s' 得到的即时奖励， γ 是折扣因子，用于调整未来奖励的当前价值。 f 是探索函数。 $N(s', a')$ 是状态 s' 和动作 a' 的访问计数。



OR

Q-learning:

$$Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha [Sample]$$

Difference

weak update

→ Large number of State

基本的 Q-Learning

- Q值表：在基本的Q学习中，我们为每个可能的状态-动作对维护一个Q值表。这个表可以用来在任何给定状态下选择最佳动作。

现实问题

- 状态太多：在实际情况中，状态的数量可能是如此之多，以至于我们不可能在训练过程中访问到每一个状态。
- 内存问题：即使我们能访问到每个状态，将这么大的Q值表保存在内存中也是不切实际的。

解决方案：泛化 Generalizing

- 泛化：由于上述问题，我们需要采用泛化的策略：
 - 学习一小部分状态：从经验中学习一些少量的训练状态。
 - 经验泛化：将从这些状态中学到的经验泛化到新的、相似的情况。
 - 机器学习中的基本理念：这种从有限样本中学习并泛化到新情况的方法是机器学习的一个基本理念。

→ Linear Value Function

线性值函数

Feature

- 特征表示：**我们可以用特征向量来表示一个状态或状态-动作对（也称为Q状态）。这些特征是一些函数，它们将状态（或状态-动作对）映射到实数值，通常是0或1。
- 线性组合：**使用特征表示，我们可以将Q函数（或值函数）写成特征函数 f_i 和相应权重 w_i 的线性组合。
 - 值函数 $V(s)$ 可以表示为 $V(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$ 。
 - Q函数 $Q(s, a)$ 可以表示为 $Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$ 。

优势

- 强大的数字总结：我们的经验可以被几个有力的数字（即权重）总结起来，这些数字与各特征结合后代表了状态或状态-动作对的价值。

缺点

- 特征共享的问题：不同的状态可能共享某些特征，但它们的价值却可能非常不同。线性模型可能无法捕捉这种差异，因为它无法表示特征之间的复杂相互作用。

→ Feature-based Representation.

基于特征的表示

- 解决方案：**使用特征向量来描述状态，这些特征是从状态映射到实数（通常是0或1）的函数，用于捕捉状态的重要属性。
- 示例特征：**
 - 最近的幽灵距离。
 - 最近的点（或食物）距离。
 - 幽灵的数量。
 - 点到点的倒数平方 $(1/\text{dist to dot})^2$ 。
 - 判断吃豆人是否在通道中（二元特征，是或否）。
- Q状态描述：**也可以用特征来描述一个Q状态 (s, a) ，例如让吃豆人接近食物的行为。
- 从状态评估特征征兆：**这表明我们正在从基于具体状态的评估转向基于特征的评估，这有助于泛化和减少需要直接学习的统计数量。

泛化的意义

泛化允许学习算法从有限的经验中推广到新的相似情况。这是机器学习的一个基本理念。在复杂环境中，如视频游戏或现实世界任务，可能存在数百万个状态。为每个状态维护一个Q值是不现实的。相反，基于特征的表示可以捕捉决定最佳行动所需的关键信息。

例如，如果吃豆人游戏中的吃豆人在学习如何避开幽灵，那么知道每个具体位置的确切Q值不如知道与幽灵的相对距离更有用。通过学习这些关键特征与奖励之间的关系，学习代理可以更快地泛化并在新环境中做出决策。

→ Approximate Q-learning

近似 Q-Learning

- 线性Q函数：**使用线性组合的特征表示，Q值可以写成特征函数 $f_i(s, a)$ 和相应权重 w_i 的线性组合。 $\hat{Q}(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \dots + w_n f_n(s, a)$

Q-Learning with Linear Q-Functions

- 转换：**每次代理从状态 s 采取动作 a 并观察到即时奖励 r 和新状态 s' 时，它会经历一个状态转换。 $s \rightarrow a \rightarrow s' \rightarrow r, s'$
- 差值：**计算前一个Q值和当前Q值之间的差异（TD误差）。
- 差值更新：**然后用这个差异来更新 s, a 的Q值。 $Q(s, a) \leftarrow Q(s, a) + \alpha[\text{difference}] \rightarrow \text{Exact } Q$
- 权重更新：**同时更新影响Q值的特征权重。 $w_i \leftarrow w_i + \alpha[\text{difference}] f_i(s, a) \rightarrow \text{Approx } Q$

直观解释

- 调整激活特征的权重：当状态 s 下采取动作 a 时，如果发生了意外的坏事，可以调整与当前状态-动作对应的特征的权重。

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

Es.

Agent → ghost distance $f_1(s, a)$, w_1
 → dot distance $f_2(s, a)$, w_2

Initial:

$$w_1 = 2, w_2 = 5$$

$$f_1(s, a) = 3, f_2(s, a) = 2$$

$$\alpha = 0.1$$

weight update?

$$w_1 \leftarrow w_1 + \alpha \cdot \text{difference} \cdot f_1(s, a)$$

$$= 2 + 0.1 \cdot (4 - 2) \cdot 3 = 2.6$$

$$w_2 \leftarrow w_2 + \alpha \cdot \text{difference} \cdot f_2(s, a)$$

$$= 5 + 0.1 \cdot (4 - 2) \cdot 2 = 5.2$$

当前 Q 值:

$$\hat{Q} = w_1 f_1(s, a) + w_2 f_2(s, a)$$

$$= 2 \cdot 3 + 5 \cdot 2 = 16$$

Transition:

$$s \rightarrow a \rightarrow r = 4, s' \rightarrow \text{ghost}$$

$$\text{difference} = (r + \gamma \max Q(s', a)) - Q(s, a)$$

$$= (4 + 0.9 \cdot 14) - 16$$

$$= 0.6$$

