绪论、马尔可夫决策过程、动态规划

绪论

Basic Information

- 在学习过程中个人做出的每一次尝试都是是一次**决策** （ decision ），每一次决策都会带来相应的后果（好的结果：奖励 reward；坏的结果：惩罚 punishment）。
- **序列决策**（ sequential decision making ）过程：最终通过一次次的决策来实现目标，这个目标通常是以最大化累积的奖励来呈现的。
- 为什么学习强化学习？对于任意问题，只要能够建模成序列决策问题或者带有鲜明的试错学习特征，就可以使用强化学习来解决，并且这是截至目前最为高效的方法之一。
- 强化学习的广泛应用领域：游戏、机器人、金融、自动驾驶、推荐系统、交通派单、广告投放、ChatGPT等。

强化学习方向概述

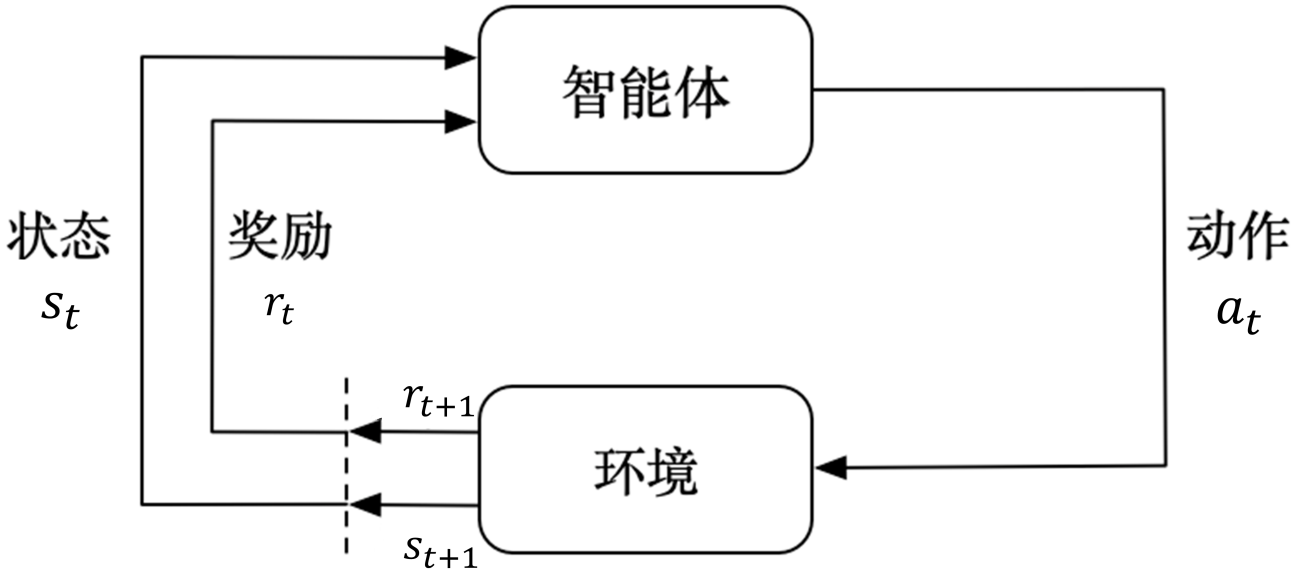
- 多智能体强化学习
  - 环境的状态不仅由智能体的动作决定，还受到其他智能体的动作的影响。
  - 智能体之间可能需要进行通信以合作或竞争，如何高效地通信并从信号中学习是一个难题。
  - 还存在信誉分配问题，在多智能体的合作任务中，确定每个智能体对于整体目标的贡献（或责任）是一个挑战。
- 从数据中学习（从演示中学习）
  - 模仿学习
  - 逆强化学习
  - 从人类反馈中学习
  - 离线强化学习
  - 世界模型
  - ……
- 探索策略
  - 在探索（尝试未知的动作，从而会获得更多的奖励或更高的惩罚）和利用之间做出权衡。
  - 目前比较常用的方法有 epsilon−greedy和置信上界（ upper confidence bound ）等等。
  - 避免局部最优问题，从而提高智能体的鲁棒性，近年来也有研究结合进化算法来提高探索的效率，例如 NEAT（neuro evolution of augmenting topologies）和 PBT（population based training）等算法。

- 实时环境
  - 在离线环境中进行训练，然后将训练好的模型部署到在线环境中进行决策
  - 缺陷：离线环境和在线环境之间可能存在着分布漂移，即两个环境的状态分布不同，这就导致了训练好的模型在在线环境中可能会出现意外。
  - 世界模型的思路是将环境分为两个部分，一个是世界模型，另一个是控制器。世界模型的作用是预测下一个状态，而控制器的作用是根据当前的状态来决策动作。这样就可以在离线环境中训练世界模型，然后将世界模型部署到在线环境中进行决策。**世界模型的预测误差会导致控制器的决策出错，因此如何提高世界模型的预测精度也是一个难题。**
- 多任务强化学习
  - 目前比较常用的方法：
    - 联合训练（ joint training ）：将多个任务的奖励进行加权求和，然后通过强化学习来学习一个策略。
    - 分层强化学习（ hierarchical reinforcement learning ）：将多个任务分为两个层次，一个是高层策略（决策当前的任务），另一个是低层策略（决策当前任务的动作）。
    - ……

马尔可夫决策过程

- 环境：智能体的交互主体。具有状态。
- 智能体：决策者和学习者。执行动作时会观测到信息，也就是状态；执行完动作后会得到反馈，纠正下一次动作。

下图描述了马尔可夫决策过程中智能体与环境的交互过程。智能体每一时刻都会接收环境的状态，并执行动作，进而接收到环境反馈的奖励信号和下一时刻的状态。



在强化学习中我们通常考虑的是有限马尔可夫决策过程（ Finite MDP ），即 t 是有限的，这个上

限一般用 T 表示，也就是当前交互过程中的最后一个时步或最大步数，从 t=0 和 t=T 这一段时步我们称为一个回合（ episode ），比如游戏中的一局。

马尔可夫性质

在给定历史状态

$$s_0, s_1, \ldots, s_t$$

的情况下，某个状态的未来只与当前状态s_t有关，与历史的状态无关。

实际问题中，有很多例子不仅取决于当前状态，还依赖于历史状态。当然这也并不意味着完全不能用强化学习来解决，实际上此时我们可以用深度学习神经网络来表示当前的棋局，并用蒙特卡洛搜索树等技术来模拟玩家的策略和未来可能的状态，来构建一个新的决策模型，这就是著名的 AlphaGO 算法。

回报

累积奖励称为回报：

$$G_t = r_{t+1} + r_{t+2} + \ldots + r_T$$

这个公式其实只适用于有限步数的情境，例如玩一局游戏，无论输赢每回合总是会在有限的步数内会以一个特殊的状态结束，这样的状态称之为终止状态。
但也有一些情况是没有终止状态的。为了解决这个问题，我们引入一个折扣因子（discount factor）γ，并可以将回报表示为

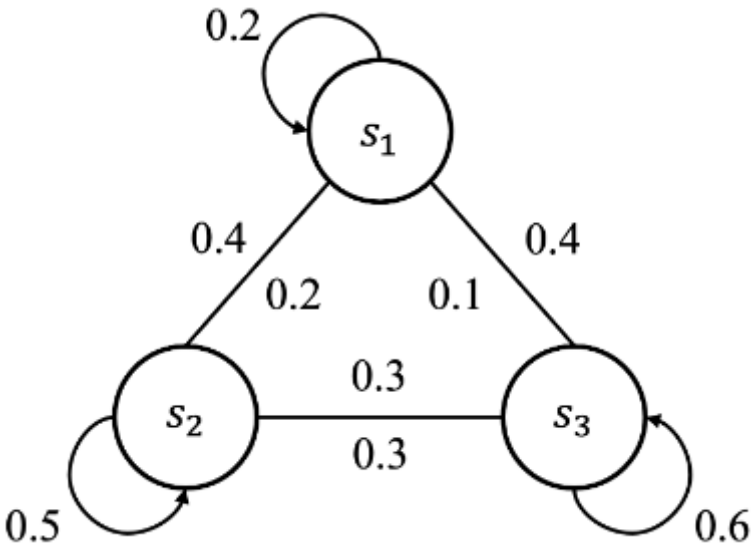
$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots$$

其中γ 取值范围在 0 到 1 之间，它表示了我们在考虑未来奖励时的重要程度，控制着当前奖励和未来奖励之间的权衡。同时，它建立了当前时步的回报与下一个时步回报的关联：

$$G_t = t_{t+1} + \gamma G_{t+1}$$

状态转移矩阵

我们可以用一种状态流向图来表示智能体与环境交互过程中的走向。**有时智能体和环境的角色是能相互对调的，只要能各自建模成马尔可夫决策过程即可。**



整张图表示了马尔可夫决策过程中的状态流向。严格意义上来讲，这张图中并没有完整地描述出马尔可夫决策过程，因为没有包涵动作、奖励等元素，所以一般我们称之为**马尔可夫链（Markov Chain）**，又叫做离散时间的马尔可夫过程（Markov Process）。
状态转移概率（State Transition Probability）

$$P_{ss'} = P(S_{t+1} = s' | S_t = s)$$

用矩阵形式表示出来就是**状态转移矩阵（State Transition Matrix）**。**状态转移矩阵是环境的一部分**，跟智能体是没什么关系的，而智能体会根据状态转移矩阵来做出决策。
此外，在马尔可夫链（马尔可夫过程）的基础上增加奖励元素就会形成**马尔可夫奖励过程（Markov reward process, MRP）**，在马尔可夫奖励过程基础上增加动作的元素就会形成马尔可夫决策过程，也就是强化学习的基本问题模型之一。
马尔可夫决策过程的描述

$$< S, A, R, P, \gamma >$$

本章练习题

1. 强化学习所解决的问题一定要严格满足马尔可夫性质吗？请举例说明。
    1. 不一定。历史发展的路径会影响当下的决策，例如在扑克牌游戏中，出牌顺序的不同会影响当下智能体的判断。
2. 马尔可夫决策过程主要包含哪些要素?
    1. 上文提到的五元组
3. 马尔可夫决策过程与金融科学中的马尔可夫链有什么区别与联系?
    1. 前者有更多元素，例如奖励和动作。
    2. 马尔可夫链是一种具有马尔可夫性质的随机过程；马尔可夫决策过程可的目标是寻找一个最大化累积奖励的最优策略。

动态规划

在强化学习中，动态规划被用于求解值函数和最优策略。常见的动态规划算法包括值迭代（value iteration, VI）、策略迭代（policy iteration, PI）和 Q-learning 算法等。

我的评价是：dp虐我千百遍，我待dp如初学：）再不刷题我是🐶：（
刷leetcode的（迄今为止发现的最好的）（python可）材料 https://labuladong.github.io/algo/di-er-zhan-a01c6/dong-tai-g-a223e/dong-tai-g-1e688/

Basic Information

编程思路可以概括为：

- 找出子问题以及和母问题的关系
- 特殊处理一些约束/边界条件
- 如果是计算量比较大/重复多的问题，可以把之前的计算结果存成meme

动态规划问题有三个性质，最优化原理、无后效性和有重叠子问题。
无后效性指的是即某阶段状态一旦确定，就不受这个状态以后决策的影响。也就是说，某状态以后的过程不会影响以前的状态，只与当前状态有关，这其实就是前面所说的马尔可夫性质。
而最优化原理是指，如果问题的最优解所包含的子问题的解也是最优的，就称该问题具有最优子结构，即满足最优化原理。

我们知道当前时步的回报跟下一个时步的回报是有关系的，故可以在不同时步上通过某种方法最大化对应时步的回报来解决马尔可夫决策问题，我们要解决G_{t+1}的问题，可以一次拆分成解决$G_t, \dots G_2, G_1$的问题，这其实就满足动态规划性质中的最优化原理。

状态价值函数和动作价值函数

——策略迭代和价值迭代算法的铺垫。

我们认为每一个状态都具有价值，状态价值函数：

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

从特定状态出发，按照某种策略π进行决策所能得到（折扣后的）回报期望值，
想起了在风险中性概率下计算B-S framework欧式期权价格的过程……

引入的动作也有价值，动作价值函数：

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid s_t = s, a_t = a \right]$$

动作价值函数和状态价值函数的关系：（全期望公式）

$$V_\pi(s) = \sum_{a \in A} \pi(a \mid s) Q_\pi(s, a)$$

其中$\pi(a|s)$表示策略函数，一般指在状态s下执行动作a的概率分布。这个公式的意思就是在给定状态s的情况下，智能体所有动作的价值期望就等于该状态的价值。

Bellman Equation

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} \mid s_t = s\right] + \gamma \mathbb{E}\left[R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots \mid S_t = s\right] \\
&= R(s) + \gamma \mathbb{E}\left[G_{t+1} \mid S_t = s\right] \\
&= R(s) + \gamma \mathbb{E}\left[V_\pi\left(s_{t+1}\right) \mid S_t = s\right] \\
&= R(s) + \gamma \sum_{s' \in S} P\left(S_{t+1} = s' \mid S_t = s\right) V_\pi\left(s'\right) \\
&= R(s) + \gamma \sum_{s' \in S} p\left(s' \mid s\right) V_\pi\left(s'\right)
\end{aligned}
$$

贝尔曼方程将前后两个状态之间联系起来，以便于递归地解决问题。

类似地，动作价值函数贝尔曼方程推导为：

$$
Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} p\left(s' \mid s, a\right) \sum_{a' \in A} \pi\left(a' \mid s'\right) Q_\pi\left(s', a'\right)
$$

状态价值函数的定义就是按照某种策略π进行决策得到的累积回报期望，换句话说，在最优策略下，状态价值函数也是最优的，相应的动作价值函数也最优。我们的目标是使得累积的回报最大化，那么最优策略下的状态价值函数可以表示为：

$$
\begin{aligned}
V^*(s) &= \max_a \mathbb{E}\left[R_{t+1} + \gamma V^*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right] \\
&= \max_a \sum_{s', r} p(s', r|s, a)[r + \gamma V^*(s')]
\end{aligned}
$$

这个公式叫做**贝尔曼最优方程（Bellman optimality equation）**。对于动作价值函数也是同理，如下

$$
\begin{aligned}
Q^*(s, a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} Q^*\left(S_{t+1}, a'\right) \mid S_t = s, A_t = a\right] \\
&= \sum_{s', r} p\left(s', r \mid s, a\right)\left[r + \gamma \max_{a'} Q^*\left(s', a'\right)\right]
\end{aligned}
$$

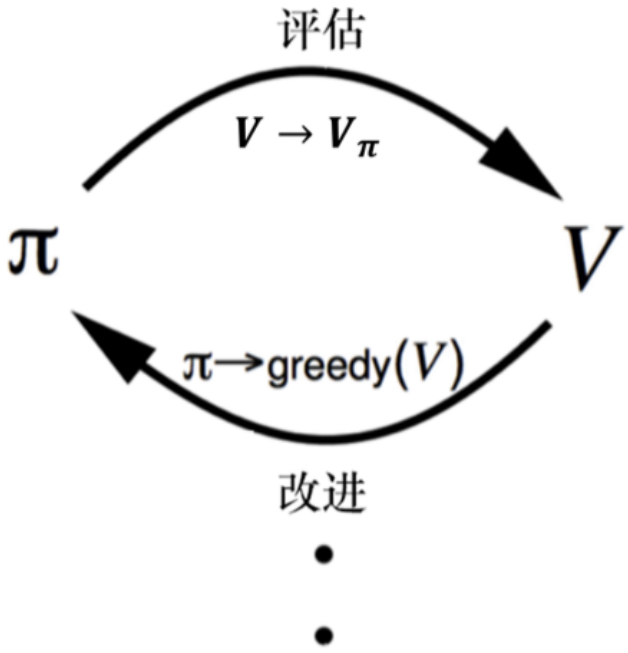
策略迭代

尽管在最优策略下，对应的状态和动作价值函数也都是最优的，但是实际求解中在优化策略的过程中，同时我们还需要优化状态和动作价值函数——多目标优化问题。
策略迭代算法的思路：

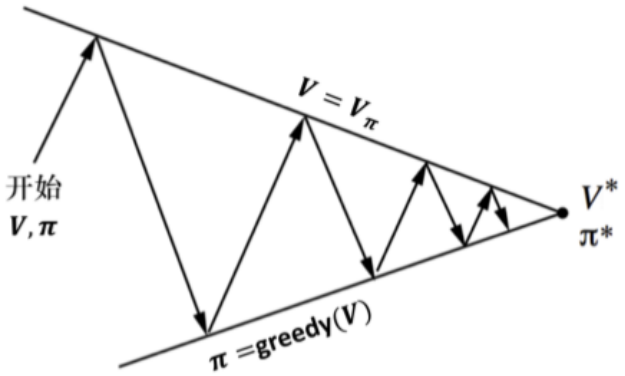
- **策略估计**：首先固定策略不变，然后估计对应的状态价值函数
- **策略改进**：根据估计好的状态价值函数，结合策略推算出动作价值函数，并对其优化然后进一步改进策略
  在策略改进的过程中一般是通过贪心策略来优化的，即定义策略函数为：

$$\pi(a|s) = \max_a Q(s,a)$$

然后在策略改进时选择最大的$Q(s,a)$值来更新。在一轮策略估计和改进之后，又会进入新的一轮策略估计和改进，直到收敛为止。



(a) 策略迭代的步骤



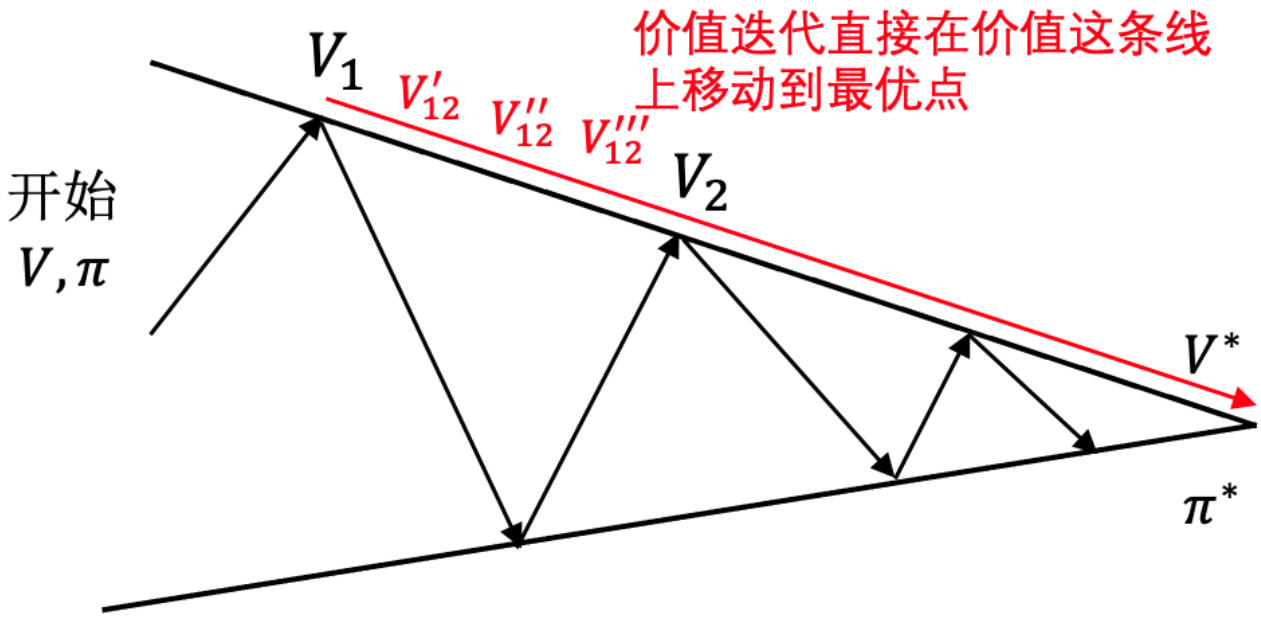
(b) 策略迭代的过程

价值迭代

直接根据以下公式：

$$V(s) \leftarrow \max_{a \in A} \left(R(s,a) + \gamma \sum_{s' \in S} p\left(s' \mid s, a\right) V\left(s'\right) \right)$$

它首先将所有的状态价值初始化，然后不停地对每个状态迭代，直到收敛到最优价值，并且根据最优价值推算出最优策略。



在策略迭代算法中借助了策略这条线跳过了中间的$V_{12}', V_{12}'', V_{12}'''$这些点，而在价值迭代算法的时候会经过价值这条线上的所有点，直到最优，从这个角度来看策略迭代算法是要比价值迭代更快的。

本章练习题

- 动态规划问题的主要性质有哪些?
  - 最优化原理、无后效性和有重叠子问题
- 状态价值函数和动作价值函数之间的关系是什么?
  - 在给定状态s的情况下，智能体所有动作的价值期望就等于该状态的价值。
- 策略迭代和价值迭代哪个算法速度会更快?
  - 策略迭代