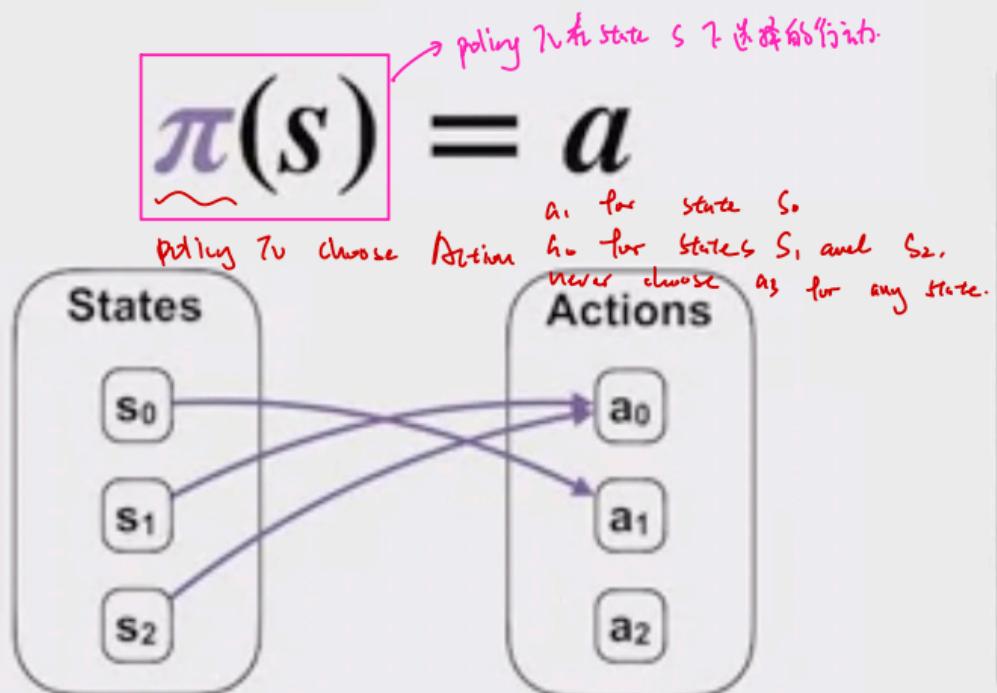


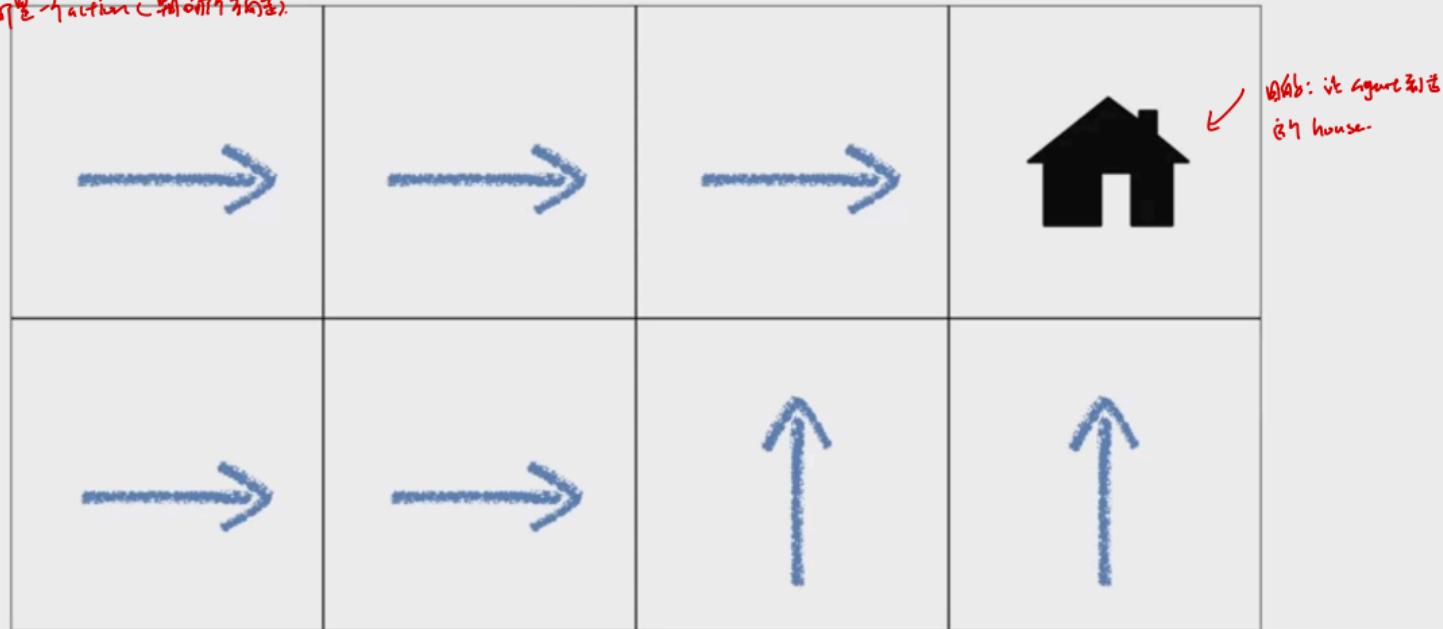
Deterministic policy notation



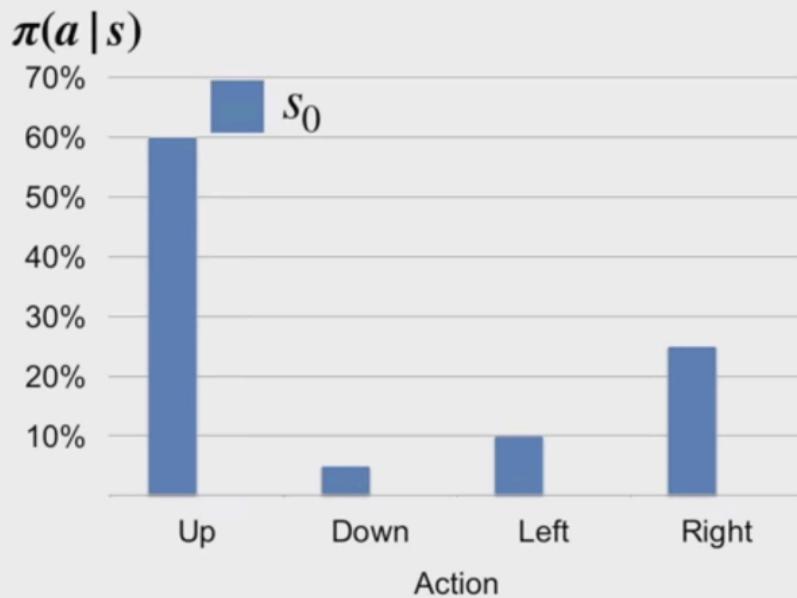
State	Action
s_0	a_1
s_1	a_0
s_2	a_0

Example: deterministic policy

每個點都是 action (朝向行動)



Stochastic policy notation



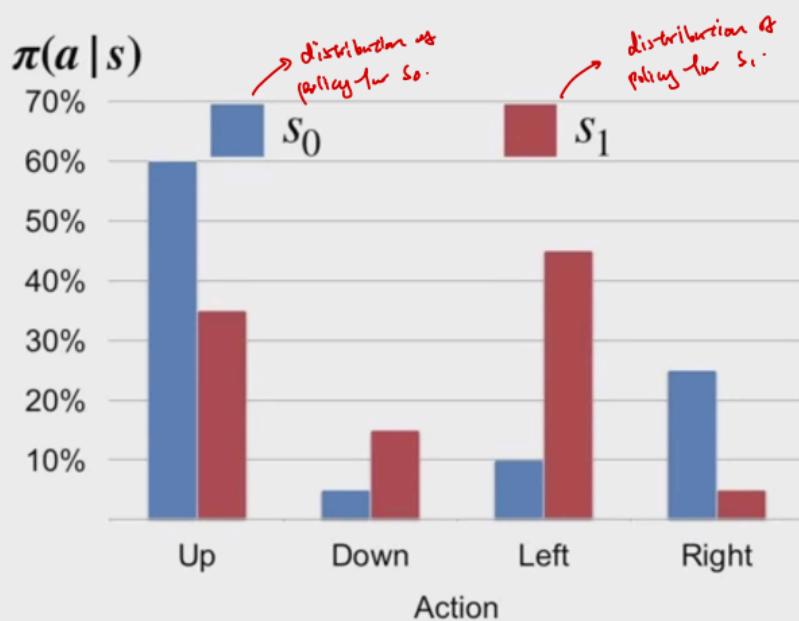
$$\sum_{a \in \mathcal{A}(s)} \pi(a | s) = 1$$

the sum of all action probabilities must be 1 in each state.

$$\pi(a | s) \geq 0$$

\Rightarrow each action probabilities must be non-negative.

Stochastic policy notation

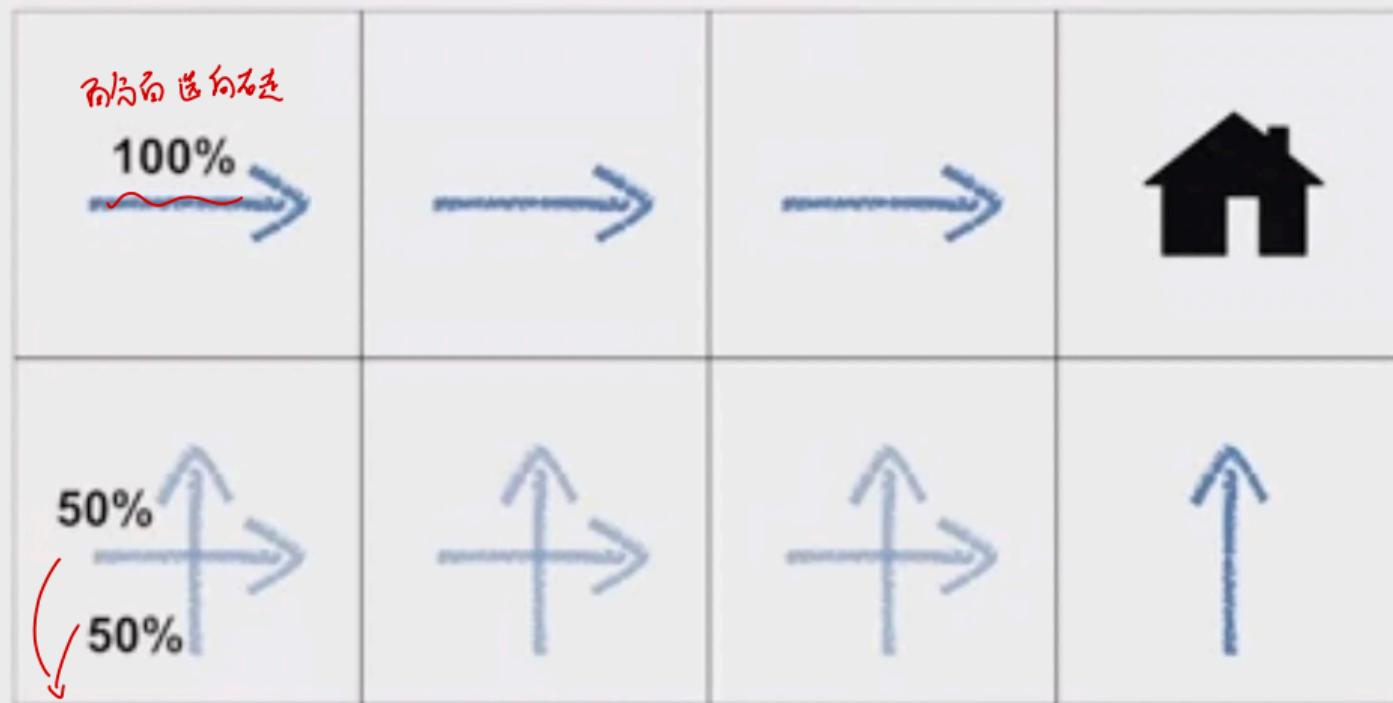


$$\pi(a | s)$$

$$\sum_{a \in \mathcal{A}(s)} \pi(a | s) = 1$$

$$\pi(a | s) \geq 0$$

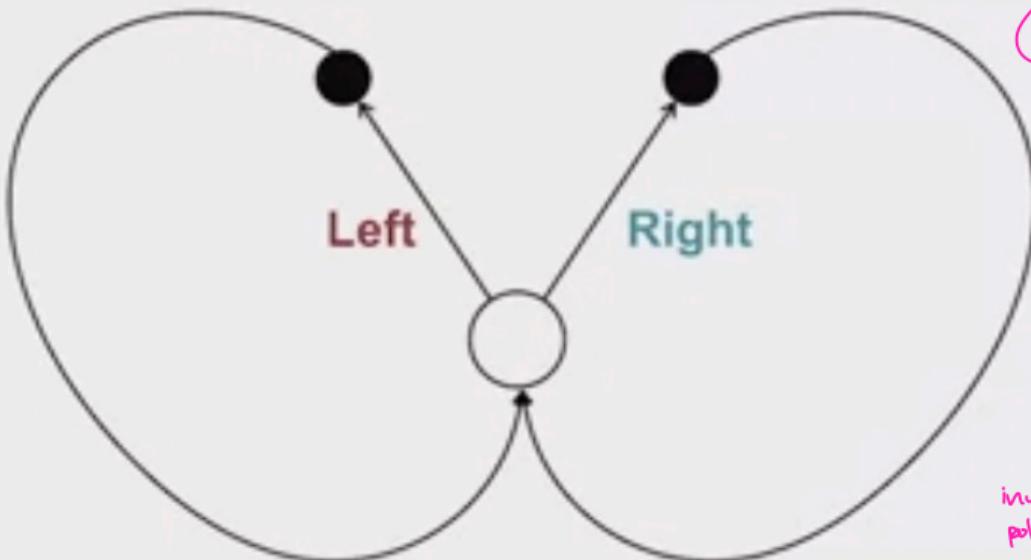
Example: stochastic policy



進向右走向右走有 50%

机率

Valid and invalid policies



if: In MDPs, we assume that the state includes all the information required for decision-making.

valid policy!! \Rightarrow the policy only depend on the current state, not on other things like the previous state.

1: **Left with 50% probability and Right with 50% probability**

LLRLRLRLRRR...

2: **Alternate Left and Right**

invalid policy!!

LRLRLRLRLRL...

MDP is (S, A, P, R, γ) but γ is last action!! Which is other than the state!! \Rightarrow γ is last action \Rightarrow the last action must be included in the state !!

Summary

- A **policy** maps the current **state** onto a set of **probabilities** for taking each **action**
- Policies can **only** depend on the current **state**

许多问题都涉及到某种程度的延迟回报 (delayed reward)。一个商店经理可以降低他们的价格，卖掉他们的全部库存，以使短期收益最大化。但从长远来看，他们会在需求量大的时候保持库存来销售，这样做会更好。在强化学习中，奖励抓住了短期收益 (short-term gain) 的概念。然而，目标是学习一个能在长期内获得最大回报的政策。价值函数 (Value functions) 正式说明了这一含义。

粗略地说，状态价值函数 (state value function) 是一个代理人从一个特定的状态开始可以期望得到的未来奖励 (future award)。更准确地说，状态价值函数是来自给定状态的预期回报 (expected return)。代理人的行为也将决定它能期待多少总奖励。因此，一个价值函数是针对给定的政策而定义的。下标 P_i 表示价值函数是以代理人根据 P_i 选择行动为条件的。同样地，期望值上的下标 P_i 表示期望值是针对政策 P_i 计算的。我们还可以定义一个行动价值函数。行动值描述了代理人第一次选择一个特定行动时发生的情况。更正式地说，一个状态的行动值是如果代理人选择行动 A ，然后遵循政策 P_i 的预期回报。

价值函数在强化学习中至关重要，它们允许代理人查询其当前情况的质量，而不是等待观察长期结果。其好处是双重的。首先，回报不是立即可用的，其次，由于政策和环境动态的随机性，回报可能是随机的。价值函数通过对回报的平均化来总结所有可能的未来。归根结底，我们最关心的是学习一个好的政策。价值函数使我们能够判断不同政策的质量。

例如，考虑一个玩国际象棋的代理人。国际象棋有一个偶发的MDP，状态是由棋盘上所有棋子的位置给出的，行动是合法的动作，当游戏以赢、输或平局结束时就会发生终止。我们可以将奖励定义为获胜时的加一，以及所有其他动作的零。这个奖励并不能告诉我们代理人在比赛中的表现如何，我们必须等到比赛结束后才能看到任何非零的奖励。价值函数告诉我们更多。状态值等于未来奖励的预期总和。由于唯一可能的非零奖励是获胜后的加一，所以状态值仅仅是如果我们遵循当前政策 P_i 的获胜概率。在这个双人游戏中，对手的行动是状态转换的一部分。例如，设想同时移动代理的棋子 (蓝色圈出) 和对手的棋子 (红色圈出)。这使棋盘进入一个新的状态，即 S_{prime} 。注意， S_{prime} 状态的价值低于 S 状态的价值。这意味着假设我们继续遵循政策 P_i ，我们在这个新状态下赢得游戏的可能性较小。

一个行动价值函数可以让我们评估在游戏的其余部分遵循政策 P_i 的情况下，每个可能的行动的获胜概率。为了建立一些直觉，让我们看一下一个简单的持续MDP。状态是由网格上的位置定义的，行动使代理人向上、向下、向左或向右移动。代理人不能离开网格，碰撞产生的奖励是 -1。大多数其他行动都没有产生任何奖励。然而，有两个特殊的状态，这些特殊的状态被标记为 A 和 B。在 A 状态下的每个动作都会产生正 10 的奖励，在 B 状态下的每个动作都会产生正 5 的奖励。记住，我们必须在弄清楚价值函数之前指定政策。让我们来看看均匀随机策略 (uniform random policy)。由于这是一个持续的任务，我们需要指定 Γ ，让我们选择 0.9。稍后，我们将学习几种计算和估计价值函数的方法。在右边，我们写了每个状态的值。首先，注意靠近底部的负值，这些值很低，因为代理人很可能在到达距离状态 A 和 B 之前就撞到了墙上。记住，A 和 B 都是这个MDP中唯一的正奖励来源。状态 A 的值最高，注意这个值小于 10，尽管从状态 A 出发的每一个行动都会产生一个加 10 的奖励，为什么？因为从 A 开始的每一次转换都会使代理人接近下墙，在下墙附近，随机政策很可能发生碰撞，获得负的奖励。另一方面，状态 B 的值略大于 5。这个从 B 的移动将代理人移到中间。在中间位置，代理人不太可能发生碰撞，并且接近高价值的状态 A 和 B。

现在，你应该明白，状态价值函数（state value function）指的是在特定政策下来自特定状态的预期回报，而行动价值函数（action value function）指的是在选择特定行动，然后遵循特定政策后来自特定状态的预期回报。

State-value functions

\Rightarrow expected reward from a given state under a specific policy.

Recall that

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Action-value functions

→ expected reward from a given state after taking a specific action, and then following a specific policy.

$$q_{\pi}(s, a) \doteq \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a]$$

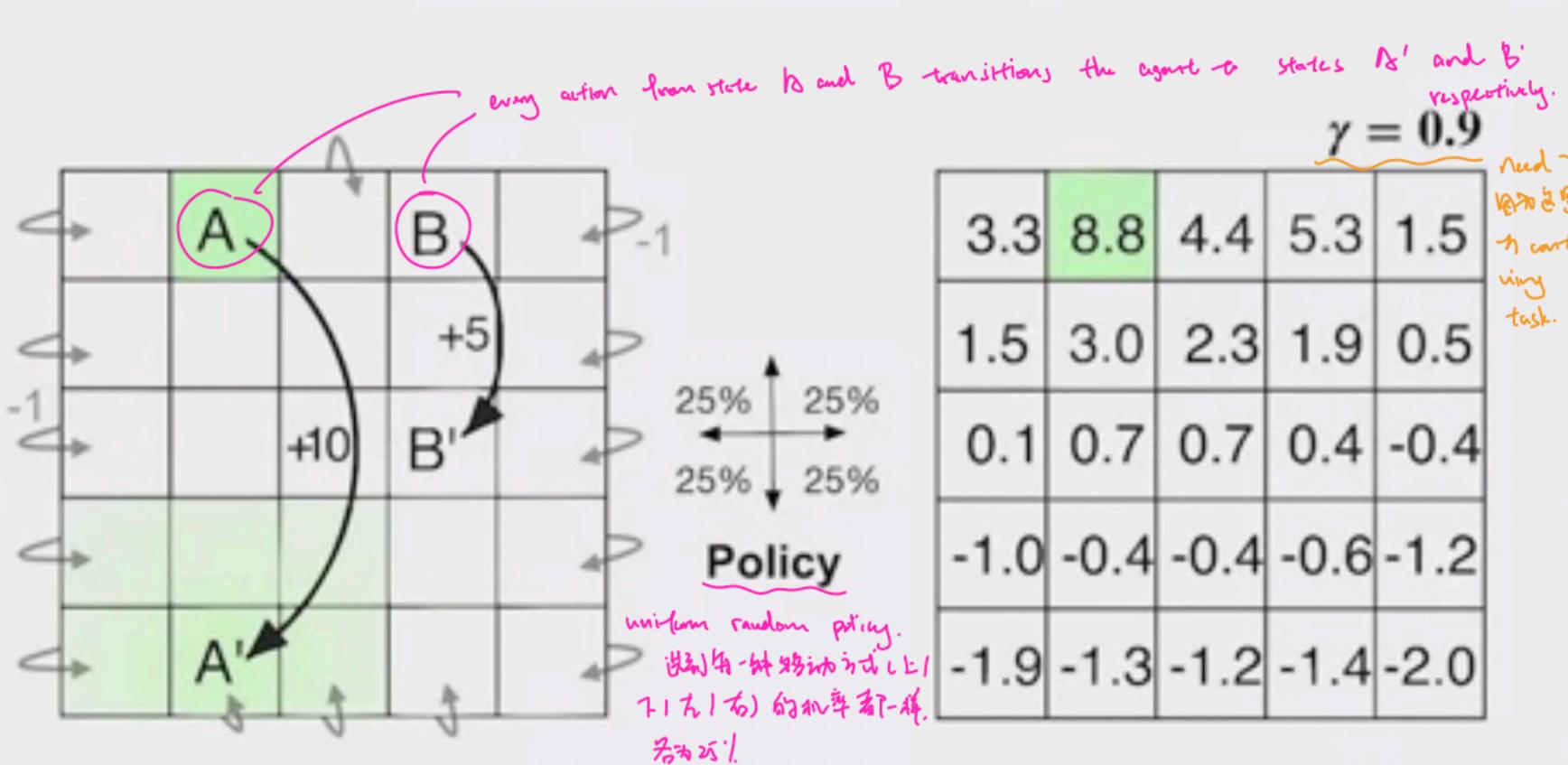
Value function example: Chess

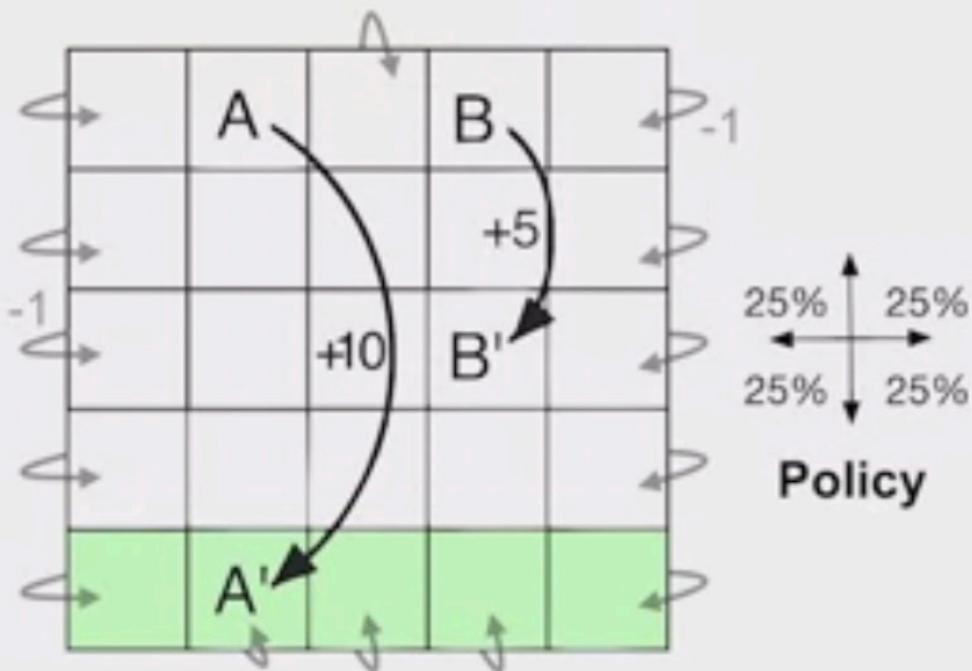


$$P(\text{win}) = V_{\pi}(s) = 0.34$$

$$V_{\pi}(s') = \underline{0.31}$$

状态价值变低了!! ∵ policy
更差的机率就更小了。



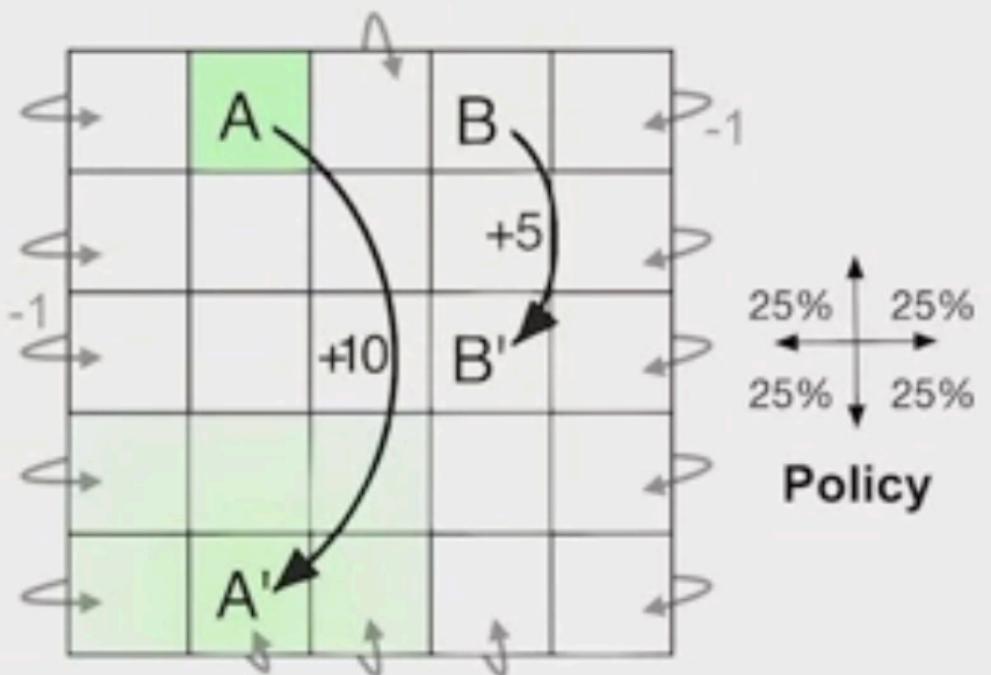


$$\gamma = 0.9$$

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Policy

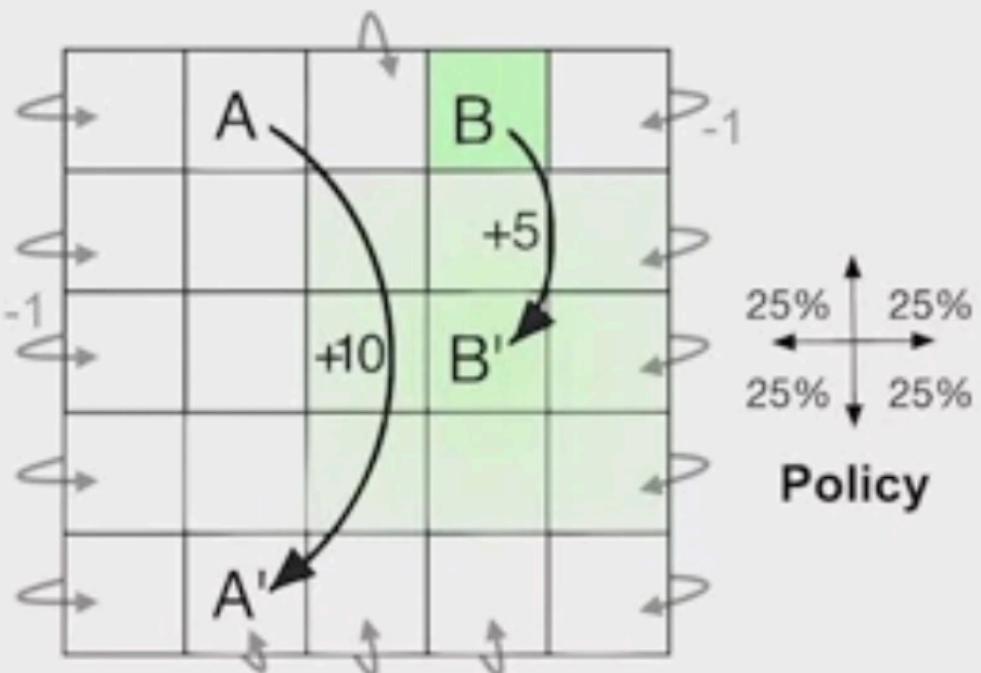
箭頭下的這些 state value 值很小，說明 agent 很難接觸到了。(接觸會好)



$\gamma = 0.9$

l State horizon is 10, 1000 many action in state B, limit the reward is B'.
lower wall.
Policy 送
action 送
隨機並傳
到-1是行動

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

Mark B 的 state value > Mark B 的期望
 TDS. 即对 every action in Mark B
 Inimitable agent \rightarrow 模倣中間, i. 亂數值
 ↗ Random policy to take action w/ $\gamma = 0.9$ 易達標.

Summary

- State-value functions represent the expected return from a given state under a specific policy
- Action-value functions represent the expected return from a given state after taking a specific action, later following a specific policy

RL是一个记忆化的搜索 (memorized search)。你做了一些操作，然后你记住了结果，下次你要做的时候，你查一下，而不是重新计算，这样可以节省很多时间。从某种意义上说，我们的RL在其根源上是：记忆性的上下文敏感搜索 (memorized context-sensitive search)。Pople stone，实际上在他的一篇关于这个的论文的末尾，谈到了插值器 (interpolator)，比如用多项式代替Lookup table来查找具有概括性的东西。例如，这就是神经网络的作用。所以我们没有--我们发现了。我们没有发明记忆，而是通过对它的新用途。我不知道人们是否以强化学习者的方式进行记忆性搜索。在这里，他有一个分布式方法的想法。寻金系统 (Gold seeking systems) 是由寻金组件 (gold seeking components) 组成的。他还提出了你所说的广义强化 (generalized reinforcement) 的想法，即这些单元中的一个可以被各种信号强化，而不仅仅是我们的二进制信号。就目前而言，有一个观点，我认为这就是强化学习。它只是专注于一个学习系统中实际上是要做试错的东西，并记住它，并有到专门的奖励信号。 (It is just focusing on a learning system that actually wants something that does trial and error and remembers it, and has to specialized reward signal.)

在日常生活中，我们在没有得到明确的、积极的或消极的反馈的情况下学到很多东西。例如，想象一下，你在骑自行车时，撞到了一块石头，使你失去了平衡。假设你恢复了，所以你没有受到任何伤害。你可能会学会在未来避开岩石，如果你真的撞到了岩石，也许会有更快的反应。即使这次没有发生什么坏事，我们怎么知道撞到石头是坏事呢？答案是，我们认识到失去平衡的状态是不好的，即使没有摔倒和伤害自己。也许我们在过去也有过类似的经历，当时事情的发展并不顺利。在强化和学习中，一个类似的想法使我们能够将当前状态的价值与未来状态的价值联系起来，而无需等待观察所有未来的回报。我们使用贝尔曼方程 (Bellman equation) 来正式确定一个状态的价值 (value of a state) 和其可能的继任者 (its possible successors) 之间的这种联系。

首先，我们来谈谈状态值函数的贝尔曼方程。状态价值函数的贝尔曼方程定义了一个状态的价值和他可能的继任状态的价值之间的关系。现在，我将说明如何从状态值函数和回报的定义中推导出这种关系。让我们首先回顾一下，状态价值函数被定义为从状态S开始的预期收益。回顾一下，回报被定义为未来回报的折现总和。我们之前看到，时间t的回报可以递归地写成即时回报加上时间t的折现回报加1 (immediate reward + the discounted return at time $t + 1$)。现在，让我们来扩展这个预期收益。首先，我们把预期收益扩展为代理人可能做出的行动选择的总和。其次，我们对可能的奖励和下一个状态进行扩展，条件是状态S和行动a。我们可以按这个顺序分解，因为行动选择只取决于当前状态，而下一个状态和奖励只取决于当前状态和行动。其结果是一个加权项的总和，由眼前的回报加上下一个状态S首要的预期未来回报组成。

我们所做的就是明确地把期望写成它所定义的那样，作为可能结果的总和，按其发生的概率加权。注意，资本 R_{t+1} 是一个随机变量，而小 r 代表每个可能的奖励结果。预期收益取决于无限远的未来状态和奖励。我们可以随心所欲地递归展开这个方程，但这只会使表达式更加复杂。相反，我们可以注意到，这个预期收益也是状态 S' 的价值函数的定义。唯一的区别是时间指数是 $t+1$ 而不是 t 。这不是一个问题，因为政策和 p 都不取决于时间。通过这种替换，我们得到了状态价值函数的贝尔曼方程。价值函数的神奇之处在于，我们可以用它们作为无限多可能的未来的平均值的替身。我们可以为行动价值函数推导出一个类似的方程。它将是一个递归方程，用于计算一个状态行动对在其可能的继任状态行动对方面的价值。在这种情况下，这个方程并不是从政策选择一个行动开始的。这是因为该行动已经作为状态行动对的一部分固定下来了。相反，我们直接跳到动态函数 p 来选择即时奖励和下一个状态 S' 。同样，我们有一个加权总和的条款，包括立即奖励和预期未来回报，给定一个特定的下一个状态小 S' 。然而，与状态价值函数的贝尔曼方程不同，我们不能在这里停止。我们想用递归方程来计算一个状态行动对在下一个状态行动对方面的价值。此刻，我们只有给定下一个状态的预期收益。为了改变这种情况，我们可以把下一状态的预期收益表达为代理人可能的行动选择的总和。特别是，我们可以将期望值改为以下一个状态和下一个行动为条件，然后对所有可能的行动进行求和。每项都由在状态 S' 中选择 A' 的 $P_{i'}$ 下的概率来加权。现在，这个预期收益与 S' 和 A' 的行动价值函数的定义相同。通过这种替换，我们得到了行动价值函数的贝尔曼方程。所以我们现在已经涵盖了如何推导出状态和行动价值函数的贝尔曼方程。

State-value Bellman equation

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

$$= \mathbb{E}_{\pi} [R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s'] \right]$$

↳ each possible reward outcome.

Recall that

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

奖励
回报

info:

State-value Bellman equation

$$v_\pi(s) \doteq \mathbb{E}_\pi [G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi [R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi \left[\text{state value of state } s' \right] \right]$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \sum_a \pi(a' \mid s') \sum_{s''} \sum_{r'} p(s'', r' \mid s', a') \left[r' + \gamma \mathbb{E}_\pi [G_{t+2} \mid S_{t+2} = s''] \right] \right]$$

Recall that

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

State-value Bellman equation

$$v_\pi(s) \doteq \mathbb{E}_\pi [G_t \mid S_t = s]$$

$$= \mathbb{E}_\pi [R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_\pi [G_{t+1} \mid S_{t+1} = s'] \right]$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \underbrace{v_\pi(s')}_{\text{pink arrow}} \right]$$

Recall that

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Action-value Bellman equation

$$q_{\pi}(s, a) \doteq \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s'] \right]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') \mathbb{E}_{\pi} [G_{t+1} \mid S_{t+1} = s', A_{t+1} = a'] \right]$$

$$= \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma \sum_{a'} \pi(a' \mid s') q_{\pi}(s', a') \right]$$

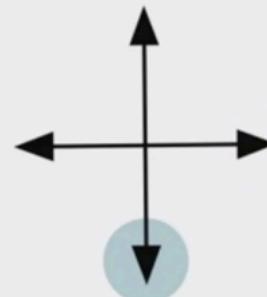
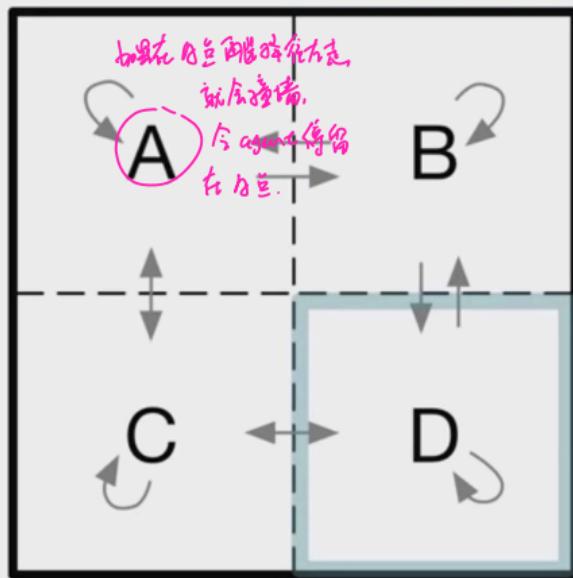
Summary

- We have derived the Bellman Equations for state-value and action-value functions
- The current time-step's state/action values can be written recursively in terms of future state/action values

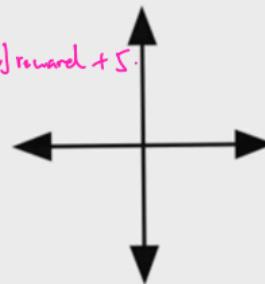
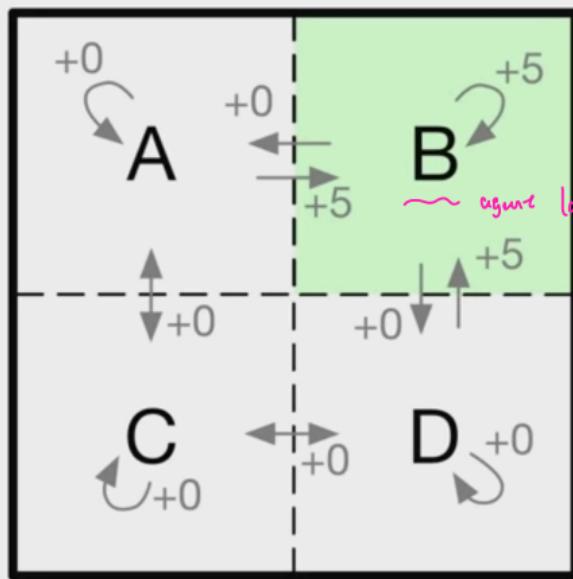
之前，我们学习了贝尔曼方程如何让我们用一个状态或状态动作对的可能继任者来表达其价值。但你可能想知道，这一切的意义何在？在本视频结束时，你将能够使用贝尔曼方程来计算价值函数。

为了说明贝尔曼方程的威力，请考虑这个小例子，它只由四个状态组成，在一个网格上标有A、B、C和D。行动空间包括向上、向下、向左和向右移动。将会移出网格的行动反而使代理人保持原位。例如，我们从C状态开始，向上移动会把我们带到A状态。如果我们试图向左移动，我们会撞到墙并停留在A状态。这包括从B状态开始，撞到墙而留在那里。让我们考虑统一的随机政策，它在每个方向上都有25%的时间移动。折扣系数gamma 0.7。在这个政策下，我们如何才能实际算出这些状态A、B、C和D中每一个的价值呢？回顾一下，价值函数被定义为政策派下的预期收益。这是对代理人可能选择的每一个行动序列所获得的回报的平均数，无限地，许多可能的特征。幸运的是，状态价值函数的贝尔曼方程提供了一个优雅的解决方案。使用贝尔曼方程，我们可以用四种可能的行动和由此产生的可能的后续状态的总和来写出状态A的价值表达。在这种情况下，我们可以进一步简化表达式，因为对于每个行动，只有一个可能的相关的下一个状态和奖励。这就使s'和r的总和减少到一个单一的值。请注意，这里的s'和r仍然取决于所选择的行动和当前的状态s。但为了保持符号的简短，我们没有明确说明这一点。如果我们从状态A向右走，我们就会落在状态B，并得到+5的奖励。这种情况在随机政策下有四分之一的时间发生。如果我们往下走，我们会落在状态C，并且没有立即得到奖励。同样，这种情况也有四分之一的时间发生。如果你向上或向左走，我们将再次回到状态A。每一个动作，向上和向左，同样有四分之一的时间发生。由于它们都降落在A状态，并且没有得到任何奖励，我们把它们合并为一个系数为1大于2的单项。最后，我们得到了这里显示的状态A的值的表达式。我们可以为其他每个状态B、C和D写下类似的方程。如果你愿意，你可以用手来解决这个问题，或者把它放到一个自动方程解决器中。

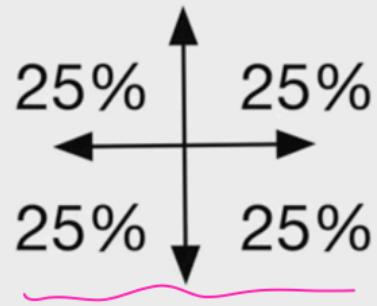
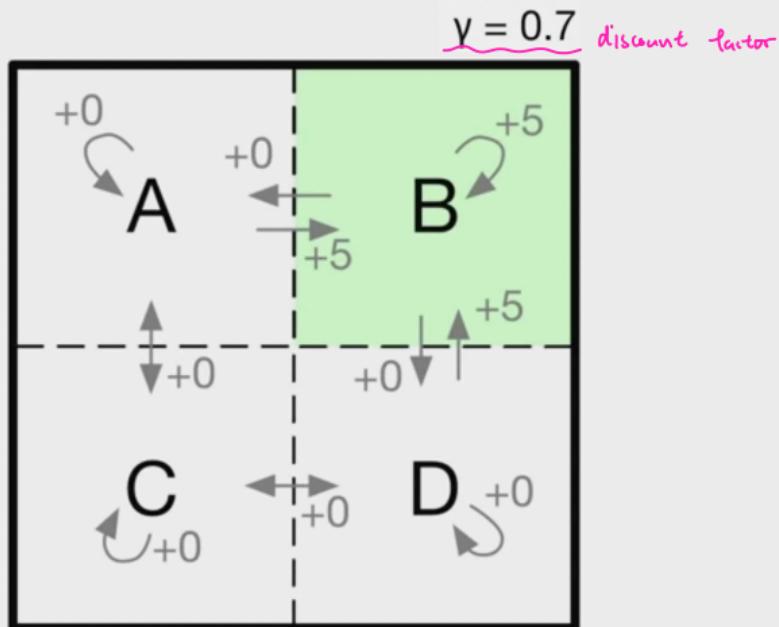
Example: Gridworld



Example: Gridworld

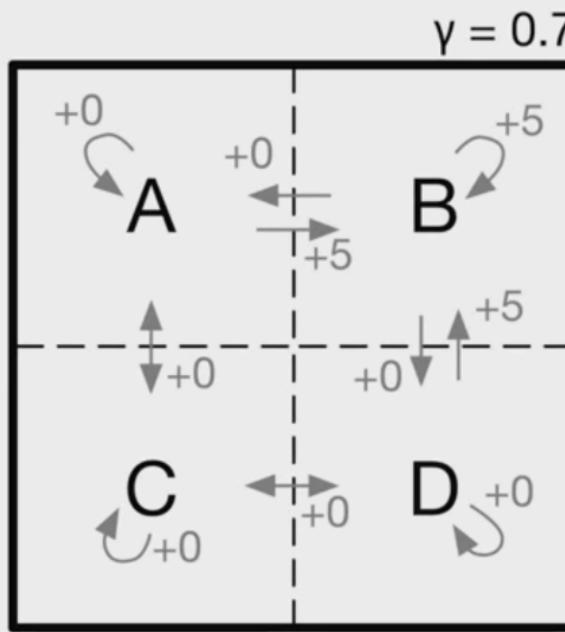


Example: Gridworld



uniform random policy. (均匀随机动作概率)
都为 25%.

Example: Gridworld



action
value
function

$$V_{\pi}(A) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = A]$$

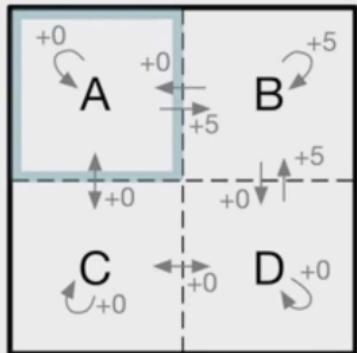
$$V_{\pi}(B) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = B]$$

$$V_{\pi}(C) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = C]$$

$$V_{\pi}(D) \doteq \mathbb{E}_{\pi}[G_t \mid S_t = D]$$



Example: Gridworld



$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

π = 3.14

从A走到B, 得到15

E. uniformis (L.)

Polym 7, right action

(向右走) 4. 251. 27 右的
被偷出現

$$\frac{1}{4} \underbrace{0.7V_\pi(C)}_{\text{Value Function}}$$

LL 1833c.

获得 + reward

be uniform random

Poling 7, right action

$$\frac{1}{2} \underbrace{0.7V_\pi(A)}_{\text{Reward}}$$

向上走和向左走的志向志向

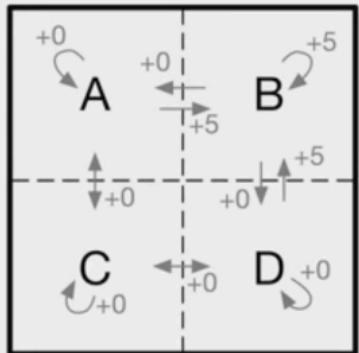
福壽，舊留在 State A.

∴ 蘇鶴獎助 10
元

• Fe uniform random policy π , 选择 action
• 有 π 为 1. 27 3 的概率出现,

Example: Gridworld

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



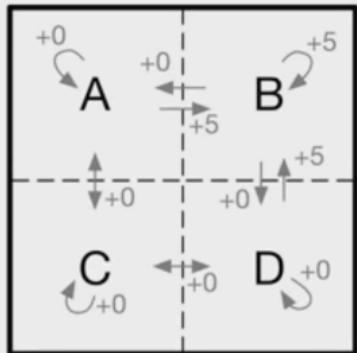
π 25% 25%
 ← →
 25% 25%

is system
of 4
equations
vs 4
variables.
(4元方程组)

$$\left. \begin{aligned} V_{\pi}(A) &= \frac{1}{4}(5 + 0.7V_{\pi}(B)) + \frac{1}{4}0.7V_{\pi}(C) + \frac{1}{2}0.7V_{\pi}(A) \\ V_{\pi}(B) &= \frac{1}{2}(5 + 0.7V_{\pi}(B)) + \frac{1}{4}0.7V_{\pi}(A) + \frac{1}{4}0.7V_{\pi}(D) \\ V_{\pi}(C) &= \frac{1}{4}0.7V_{\pi}(A) + \frac{1}{4}0.7V_{\pi}(D) + \frac{1}{2}0.7V_{\pi}(C) \\ V_{\pi}(D) &= \frac{1}{4}(5 + 0.7V_{\pi}(B)) + \frac{1}{4}0.7V_{\pi}(C) + \frac{1}{2}0.7V_{\pi}(D) \end{aligned} \right\}$$

Example: Gridworld

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



$$V_{\pi}(A) = 4.2$$

$$V_{\pi}(B) = 6.1$$

$$V_{\pi}(C) = 2.2$$

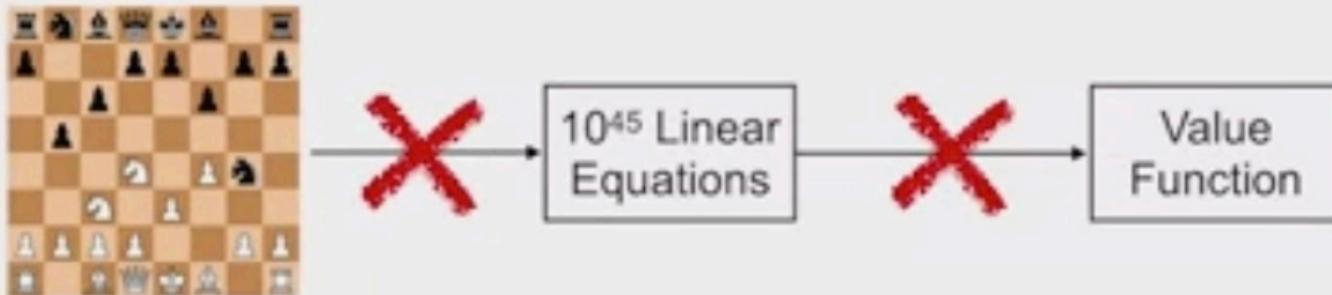
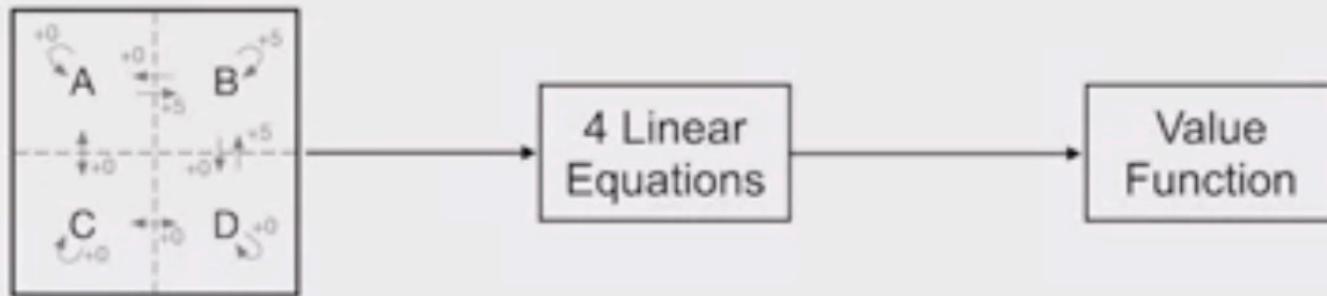
$$V_{\pi}(D) = 4.2$$



这里显示的是唯一的解。值得注意的是，贝尔曼方程将一个难以管理的对可能的未来的无限和减少到一个简单的线性代数问题。也许对于这个小问题，你可以想出其他方法来计算这些状态中的每一个值。然而贝尔曼方程为MDP提供了一个强大的一般关系。在这个案例中，我们用贝尔曼方程直接写下了个状态值的方程组，然后用一些系统来寻找值。这种方法对于中等规模的MDP来说可能是可行的。然而，在更复杂的问题中，这并不总是实用的。

例如，考虑一下国际象棋游戏。我们可能甚至无法列出所有可能的状态，大约有10到45种状态。构建和解决由此产生的贝尔曼方程系统将是一个完全不同的故事。是的，人类可以很好地学习下棋。然而，这个简单的游戏只代表了人类经验的一小部分，而人类可以学习做很多事情。我们的代理也应该能够学习很多东西。在接下来的课程中，我们将讨论基于贝尔曼方程的算法，这些算法可以扩展到大型问题。希望你能明白为什么贝尔曼方程是强化学习的基础。总而言之，如果没有贝尔曼方程，我们可能不得不考虑无限多的可能的未来。贝尔曼方程利用MDP公式的结构，将这个无限的总和减少为一个线性方程系统。然后，我们可以直接解决贝尔曼方程，以找到状态值。

We can only directly solve small MDPs



Summary

- You can use the **Bellman Equations** to solve for a value function by writing a **system of linear equations**
- We can only solve **small MDPs** directly, but **Bellman Equations** will factor into the solutions we see later for **large MDPs**

到此为止，我们通常把政策说成是被给予的东西。政策规定了一个代理人的行为方式。鉴于这种行为方式，我们的目标是找到价值函数。但强化学习的目标不仅仅是评估具体的政策。最终，我们希望找到一个政策，在长期内获得尽可能多的奖励。在这段视频中，我们将用最优策略的概念来精确这一概念。在本视频结束时，你将能够定义一个最优政策，理解政策如何在每个状态下至少与其他政策一样好，并为一个给定的MDP确定一个最优政策。

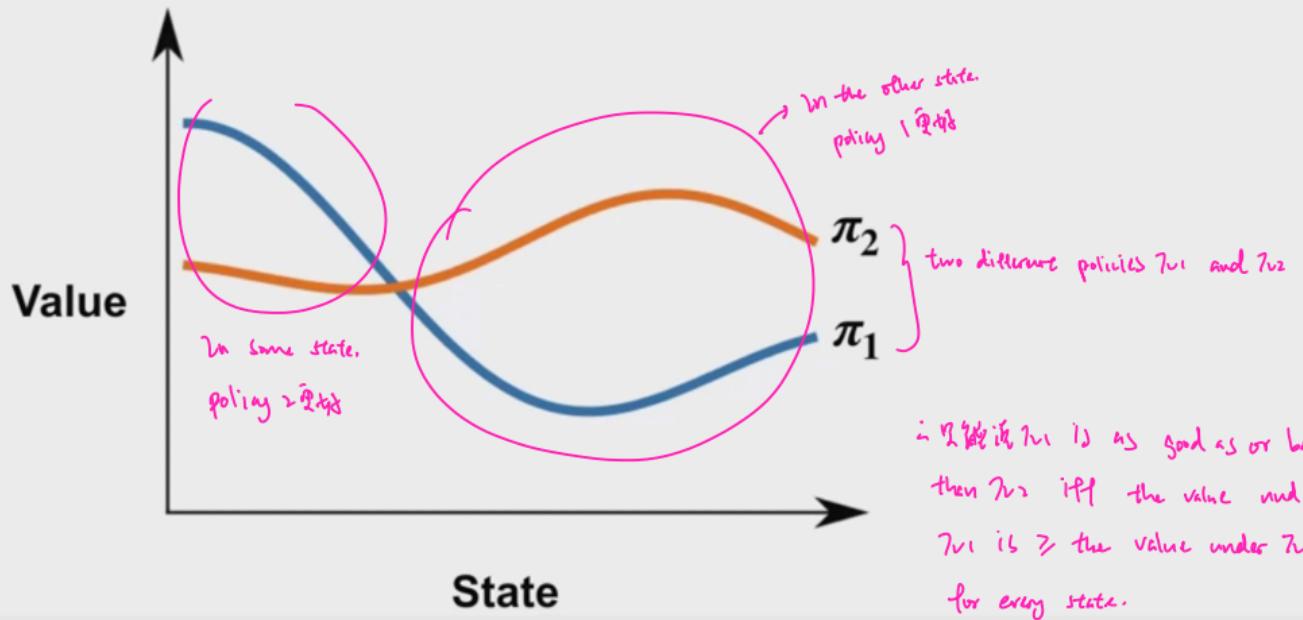
要定义一个最优政策，我们首先要理解一个政策比另一个政策好意味着什么。这里我们可以看到两个政策在不同状态下的价值。请注意，我们把它绘制成一条连续的线，只是为了说明问题。我们仍然在考虑离散的状态，而且不同状态下的价值可能不是非常平稳。这张图说明，在某些状态下， π_1 取得了更高的价值，而在其他状态下， π_2 取得了更高的价值。因此，说 π_1 比 π_2 好，或者说 π_2 比 π_1 好，都没有多大意义。我们将说政策 π_1 和政策 π_2 一样好或更好，当且仅当 π_1 下的值 $>$ 或 $=$ π_2 下的值时，每个状态下都是如此。我们用 $>$ 或 $=$ 符号来表示政策之间的这种关系。在这里显示的图表中，显示 π_1 的价值的线总是在 π_2 的线之上。所以在这种情况下， π_1 和 π_2 一样好或者更好。

一个最优的政策，是一个与所有其他政策一样好或更好的政策。也就是说，一个最优政策在每个状态下都会有最高的可能值。总是至少有一个最优政策，但可能不止一个。我们将使用符号 π^* 来表示任何最优政策。我们可能不清楚为什么一定存在最优政策。也就是说，政策至少与所有其他政策和每个状态一样好。难道我们就不能有一种情况，我们在一种状态下做得好，需要在另一种状态下做得不好？比方说，有这样一个政策 π_1 ，在一些州做得很好，而政策 π_2 ，在其他州也做得很好。我们可以把这些政策组合成第三个政策 π_3 ，它总是根据政策 π_1 和 π_2 中的哪一个在当前状态下具有最高价值来选择行动。在每个状态下， π_3 的值必然 $>$ 或 $=$ π_1 和 π_2 。所以我们永远不会有这样的情况，我们在一个状态下做得很好，却要牺牲另一个状态下的价值。正因为如此，总是存在着一些在每个状态下都是最好的政策。这当然只是一个非正式的论证，但事实上有一个严格的证明，表明一定总是存在至少一个最佳的确定性政策。

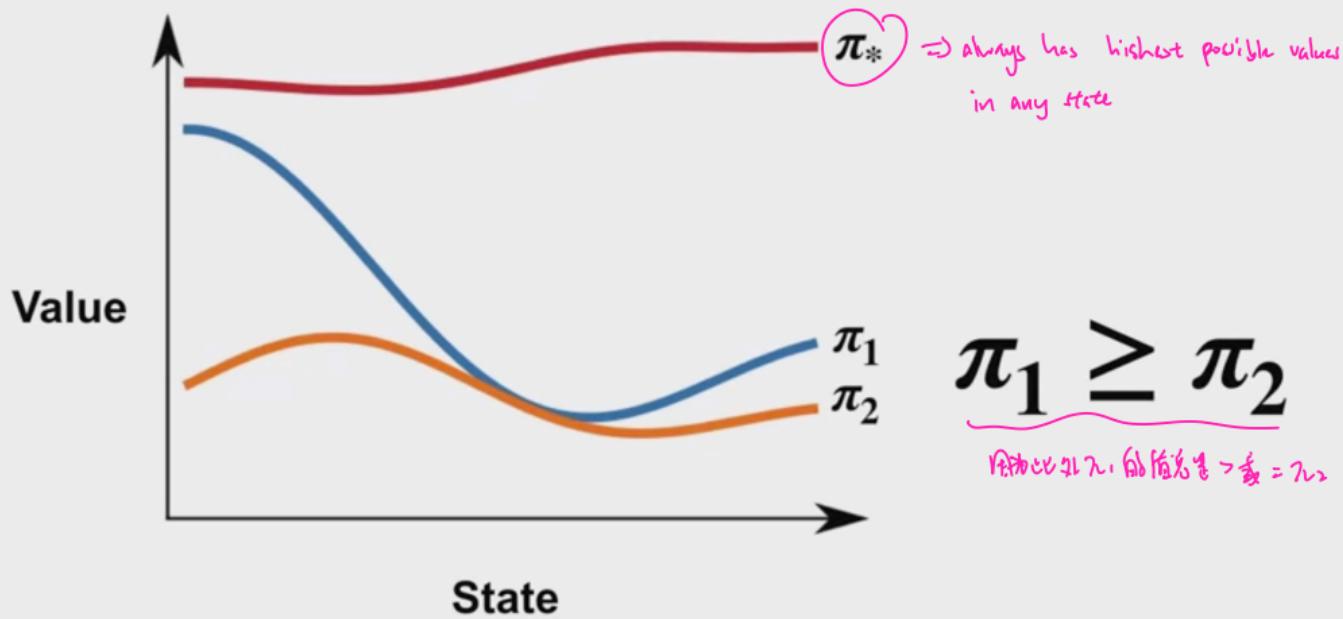
请注意，在某些状态下， π_3 的值严格地大于 π_1 或 π_2 。作为一个练习，试着解释一下，鉴于 π_3 只是根据 π_1 和 π_2 中的哪一个在每个状态下有更高的价值来选择行动，这是怎么做到的呢？让我们看一个具体的例子来建立一些关于最优策略的直觉。考虑一下这里显示的两个选择的MDP。唯一要做的决定是在标记为X的顶层状态。在Y状态下，只有行动A1是可用的，它把代理人带回到X状态。另一方面，行动A2和X把代理人带到Z状态。请注意，虽然A1提供了一个即时的奖励，但A2在单步延迟后提供了一个更大的奖励，即加2。在这个MDP中只有两个确定的政策，它们完全由代理人在状态X中的选择来定义。让我们把它们称为 π_1 和 π_2 。这两个政策中哪一个是最优的？最佳政策是X值最高的那个。答案取决于折扣系数Gamma。考虑到Gamma等于零。在这种情况下，价值的定义只使用即时奖励。在 π_1 下X状态的值是加一，而在 π_2 下的值是零，因为加到奖励是在延迟一步后发生的，当Gamma被设置为零时，这不会影响回报。所以在这种情况下， π_1 是最优政策。如果相反，Gamma等于0.9呢？在这种情况下，每个政策下的X值都是一个无限的总和。 π_1 立即得到1的奖励，然后是0，然后又是1，以此类推。每个奖励和状态X的值的表达都被Gamma的某个幂打了折扣。我们可以将其紧凑地表达为一个几何数列。

应用几何数列公式，我们可以得到 π_1 的数值，其估值约为5.3。我们可以把这个值写在 π_2 下面。同样，除了我们将在每一个奇数步骤中获得2的奖励，而不是每一个偶数步骤中获得1的奖励。我们可以再次将其写成一个几何数列，并得到一个封闭式的解决方案。在这种情况下，解决方案的估值约为9.5。由于9.5比5.3高，在这种情况下， π_2 是最佳的。在这两个选择的MDP中，找到最

优政策是相对简单的。只有两个确定的政策，我们只需要计算每个政策的价值函数。在一般情况下，这就不那么容易了。即使我们把自己限制在确定性的政策上，可能的政策数量也等于可能的行动数量与状态数量的幂。我们可以使用蛮力搜索 (brute force search)，计算每个政策的价值函数，以找到最佳政策。但不难看出，即使是中等规模的MDP，这也是难以实现的。幸运的是，有一个更好的方法来组织我们对政策空间的搜索。该解决方案将以另一组贝尔曼方程的形式出现，称为贝尔曼最优化方程 (Bellman's Optimality equations)，我们将在接下来的视频中介绍。最重要的事情是，最优策略被定义为在所有状态下具有最高价值的策略。至少有一个最优政策始终存在，但可能不止一个。可能政策的指数级数量，使得用蛮力搜索最优政策变得难以实现。

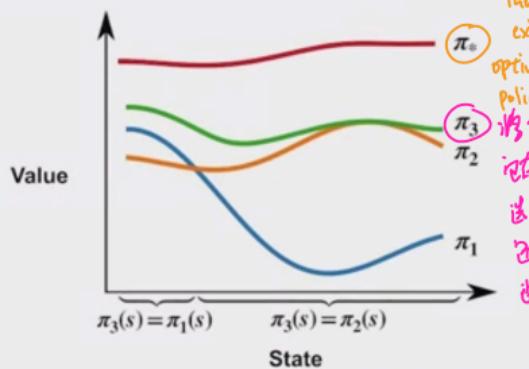


An **optimal policy** π_* is as good as or better than all the other policies



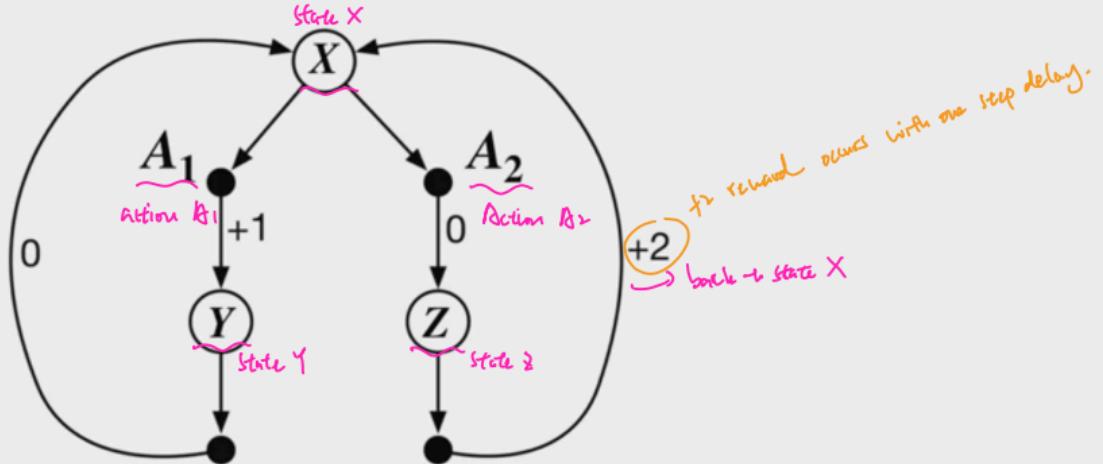


There is always an optimal policy



there must always
exists at least one
optimal deterministic
policy.

将 π_1 与 π_2 逐步迭代
直到 π_3 为最优
这儿获得的 value 里面
还是 π_1 更高，哪个遍历
遍历后。



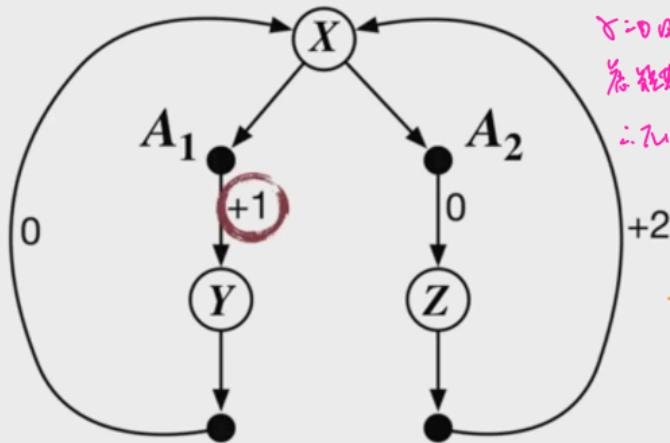
$$\pi_1(X) = A_1 \quad \pi_2(X) = A_2$$

two policies π_1 and π_2 under state X .

Following action A_1 and A_2 .

⇒ The optimal policy will be the one which value of X is highest.
it depends on the discount factor γ .

① Brute Force Search 以 π_1 和 π_2 是 optimal policy:



$$\pi_1(X) = A_1 \quad \pi_2(X) = A_2$$

$$\left. \begin{array}{l} \gamma = 0 \text{ 只是} \\ \text{若预期利益} \end{array} \right\} \begin{array}{l} \gamma = 0 \\ v_{\pi_1}(X) = 1 \quad \checkmark \\ v_{\pi_2}(X) = 0 \Rightarrow \text{因为从 right 走的第一步得到 reward 为 0.} \end{array}$$

$\therefore \pi_1$ 为 optimal policy. 因为从 A_1 走到 next step 有 1 的 reward +1.

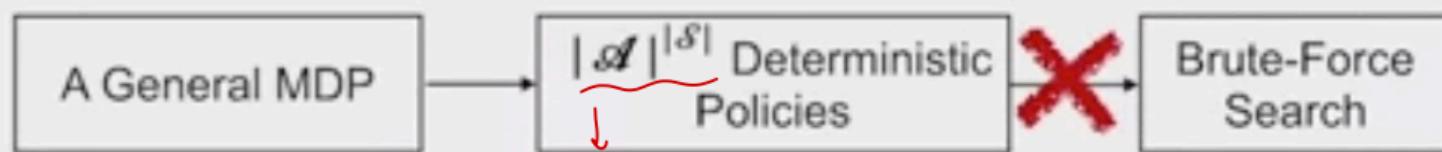
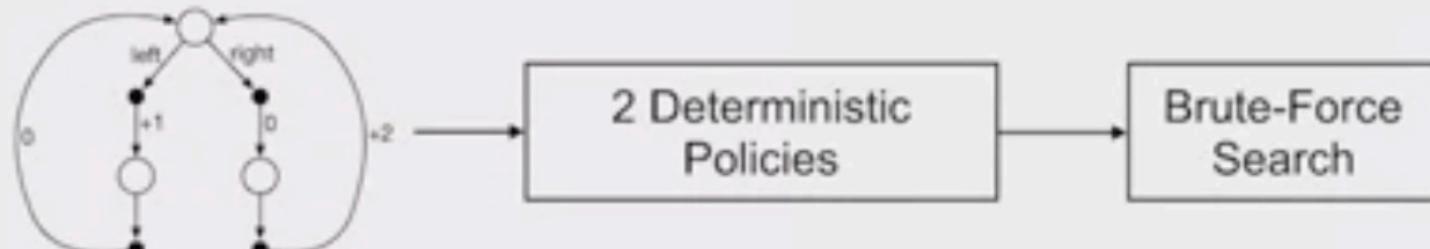
$$\left. \begin{array}{l} \gamma = 0.9 \\ v_{\pi_1}(X) = 1 + 0.9 * (0 + (0.9)^2 * 1 + \dots) = \sum_{k=0}^{\infty} (0.9)^{2k} = \frac{1}{1 - 0.9^2} \approx 5.3 \\ v_{\pi_2}(X) = 0 + 0.9 * 2 + (0.9)^2 * 0 + \dots = \sum_{k=0}^{\infty} (0.9)^{2k+1} * 2 = \frac{0.9}{1 - 0.9^2} * 2 \approx 9.5 \quad \checkmark \end{array} \right\} \begin{array}{l} \gamma = 0.9 \text{ 考虑从 A1 走} \\ \text{到 state } X \end{array}$$

$$\left. \begin{array}{l} \gamma = 0.5 : \quad v_{\pi_1}(X) = 1 + 0.5 * 0 = 1 \\ v_{\pi_2}(X) = 0 + 0.5 * 2 = 1.6 \end{array} \right\} \begin{array}{l} \gamma = 0.5 \text{ 考虑从 A2 走} \\ \text{到 state } X \end{array}$$

$$\left. \begin{array}{l} \gamma = 0.5 : \quad v_{\pi_1}(X) = 1 + 0.5 * 0 = 1 \\ v_{\pi_2}(X) = 0 + 0.5 * 2 = 1.6 \end{array} \right\} \begin{array}{l} \gamma = 0.5 \text{ 考虑从 A2 走} \\ \text{到 state } X \end{array}$$

π_2 的值更大.

We can only directly solve small MDPs



of deterministic policies = $|\# \text{ of Actions}|^{|\# \text{ of States}|}$

- i. 使用 Brute-Force Search 寫出 π^* optimal policy 3. 實用 Bellman optimality equations.

Summary

- An optimal policy is defined as the policy with the highest possible value function in all states
- At least one optimal policy always exists, but there may be more than one
- The exponential number of possible policies makes searching for the optimal policy by brute-force intractable

之前，我们了解到最优策略是如何实现强化学习的目标，即在长期内尽可能多地获得更多的收益。在这段视频中，我们将描述相关的最优价值函数的概念，并介绍一组相关的贝尔曼方程。在本视频结束时，你将能够推导出状态-价值函数的贝尔曼最优方程，推导出行动-价值函数的贝尔曼最优方程，并理解贝尔曼最优方程与之前介绍的贝尔曼方程的关系。回顾一下，对于两个政策 Pi_1 和 Pi_2 。当且仅当 Pi_1 下的值 $>$ 或 $=$ Pi_2 下所有状态的值时， Pi_1 被认为与 Pi_2 一样好或优于后者。一个最优的政策是与其他所有政策一样好或更好的政策。因此，最优政策的价值函数在每个状态下都有可能是最大的价值。

我们可以用数学的方式来表达这一点，即 S 的 v_{Pi} 等于所有政策的最大值。这对我们的状态空间中的每个状态都成立。在政策上取最大值可能不是很直观。所以让我们花点时间来分析一下它的含义。想象一下，我们要考虑每一种可能的政策，并计算它们对状态 S 的每一个值。一个最佳政策的值被定义为所有计算值中最大的一个。我们可以对每一个状态重复这个过程，最优政策的值总是最大的。所有的最优政策都有这个相同的最优状态值函数，我们用 v_{star} 表示。最佳政策也有相同的最佳行动价值函数，这也是每个状态行动对的最大可能。我们用 q_{star} 来表示这个共享的行动价值函数。回顾一下状态价值函数的贝尔曼方程。这个方程对任何政策包括最优政策的价值函数都是成立的。将最优策略 Pi_{star} 代入这个贝尔曼方程，我们就得到 v_{star} 的贝尔曼方程。到目前为止，我们还没有做任何特别的事情。这只是我们之前为最优策略的特定情况介绍的贝尔曼方程。然而，由于这是一个最优策略，我们可以用一种特殊的形式重写这个方程，它并不参考策略本身。记住，总是存在一个最优的确定性政策，一个在每个状态下都选择一个最优行动的政策。这样一个确定性的最优政策将为实现最高价值的行动分配概率 1，为所有其他行动分配概率 0。我们可以用另一种方式来表达，即用 a 的最大值来代替 Pi 星的总和。注意， Pi 星不再出现在方程中。我们已经推导出一种直接适用于 v 星本身的关系。我们把这种特殊的形式称为 V 星的贝尔曼最优性方程。我们可以对行动-价值函数的贝尔曼方程进行同样的替换。在这里，最优策略出现在内和中。再一次，我们用 a 的最大值来代替 Pi 星上的和，这就得到了 q 星的贝尔曼最优方程。在先前的讲座中，我们讨论了贝尔曼方程如何形成一个可以用标准方法解决的线性方程组。贝尔曼最优方程为我们提供了一个类似的最优值方程组。

一个自然的问题是，我们可以用类似的方法来解决这个系统，以找到最优状态值函数吗？不幸的是，答案是否定的。取行动上的最大值不是一个线性操作。因此，线性代数中用于解决线性系统的标准技术并不适用。在本课程中，我们不会以通常的方式形成和解决方程组。相反，我们将使用基于贝尔曼方程的其他技术来计算价值函数和策略。你可能想知道为什么我们不能简单地在普通的贝尔曼方程中使用 Pi 星来得到 v 星的线性方程组。答案很简单。我们不知道 Pi_{star} 。如果我们知道，那么我们就已经实现了强化学习的基本目标。如果我们能设法解决贝尔曼最优方程的 v 星，我们就能用这个结果相当容易地得到 Pi 星。我们将在下一个视频中看到如何做到这一点。因此，我们引入了最优价值函数，并推导出了关联的贝尔曼最优方程。贝尔曼最优方程将一个状态或状态-行动对的价值与任何最优策略下的可能继承者联系起来。下次见，我们将学习如何从最优价值函数中找到最优策略。

Optimal Value Functions

Recall that

$$\pi_1 \geq \pi_2 \text{ if and only if } v_{\pi_1}(s) \geq v_{\pi_2}(s) \text{ for all } s \in \mathcal{S}$$

$$V_* \quad \underbrace{v_{\pi_*}(s)}_{\text{why optimal policies have the same optimal state-value function } V^*} \doteq \mathbb{E}_{\pi_*}[G_t \mid S_t = s] = \max_{\pi} v_{\pi}(s) \text{ for all } s \in \mathcal{S}$$

why optimal policies have the same optimal state-value function V^*

$$Q_* \quad \underbrace{q_{\pi_*}(s, a)}_{\text{every optimal policy has the same optimal action-value function } Q^*(\text{maximum possible for every state-action pair})} = \max_{\pi} q_{\pi}(s, a) \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}$$

every optimal policy has the same optimal action-value function Q^* (maximum possible for every state-action pair)

Recall that

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

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

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



Bellman Optimality Equation for v_*

包括最佳策略在内的任何策略的价值函数。

Recall that

$$q_{\pi}(s, a) = \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right]$$

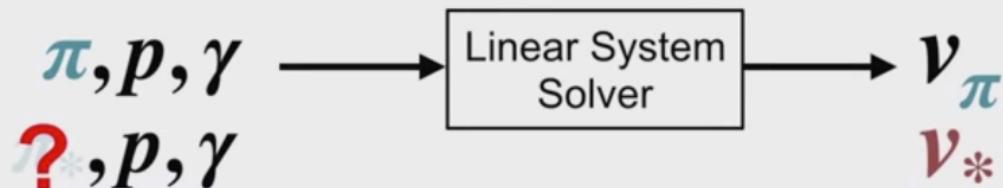
$$q_{*}(s, a) = \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \sum_{a'} \pi_{*}(a' | s') q_{*}(s', a') \right]$$

$$q_{*}(s, a) = \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \max_{a'} q_{*}(s', a') \right]$$



Bellman Optimality Equation for q_{*}

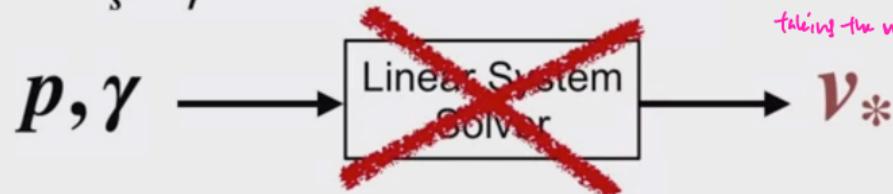
$$v_{\pi}(s) = \sum_a \pi(a | s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_{\pi}(s')]$$



Not linear

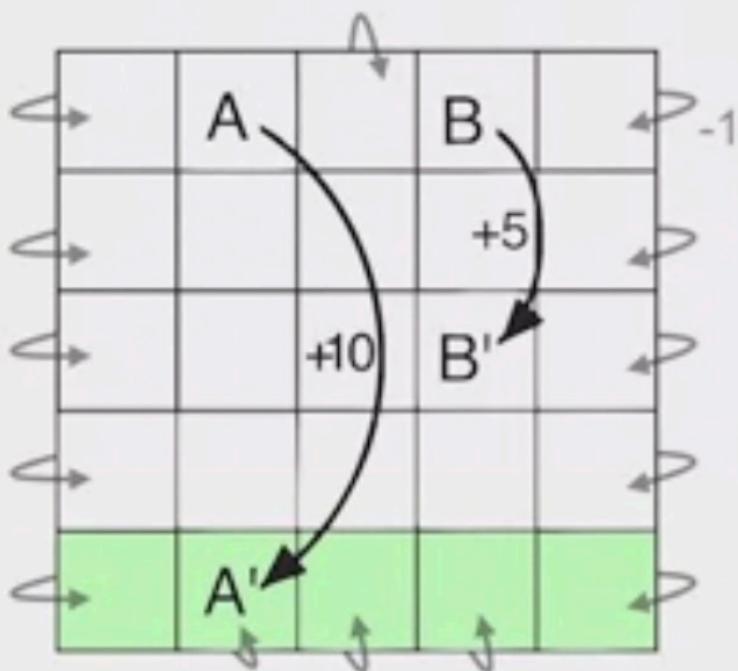
$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

2) We can't solve this system in a similar way to find the optimal state-value function, v_* !!
 taking the maximum over actions is not linear!!



Summary

- We introduced **optimal value functions** and derived the associated **Bellman optimality equations**
- The **Bellman optimality equations** relate the value of a state, or state-action pair, to its possible successors under **any optimal policy**

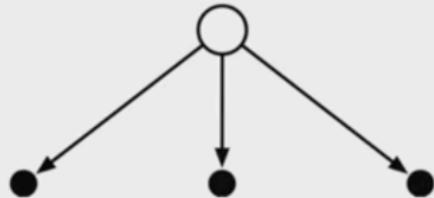


$$\gamma = 0.9$$

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_* 为什么 uniform random policy 不同。
 optimal policy 的选择永远不会碰到墙壁. 因此 state A 的 optimal value 这样 (10)

Determining an Optimal Policy



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

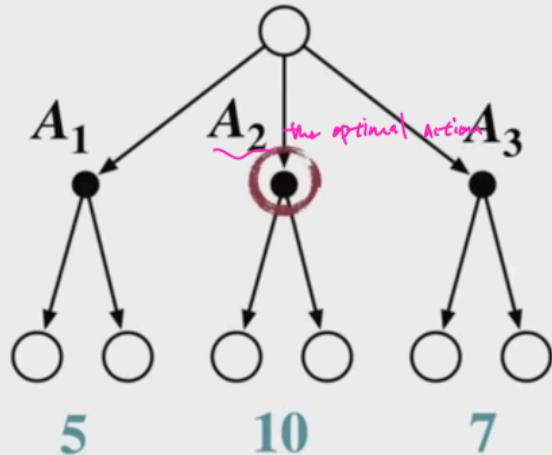
$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

The argmax ,
is the particular
action which achieves
this maximum.

\nearrow
 v^* = maximum of this term over all actions

v^* is equal to the maximum of
the boxed term over all actions.

Determining an Optimal Policy



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

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

Of these three actions, A2 maximizes the boxed term with a value of 10.

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

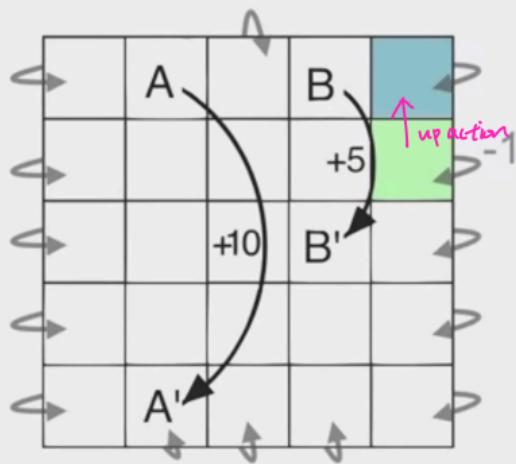
$$\gamma = 0.9$$

0 + 0.9 * 17.5 = 14.0

$v_*(s')$ at state s' is state value.

next state.

Sum of reward and discounted next state value.



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

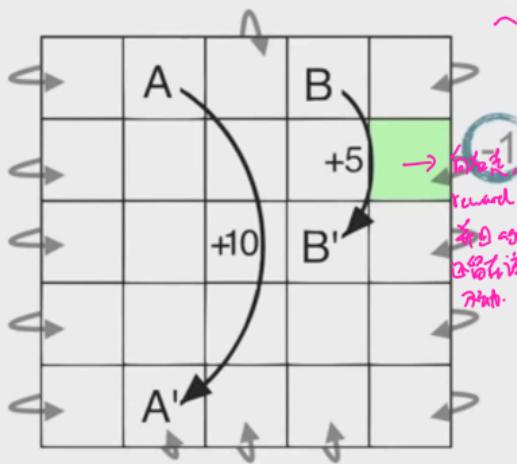
The up action leads here, giving no reward and a next state value of 17.5.

π_*
the optimal policy

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

$$\gamma = 0.9$$

$$-1 + 0.9 * 16.0 = 13.4$$



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

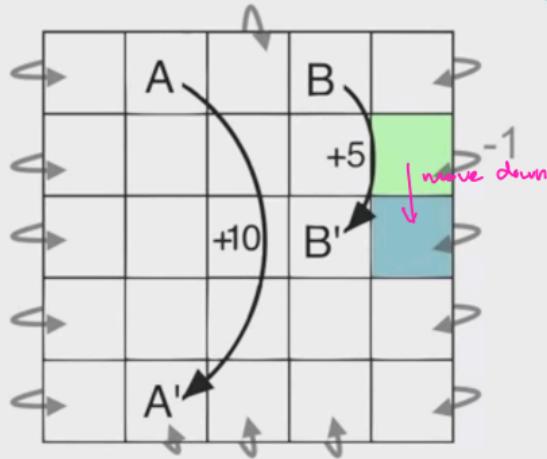
The right action hits the wall, giving -1 reward and

π_*

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

$$\gamma = 0.9$$

$$0 + 0.9 * 14.4 = 13.0$$



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

The down action leads here, giving no reward, but a next state value of 14.4.

$$\pi_*$$

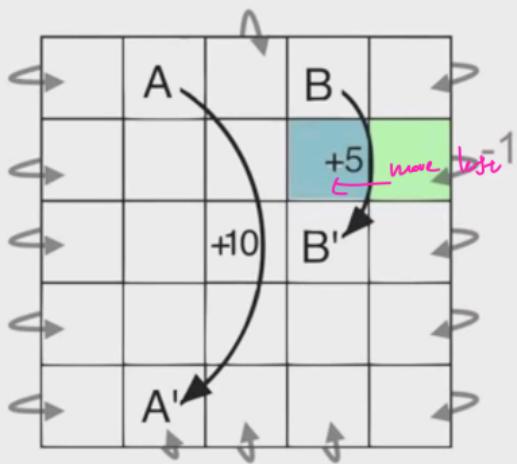
$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

→ to be 7th to 4th action of, 16 最後
 : optimal policy is move left. —
 $\gamma = 0.9$

$$0 + 0.9 * 17.8 = 16.0$$

10.0

16.0 ~~most~~ have value discovery by 8



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

A 5x5 grid of empty cells, intended for drawing or writing practice.

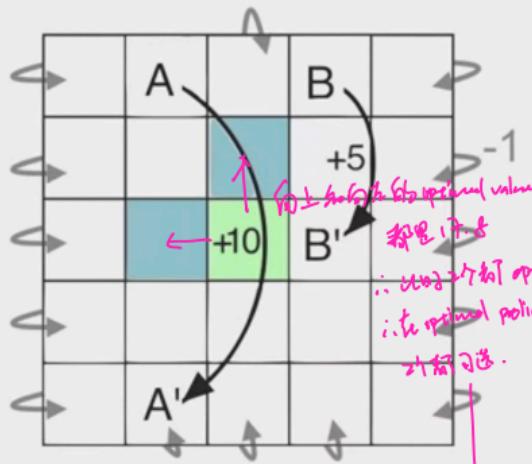
Again, giving no reward, but a next state value of 17.8.

π_*

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

$$\gamma = 0.9$$

$$0 + 0.9 * 19.8 = 17.8$$



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

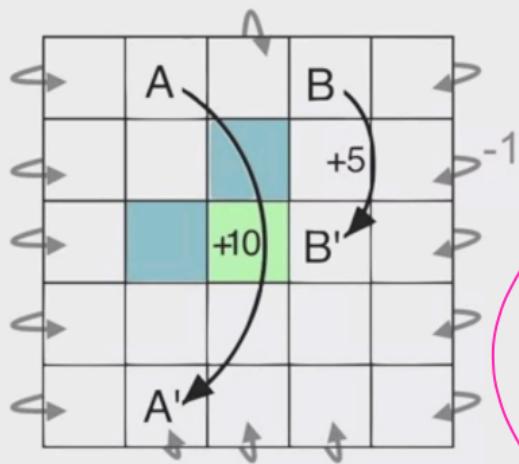
each give the same optimal value of 0.9 times 19.8, which equals 17.8.

 π_*

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

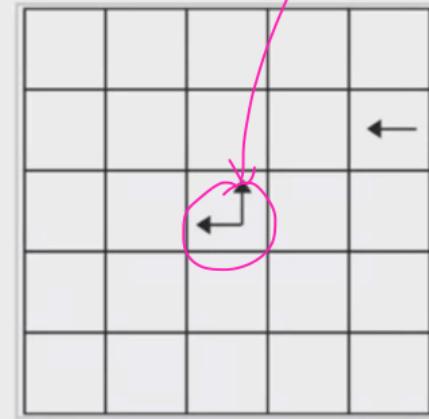
$$\gamma = 0.9$$

$$0 + 0.9 * 19.8 = 17.8$$



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

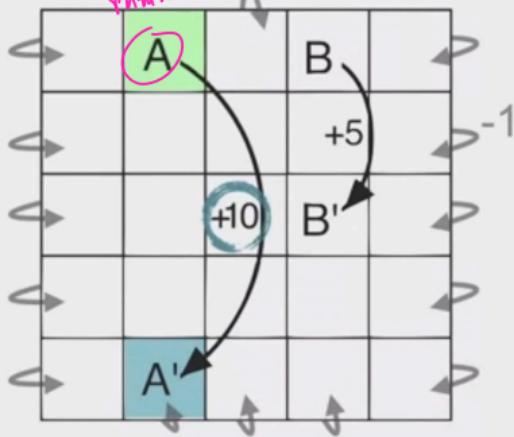
(optional) ν_* value of ν to start with



π_*

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

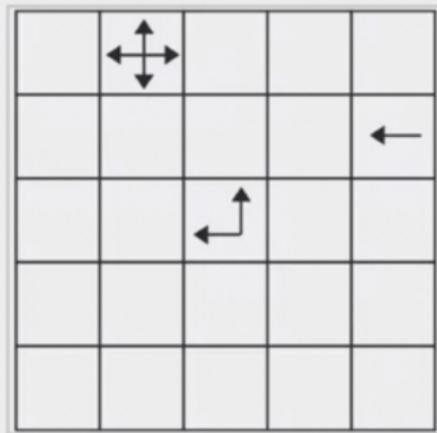
With $\gamma = 0.9$ the action s' is optimal. As $\gamma = 0.9 < 1$, the reward r is discounted. If state s is the only action is optimal. $10 + 0.9 * 16.0 = 24.4$ transitions are equivalent.



$$10 + 0.9 * 16.0 = 24.4$$

my action is optimal. $\gamma = 0.9$
transitions are equivalent.

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

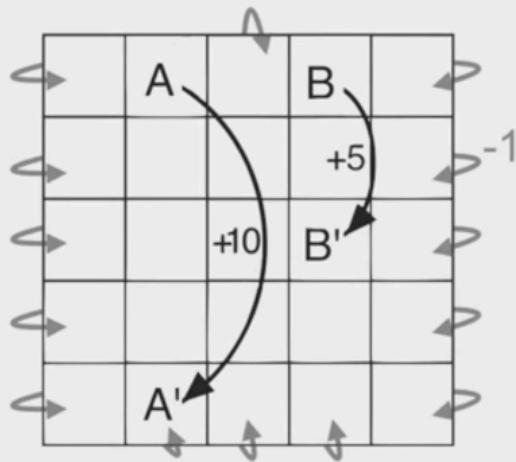


we transition to A prime with a reward of +10.

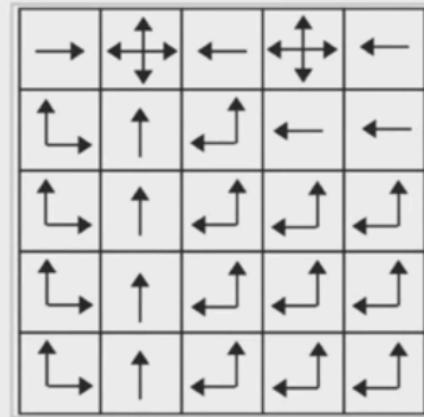
π_*

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$

$$\gamma = 0.9$$



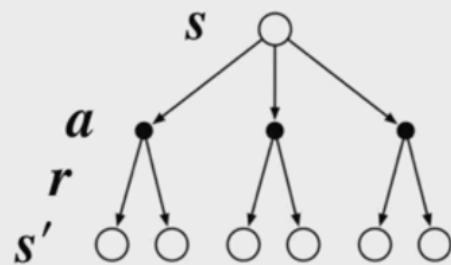
22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7



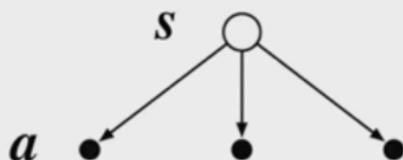
We see that the optimal policy essentially heads toward state A to obtain +10

$$\pi_*$$

Determining an Optimal Policy



$$\pi_*(s) = \operatorname{argmax}_a \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma v_*(s')]$$



$$\pi_*(s) = \operatorname{argmax}_a q_*(s, a)$$

Summary

- Once we have the **optimal state value function**, it's relatively easy to work out the **optimal policy**.
- If we have the **optimal action-value function**, working out the **optimal policy** is even easier.

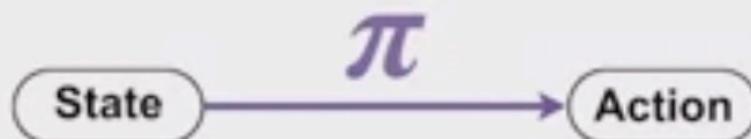
本周，我们了解了所有关于策略、价值函数以及它们之间的关系。我们还学习了贝尔曼方程，它帮助我们推理政策的价值。在这段视频中，我们将对我们所涉及的一切进行快速回顾。

策略告诉代理人如何行事。确定性的策略将每个状态映射到一个行动。每次访问一个状态，确定性策略都会选择 S 的相关行动 P_i 。随机性策略将每个状态映射为所有可能行动的分布。每次状态被访问时，随机策略从相关的分布中随机抽取一个行动，概率为给定 S 的 P_i 。它不能依赖于像时间或以前的状态。这最好被认为是对状态的限制，而不是对代理人的限制。状态应该为代理人提供它所需的所有信息以做出一个好的决定。这是强化学习中许多技术的一个重要假设。

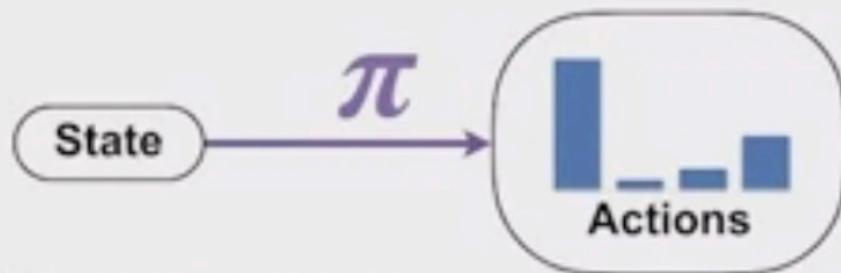
价值函数（Value functions）就像魔术一样。价值函数捕捉了特定政策下的未来总回报。我们讨论了两种价值函数：状态价值函数（state value functions）和行动价值函数（action value functions）。

Policies tell an agent how to behave in their environment

Deterministic policies



Stochastic policies



A policy depends only on the current state

The state should provide the agent all the information it needs to make a decision.

it can't depends on time \rightarrow it's previous thing.

gives the expected return from the current state under a policy.

状态值函数 (state value functions) 给出了一个政策下的当前状态的预期回报。**行动价值函数 (action value functions)** 给出了如果代理人首先选择行动A并在此后遵循 π_i 的情况下来自状态S的预期回报。价值函数通过将许多可能的未来回报汇总成一个数字来简化事情。**贝尔曼方程 (Bellman equations)** 定义了一个状态或状态-行动对的价值与它的后续状态之间的关系。**状态价值函数的贝尔曼方程**给出了当前状态的价值，即所有后续状态的价值之和，以及即时奖励。**行动价值函数的贝尔曼方程**给出了一个特定状态行动对 (state-action pair) 的价值，即所有可能的下一个状态行动对的价值和奖励的总和。贝尔曼方程可以直接求解以找到价值函数。这些贝尔曼方程帮助我们评估政策，但它们还没有实现我们的最终目标，即找到一个能获得尽可能多奖励的政策。为了使这个目标更加精确，我们定义了最优政策、最优价值函数和相关的贝尔曼最优方程。一个最优策略是在每个状态下都能达到最高值的策略。总是至少有一个最优政策，但可能不止一个。**最佳状态的价值函数**等于每个状态下可能的最高值。**每个最优政策都有相同的最优状态值函数**。对于**最优行动价值函数**和**最优政策**来说也是如此。像所有的价值函数一样，最优价值函数有贝尔曼方程。这些贝尔曼方程并不参考具体的政策。这相当于用所有行动的最大值来代替贝尔曼方程中的政策。**最佳政策必须总是选择最佳的可用行动**。我们可以从最优状态的价值函数中提取最优策略。但要做到这一点，我们还需要MDP的单步动态。如果我们有了最优行动价值函数，我们就能以更少的工作量得到最优策略。我们只需在每个状态下选择价值最高的行动。下周，我们将看到如何使用这些贝尔曼方程来计算最优策略。

V^ , equal to the highest possible value in every state.*

Value functions estimate future return under a specific policy

$$v_{\pi}(s) \doteq \mathbb{E}_{\pi} [G_t \mid S_t = s]$$

$$q_{\pi}(s, a) \doteq \mathbb{E}_{\pi} [G_t \mid S_t = s, A_t = a]$$

Value 函数通过将

help us evaluate policies.

Bellman equations define a relationship between the value of a state, or state-action pair, and its possible successors

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

$$q_{\pi}(s, a) = \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \sum_{a'} \pi(a' | s') q_{\pi}(s', a') \right]$$

求解以找到值函数。

The goal: to find a policy that obtains as much reward as possible

最优策略

Optimal policies

The optimal value function

Bellman optimality equations

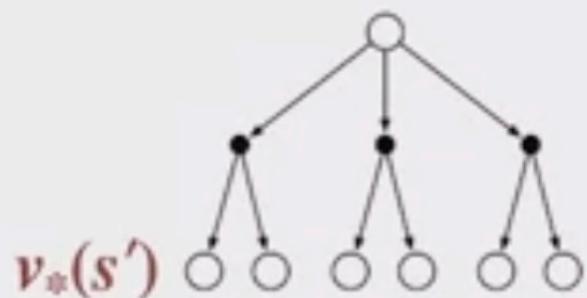
An **optimal policy** achieves the highest value possible in every state

$$\mathcal{V}_* \quad v_{\pi_*}(s) = \max_{\pi} v_{\pi}(s) \text{ for all } s \in \mathcal{S}$$

$$q_* \quad q_{\pi_*}(s, a) = \max_{\pi} q_{\pi}(s, a) \text{ for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}$$

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

$$q_*(s, a) = \sum_{s'} \sum_r p(s', r | s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]$$



I'm taking a Coursera course on Reinforcement learning. There was a question there that wasn't addressed in the learning material: Does adding a constant to all rewards change the set of optimal policies in episodic tasks?

The answer is Yes - Adding a constant to the reward signal can make longer episodes more or less advantageous (depending on whether the constant is positive or negative). 

Generally we can write for R_c the total reward with added constant c of a policy as

$$R_c = \sum_{i=0}^K (r_i + c)\gamma^i = \sum_{i=0}^K r_i \gamma^i + \sum_{i=0}^K c \gamma^i$$

So if we have two policies with the same total reward (without added constant)

$$\sum_{i=0}^{K_1} r_i^1 \gamma^i = \sum_{i=0}^{K_2} r_i^2 \gamma^i$$

but with different lengths $K_1 \neq K_2$ the total reward with added constant will be different, because the second term in R_c ($\sum_{i=0}^K c \gamma^i$) will be different.

As an example: Consider two optimal policies, both generating the same cumulative reward of 10, but the first policy visits 4 states, before it reaches a terminal state, while the second visits only two states. The rewards can be written as:

$$10 + 0 + 0 + 0 = 10$$

and

$$0 + 10 = 10$$

But when we add 100 to every reward:

$$110 + 100 + 100 + 100 = 410$$

and

$$100 + 110 = 210$$

Thus, now the first one is better.

In the continuous case, the episodes always have length $K = \infty$. Therefore, they always have the same length, and adding a constant doesn't change anything, because the second term in R_c stays the same.

[Practice] Value Functions and Bellman Equations

最新提交作业的评分 100%

1. A **policy** is a function which maps ___ to ___.

1/1分

- Actions to probability distributions over values.
- States to values.
- States to actions.
- States to probability distributions over actions.
- Actions to probabilities.

✓ 正确

Correct!

2. The term "**backup**" most closely resembles the term ___ in meaning.

1/1分

- Value
- Update
- Diagram

✓ 正确

Correct!

3. At least one deterministic optimal policy exists in every Markov decision process.

1/1分

False

True

正确

Correct! Let's say there is a policy π_1 which does well in some states, while policy π_2 does well in others. We could combine these policies into a third policy π_3 , which always chooses actions according to whichever of policy π_1 and π_2 has the highest value in the current state. π_3 will necessarily have a value greater than or equal to both π_1 and π_2 in every state! So we will never have a situation where doing well in one state requires sacrificing value in another. Because of this, there always exists some policy which is best in every state. This is of course only an informal argument, but there is in fact a rigorous proof showing that there must always exist at least one optimal deterministic policy.

4. The optimal state-value function:

1/1分

Is not guaranteed to be unique, even in finite Markov decision processes.

Is unique in every finite Markov decision process.

正确

Correct! The Bellman optimality equation is actually a system of equations, one for each state, so if there are N states, then there are N equations in N unknowns. If the dynamics of the environment are known, then in principle one can solve this system of equations for the optimal value function using any one of a variety of methods for solving systems of nonlinear equations. All optimal policies share the same optimal state-value function.

5. Does adding a constant to all rewards change the set of optimal policies in episodic tasks?

1/1分

- Yes, adding a constant to all rewards changes the set of optimal policies.
- No, as long as the relative differences between rewards remain the same, the set of optimal policies is the same.

正确

Correct! Adding a constant to the reward signal can make longer episodes more or less advantageous (depending on whether the constant is positive or negative).

6. Does adding a constant to all rewards change the set of optimal policies in continuing tasks?

1/1分

- Yes, adding a constant to all rewards changes the set of optimal policies.
- No, as long as the relative differences between rewards remain the same, the set of optimal policies is the same.

正确

Correct! Since the task is continuing, the agent will accumulate the same amount of extra reward independent of its behavior.

7. Select the equation that correctly relates v_* to q_* . Assume π is the uniform random policy.

1/1分

$v_*(s) = \max_a q_*(s, a)$

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

$v_*(s) = \sum_{a,r,s'} \pi(a|s) p(s', r|s, a) q_*(s')$

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

正确

Correct!

8. Select the equation that correctly relates q_* to v_* using four-argument function p .

1/1分

$q_*(s, a) = \sum_{s',r} p(s', r|a, s) [r + v_*(s')]$

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

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

正确

Correct!

9. Write a policy π_* in terms of q_* .

1/1 分

- $\pi_*(a|s) = q_*(s, a)$
- $\pi_*(a|s) = \max_{a'} q_*(s, a')$
- $\pi_*(a|s) = 1 \text{ if } a = \operatorname{argmax}_{a'} q_*(s, a'), \text{ else } 0$

正确

Correct!

10. Give an equation for some π_* in terms of v_* and the four-argument p .

1/1 分

- $\pi_*(a|s) = 1 \text{ if } v_*(s) = \max_{a'} \sum_{s',r} p(s', r|s, a')[r + \gamma v_*(s')], \text{ else } 0$
- $\pi_*(a|s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')]$
- $\pi_*(a|s) = \max_{a'} \sum_{s',r} p(s', r|s, a')[r + \gamma v_*(s')]$
- $\pi_*(a|s) = 1 \text{ if } v_*(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')], \text{ else } 0$

正确

Correct!

Value Functions and Bellman Equations

最新提交作业的评分 100%

1. A function which maps ___ to ___ is a value function. [Select all that apply]

1/1分

- Values to actions.
- State-action pairs to expected returns.

正确

Correct! A function that takes a state-action pair and outputs an expected return is a value function.

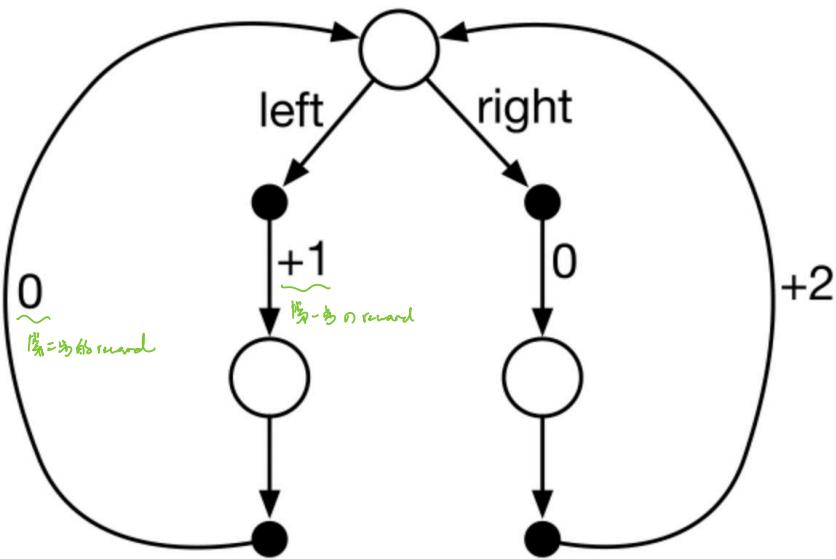
- States to expected returns.

正确

Correct! A function that takes a state and outputs an expected return is a value function.

- Values to states.

2. Consider the continuing Markov decision process shown below. The only decision to be made is in the top state, where two actions are available, left and right. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right} . Indicate the optimal policies if $\gamma = 0$? If $\gamma = 0.9$? If $\gamma = 0.5$? [Select all that apply]



- For $\gamma = 0.5$, π_{left} $V_{\pi_{\text{left}}} = 1 + 0.5 \times 0 = 1$ $V_{\pi_{\text{right}}} = 0 + 0.5 \times 2 = 1$ $V_{\pi_{\text{left}}} = V_{\pi_{\text{right}}}, \text{ i.e. } \pi_{\text{left}} \text{ is optimal.}$

正确

Correct! Since both policies return to the start state every two time steps, to determine the optimal policy, it suffices to consider the reward accumulated over the first two time steps. For the policy left, this is equal to 1; for the policy right, this is equal to 1.

- For $\gamma = 0$, π_{left} $V_{\pi_{\text{left}}} = 1 + 0 \times 1 = 1$ $V_{\pi_{\text{right}}} = 0 + 0 \times 2 = 0$ $V_{\pi_{\text{left}}} > V_{\pi_{\text{right}}}, \text{ i.e. } \pi_{\text{left}} \text{ is optimal.}$

正确

Correct! Since both policies return to the top state every two time steps, to determine the optimal policy, it suffices to consider the reward accumulated over the first two time steps. For the policy left, this is equal to 1; for the policy right, this is equal to 0.

For $\gamma = 0.9, \pi_{\text{left}}$

For $\gamma = 0.9, \pi_{\text{right}}$

$$V_{\pi_{\text{left}}} = 1 + 0.9x_0 = 1$$

$$V_{\pi_{\text{right}}} = 0 + 0.9x_2 = 1.8$$

$V_{\pi_{\text{right}}} > V_{\pi_{\text{left}}}$, $\therefore \pi_{\text{right}}$ is optimal.

正确

Correct! Since both policies return to the top state every two time steps, to determine the optimal policy, it suffices to consider the reward accumulated over the first two time steps. For the policy left, this is equal to 1; for the policy right, this is equal to 1.8.

For $\gamma = 0.5, \pi_{\text{right}}$

$$V_{\pi_{\text{left}}} = 1 + 0.5x_0 = 1$$

$$V_{\pi_{\text{right}}} = 0 + 0.5x_2 = 1$$

$V_{\pi_{\text{left}}} = V_{\pi_{\text{right}}}$, $\therefore \pi_{\text{left}}$ & π_{right} are optimal.

正确

Correct! Since both policies return to the start state every two time steps, to determine the optimal policy, it suffices to consider the reward accumulated over the first two time steps. For the policy left, this is equal to 1; for the policy right, this is equal to 1.

For $\gamma = 0, \pi_{\text{right}}$

- A stochastic optimal policy
- A deterministic optimal policy



正确

Correct! Let's say there is a policy π_1 which does well in some states, while policy π_2 does well in others. We could combine these policies into a third policy π_3 , which always chooses actions according to whichever of policy π_1 and π_2 has the highest value in the current state. π_3 will necessarily have a value greater than or equal to both π_1 and π_2 in every state! So we will never have a situation where doing well in one state requires sacrificing value in another. Because of this, there always exists some policy which is best in every state. This is of course only an informal argument, but there is in fact a rigorous proof showing that there must always exist at least one optimal deterministic policy.

- A unique optimal value function



正确

Correct! The Bellman optimality equation is actually a system of equations, one for each state, so if there are N states, then there are N equations in N unknowns. If the dynamics of the environment are known, then in principle one can solve this system of equations for the optimal value function using any one of a variety of methods for solving systems of nonlinear equations. All optimal policies share the same optimal state-value function.

- A unique optimal policy

4. The _____ of the reward for each state-action pair, the dynamics function p , and the policy π is _____ to characterize the value function v_π . (Remember that the value of a policy π at state s is $v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) [r + \gamma v_\pi(s')]$.)

Distribution; necessary

Mean; sufficient

正确

Correct! If we have the expected reward for each state-action pair, we can compute the expected return under any policy.

5. The Bellman equation for a given a policy π : [Select all that apply]

Expresses the improved policy in terms of the existing policy.

Expresses state values $v(s)$ in terms of state values of successor states.

正确

Correct!

Holds only when the policy is greedy with respect to the value function.

6. An optimal policy:

1/1分

- Is not guaranteed to be unique, even in finite Markov decision processes.
- Is unique in every finite Markov decision process.
- Is unique in every Markov decision process.

✓ 正确

Correct! For example, imagine a Markov decision process with one state and two actions. If both actions receive the same reward, then any policy is an optimal policy.

7. The Bellman optimality equation for v_* : [Select all that apply]

1/1分

- Expresses state values $v_*(s)$ in terms of state values of successor states.

✓ 正确

Correct!

- Holds when the policy is greedy with respect to the value function.

- Holds for the optimal state value function.

✓ 正确

Correct!

- Holds when $v_* = v_\pi$ for a given policy π .
- Expresses the improved policy in terms of the existing policy.

8. Give an equation for v_π in terms of q_π and π .

1 / 1 分

- $v_\pi(s) = \sum_a \gamma \pi(a|s) q_\pi(s, a)$
- $v_\pi(s) = \max_a \pi(a|s) q_\pi(s, a)$
- $v_\pi(s) = \max_a \gamma \pi(a|s) q_\pi(s, a)$
- $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$

✓ 正确

Correct!

9. Give an equation for q_π in terms of v_π and the four-argument p .

1 / 1 分

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

✓ 正确

Correct!

10. Let $r(s, a)$ be the expected reward for taking action a in state s , as defined in equation 3.5 of the textbook. Which of the following are valid ways to re-express the Bellman equations, using this expected reward function? [Select all that apply]

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

正确

Correct!

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

正确

Correct!

$q_*(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \max_{a'} q_*(s', a')$

正确

Correct!

$q_\pi(s, a) = r(s, a) + \gamma \sum_{s', a'} p(s'|s, a) \pi(a'|s') q_\pi(s', a')$

正确

Correct!

11. Consider an episodic MDP with one state and two actions (left and right). The **left action** has stochastic reward 1 with probability p and 3 with probability $1 - p$. The **right action** has stochastic reward 0 with probability q and 10 with probability $1 - q$. What relationship between p and q makes the actions equally optimal?

expected value for left action:
$$\begin{aligned} & \text{reward} \\ & 1 \times p + 3 \times (1-p) \\ & \text{probability} \\ & = p + 3 - 3p \\ & = 3 - 2p \end{aligned}$$

$13 + 2p = 10q$

$7 + 3p = 10q$

$7 + 2p = 10q$

$7 + 2p = -10q$

$13 + 3p = 10q$

$13 + 3p = -10q$

$7 + 3p = -10q$

$13 + 2p = -10q$

正确

Correct!