马尔可夫决策过程

吉建民

USTC
jianmin@ustc.edu.cn

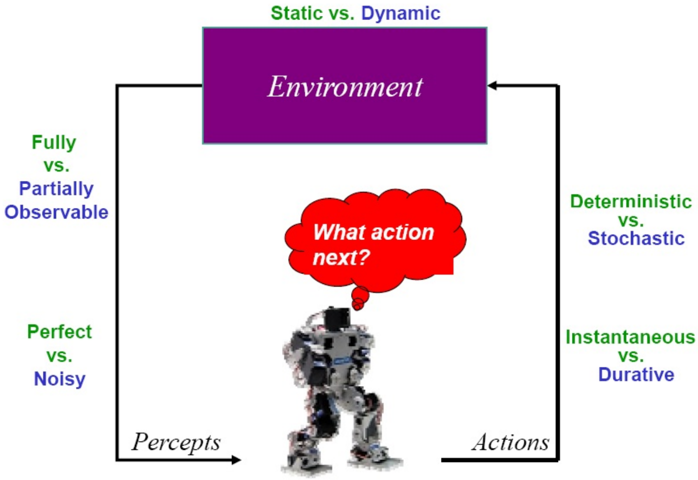
2024 年 4 月 28 日

Used Materials

Disclaimer: 本课件大量采用了 Rich Sutton's RL class, David Silver's Deep RL tutorial 和其他网络课程课件，也采用了 GitHub 中开源代码，以及部分网络博客内容

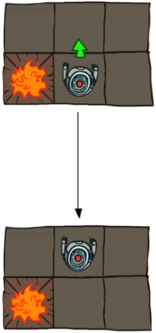
Table of Contents

智能体决策

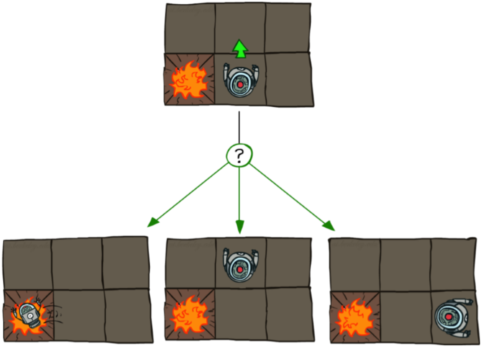

Static vs. Dynamic
Environment
Fully vs. Partially Observable
Perfect vs. Noisy
What action next?
Deterministic vs. Stochastic
Instantaneous vs. Durative
Percepts
Actions

环境的不确定性

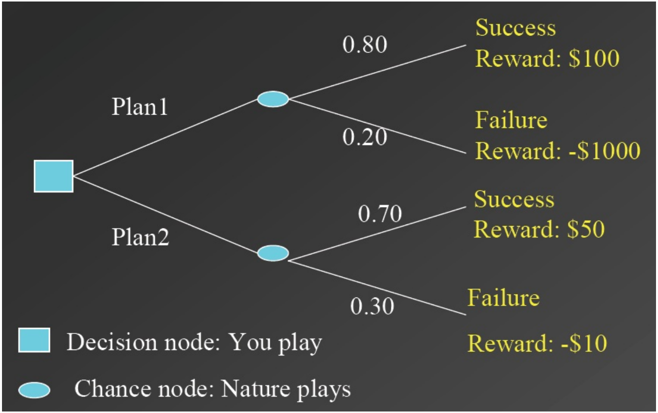
确定性的环境



不确定性的环境



决策树



- Plan1 的期望效用：
  $EU(Plan1) = 100 \times 0.80 - 1000 \times 0.20 = -120$
- Plan2 的期望效用： $EU(Plan2) = 50 \times 0.70 - 10 \times 0.30 = 32$
- 最大期望效用（MEU）：选择行动 A_i 使得期望效用最大

马尔可夫性质

- 当一个随机过程在给定现在状态及所有过去状态情况下，其未来状态的条件概率分布仅依赖于当前状态
- 在给定现在状态时，随机过程与过去状态（历史）是条件独立的

定义

A state s_t is *Markov* if and only if

$$P[s_{t+1} \mid s_t] = P[s_{t+1} \mid s_1, \ldots, s_t]$$

- 状态包含了关于历史的所有信息
- 一旦知道当前状态，过去的历史可以完全抛弃
- 当前状态足够用来预计未来
- 这一性质能够简化问题的表示和求解，但提高了对信息全局性和充分性的要求

MDPs 概述

- 马尔可夫决策过程（Markov Decision Processes, MDPs）
- 人工智能、决策分析、运筹学、控制论和经济学共同关注
- 大量实际问题可以用 MDPs 表达、研究
- MDPs 借助于概率作为刻画不确定性的基本手段
- MDPs 类型：
  - 完全可观察的 MDPs (Full Observable MDPs): FOMDPs
  - 完全不可观察的 MDPs: NOMDPs
  - 部分可观察的 MDPs (Partial Observable MDPs): POMDPs

Table of Contents

Table of Contents

Markov Property

"The future is independent of the past given the present"

Definition

A state S_t is *Markov* if and only if

$$\mathbb{P}[S_{t+1} \mid S_t] = \mathbb{P}[S_{t+1} \mid S_1, ..., S_t]$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future

State Transition Matrix

For a Markov state s and successor state s', the *state transition probability* is defined by

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states s',

$$\mathcal{P} \quad = \textit{from} \quad \begin{array}{c} \textit{to} \\ \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \end{array}$$

where each row of the matrix sums to 1.

Markov Process

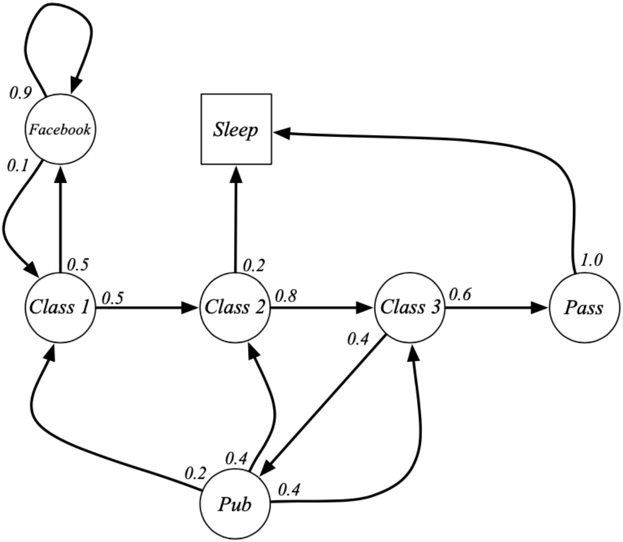
A Markov process is a memoryless random process, i.e. a sequence of random states $S_1, S_2, ...$ with the Markov property.

Definition

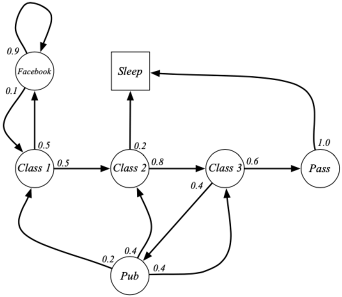
A *Markov Process* (or *Markov Chain*) is a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ is a (finite) set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$

Example: Student Markov Chain



Example: Student Markov Chain Episodes

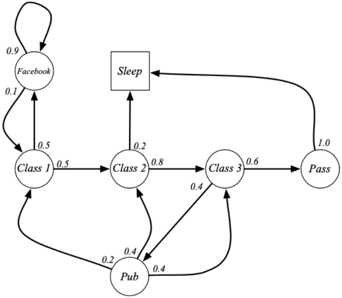


Sample episodes for Student Markov Chain starting from $S_1 = \mathrm{C1}$

$$S_1, S_2, ..., S_T$$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Example: Student Markov Chain Transition Matrix



$$\mathcal{P} = \begin{array}{cc} & \begin{array}{ccccccc} \mathit{C1} & \mathit{C2} & \mathit{C3} & \mathit{Pass} & \mathit{Pub} & \mathit{FB} & \mathit{Sleep} \end{array} \\ \begin{array}{c} \mathit{C1} \\ \mathit{C2} \\ \mathit{C3} \\ \mathit{Pass} \\ \mathit{Pub} \\ \mathit{FB} \\ \mathit{Sleep} \end{array} & \begin{bmatrix} & 0.5 & & & & 0.5 & \\ & & 0.8 & & & & 0.2 \\ & & & 0.6 & 0.4 & & \\ & & & & & & 1.0 \\ 0.2 & 0.4 & 0.4 & & & & \\ 0.1 & & & & & 0.9 & \\ & & & & & & 1 \end{bmatrix} \end{array}$$

Table of Contents

Markov Reward Process

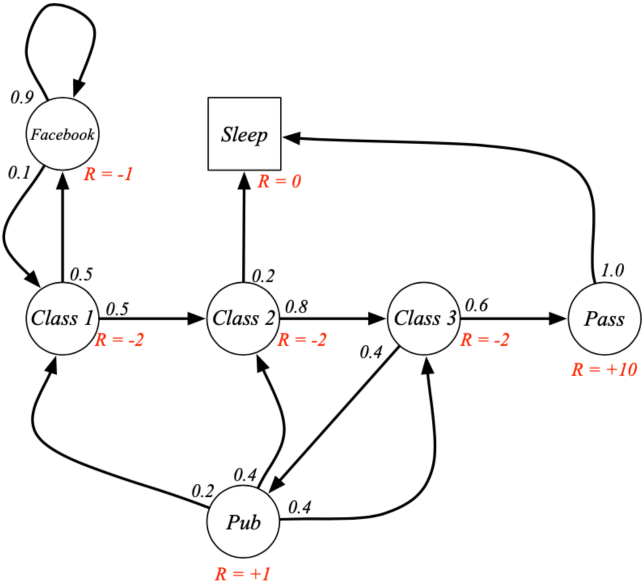
A Markov reward process is a Markov chain with values.

Definition

A *Markov Reward Process* is a tuple $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$
- γ is a discount factor, $\gamma \in [0, 1]$

Example: Student MRP



Return

Definition

The *return* G_t is the total discounted reward from time-step t.

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- The *discount* $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R after $k+1$ time-steps is $\gamma^k R$.
- This values immediate reward above delayed reward.
  - γ close to 0 leads to "myopic" evaluation
  - γ close to 1 leads to "far-sighted" evaluation

Why discount?

Most Markov reward and decision processes are discounted. Why?

- Mathematically convenient to discount rewards
- Avoids infinite returns in cyclic Markov processes
- Uncertainty about the future may not be fully represented
- If the reward is financial, immediate rewards may earn more interest than delayed rewards
- Animal/human behaviour shows preference for immediate reward
- It is sometimes possible to use *undiscounted* Markov reward processes (i.e. $\gamma = 1$), e.g. if all sequences terminate.

Value Function

The value function $v(s)$ gives the long-term value of state s

Definition

The *state value function* $v(s)$ of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right]$$

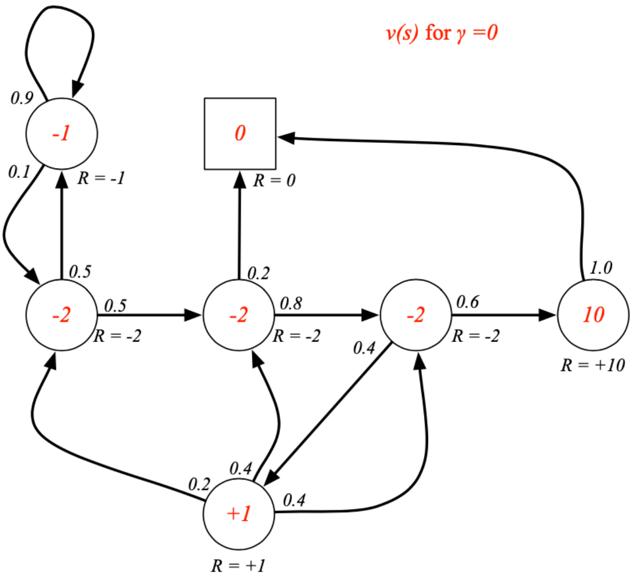
Example: Student MRP Returns

Sample returns for Student MRP:
Starting from $S_1 = C1$ with $\gamma = \frac{1}{2}$

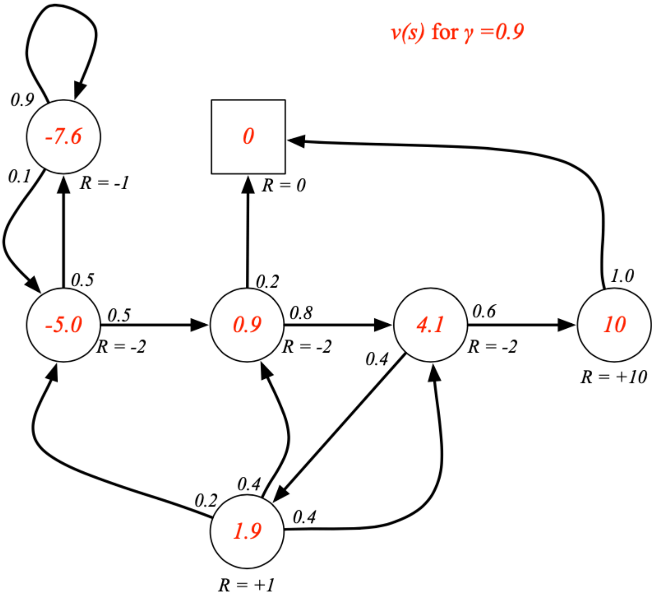
$$G_1 = R_2 + \gamma R_3 + ... + \gamma^{T-2} R_T$$

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$=$	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$=$	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.41
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$		
FB FB FB C1 C2 C3 Pub C2 Sleep		$=$	-3.20

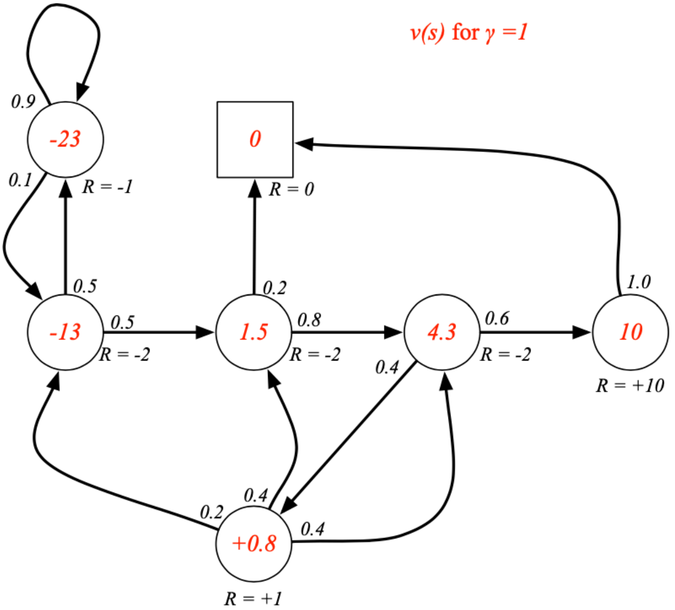
Example: State-Value Function for Student MRP (1)



Example: State-Value Function for Student MRP (2)



Example: State-Value Function for Student MRP (3)



Bellman Equation for MRPs (1)

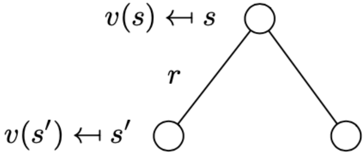
The value function can be decomposed into two parts:

- immediate reward R_{t+1}
- discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}
$$

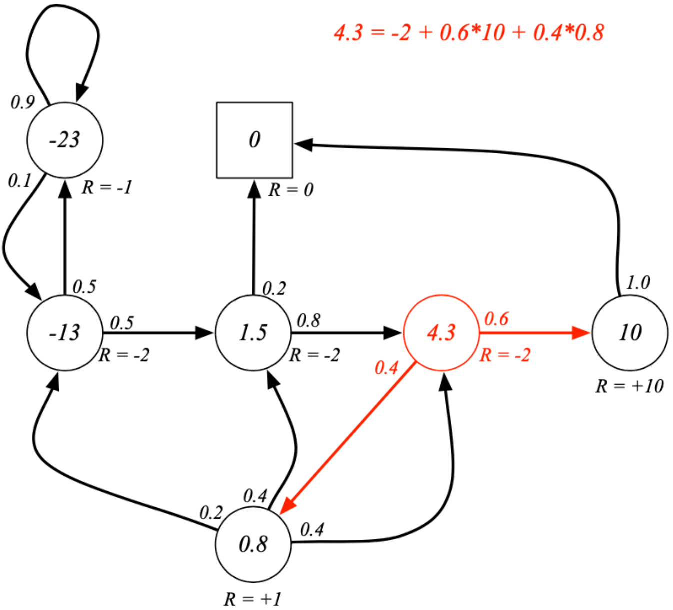
Bellman Equation for MRPs (2)

$$v(s) = \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]$$



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

Example: Bellman Equation for Student MRP



Bellman Equation in Matrix Form

The Bellman equation can be expressed concisely using matrices,

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

where v is a column vector with one entry per state

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \dots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix}$$

Solving the Bellman Equation

- The Bellman equation is a linear equation
- It can be solved directly:

$$
\begin{aligned}
v &= \mathcal{R} + \gamma \mathcal{P} v \\
(I - \gamma \mathcal{P})\, v &= \mathcal{R} \\
v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R}
\end{aligned}
$$

- Computational complexity is $O(n^3)$ for n states
- Direct solution only possible for small MRPs
- There are many iterative methods for large MRPs, e.g.
  - Dynamic programming
  - Monte-Carlo evaluation
  - Temporal-Difference learning

Table of Contents

Markov Decision Process

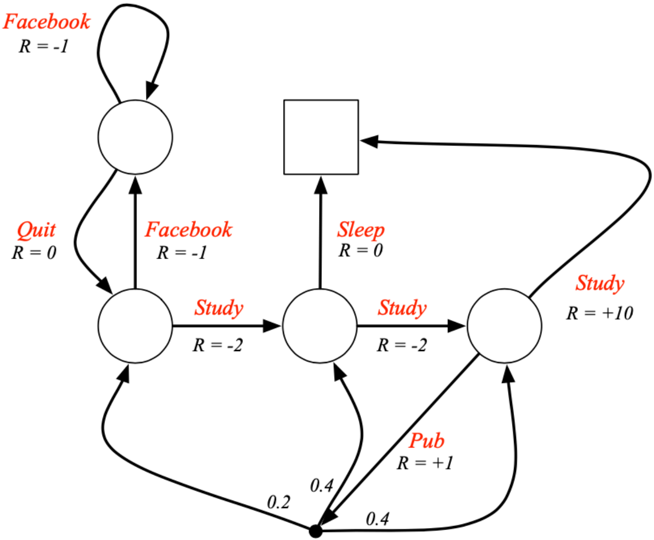
A Markov decision process (MDP) is a Markov reward process with decisions. It is an *environment* in which all states are Markov.

Definition

A *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{P}$ is a state transition probability matrix,
  $\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

Example: Student MDP



Policies (1)

Definition

A *policy* π is a distribution over actions given states,

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

- A policy fully defines the behaviour of an agent
- MDP policies depend on the current state (not the history)
- i.e. Policies are *stationary* (time-independent), $A_t \sim \pi(\cdot|S_t), \forall t > 0$

Policies (2)

- Given an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ and a policy π
- The state sequence $S_1, S_2, \ldots$ is a Markov process $\langle \mathcal{S}, \mathcal{P}^\pi \rangle$
- The state and reward sequence $S_1, R_2, S_2, \ldots$ is a Markov reward process $\langle \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma \rangle$
- where

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}^a_{ss'}$$

$$\mathcal{R}^\pi_s = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}^a_s$$

Value Function

Definition

The *state-value function* $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π

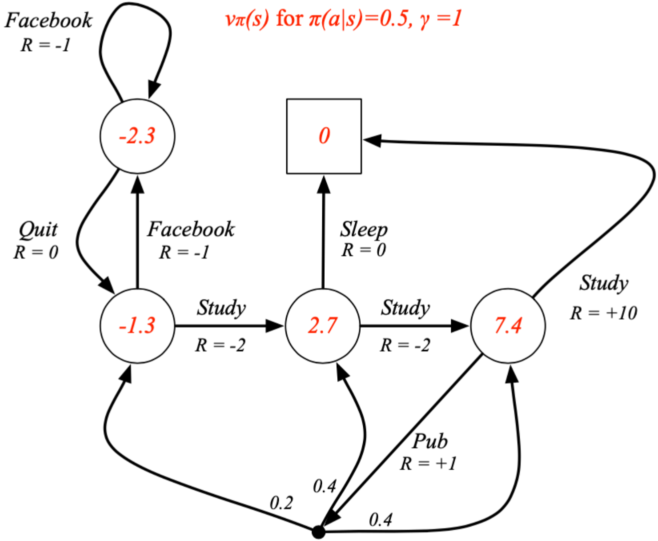
$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right]$$

Definition

The *action-value function* $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

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

Example: State-Value Function for Student MDP



Bellman Expectation Equation

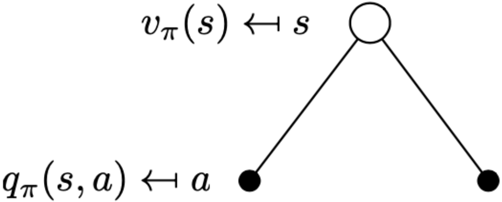
The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right]$$

The action-value function can similarly be decomposed,

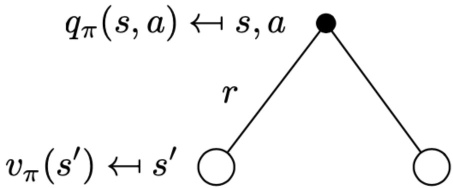
$$q_\pi(s, a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) \mid S_t = s, A_t = a\right]$$

Bellman Expectation Equation for v_π



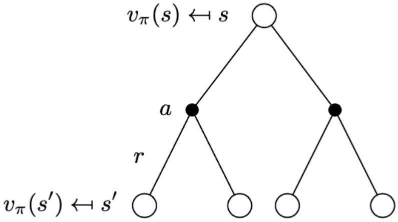
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Bellman Expectation Equation for q_π



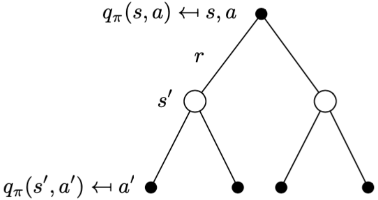
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Bellman Expectation Equation for v_π (2)



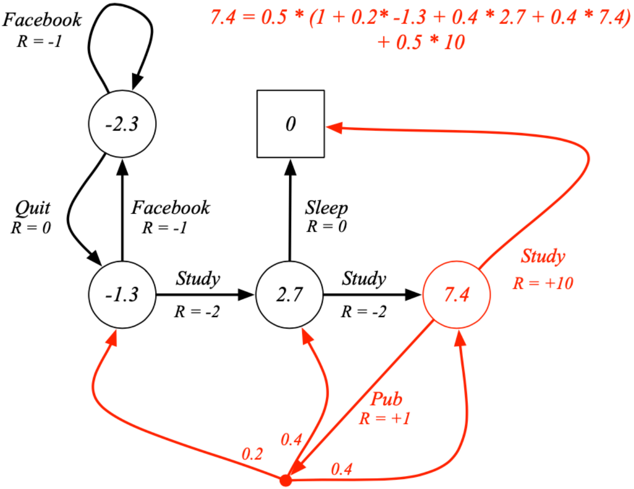
$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

Bellman Expectation Equation for q_π (2)



$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s',a')$$

Example: Bellman Expectation Equation in Student MDP



Bellman Expectation Equation (Matrix Form)

The Bellman expectation equation can be expressed concisely using the induced MRP,

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

with direct solution

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

Optimal Value Function

Definition

The *optimal state-value function* $v_*(s)$ is the maximum value function over all policies

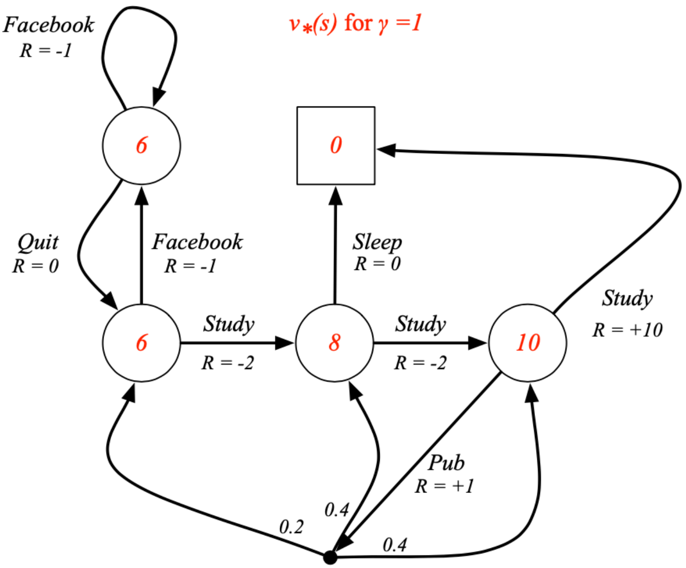
$$v_*(s) = \max_\pi v_\pi(s)$$

The *optimal action-value function* $q_*(s, a)$ is the maximum action-value function over all policies

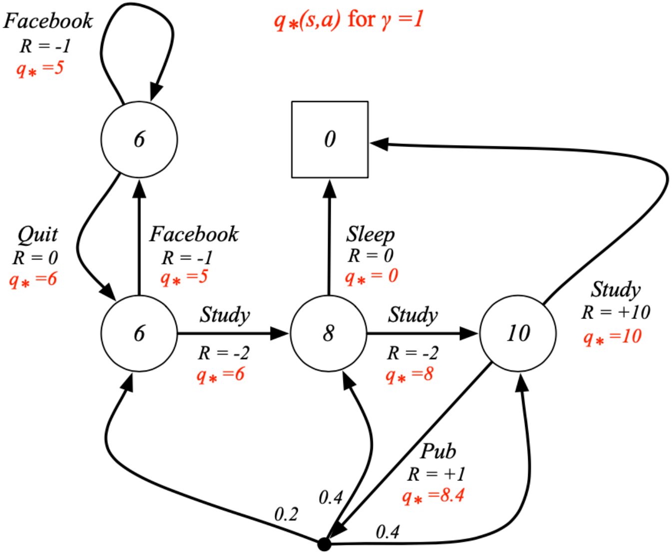
$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

- The optimal value function specifies the best possible performance in the MDP.
- An MDP is "solved" when we know the optimal value fn.

Example: Optimal Value Function for Student MDP



Example: Optimal Action-Value Function for Student MDP



Optimal Policy

Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } v_\pi(s) \geq v_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- *There exists an optimal policy* π_* *that is better than or equal to all other policies,* $\pi_* \geq \pi, \forall \pi$
- *All optimal policies achieve the optimal value function,* $v_{\pi_*}(s) = v_*(s)$
- *All optimal policies achieve the optimal action-value function,* $q_{\pi_*}(s, a) = q_*(s, a)$

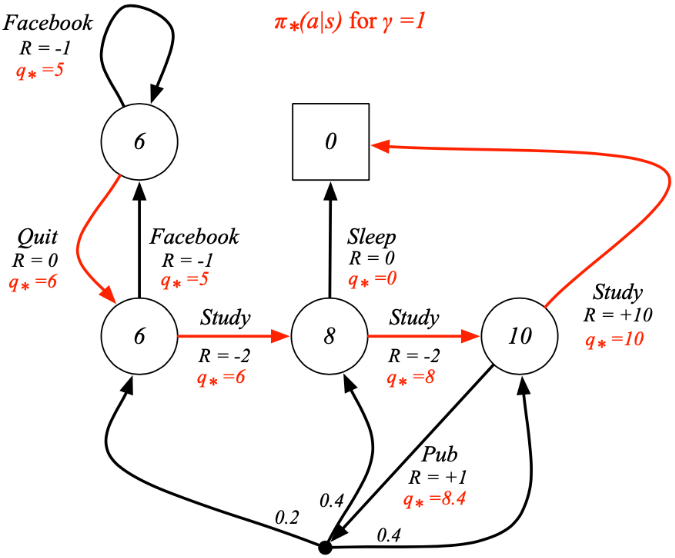
Finding an Optimal Policy

An optimal policy can be found by maximising over $q_*(s, a)$,

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ q_*(s, a) \\ 0 & \textit{otherwise} \end{cases}$$

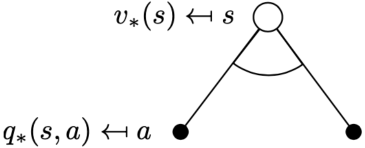
- There is always a deterministic optimal policy for any MDP
- If we know $q_*(s, a)$, we immediately have the optimal policy

Example: Optimal Policy for Student MDP



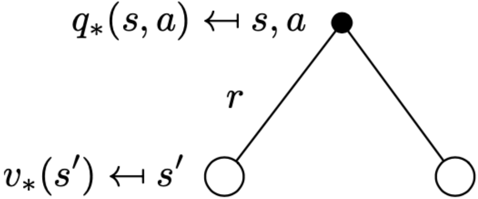
Bellman Optimality Equation for v_*

The optimal value functions are recursively related by the Bellman optimality equations:



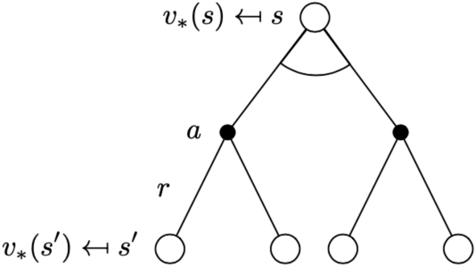
$$v_*(s) = \max_a q_*(s,a)$$

Bellman Optimality Equation for q_*



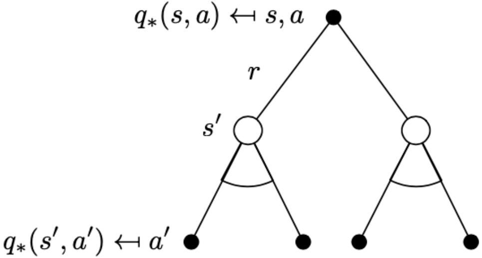
$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for v_* (2)



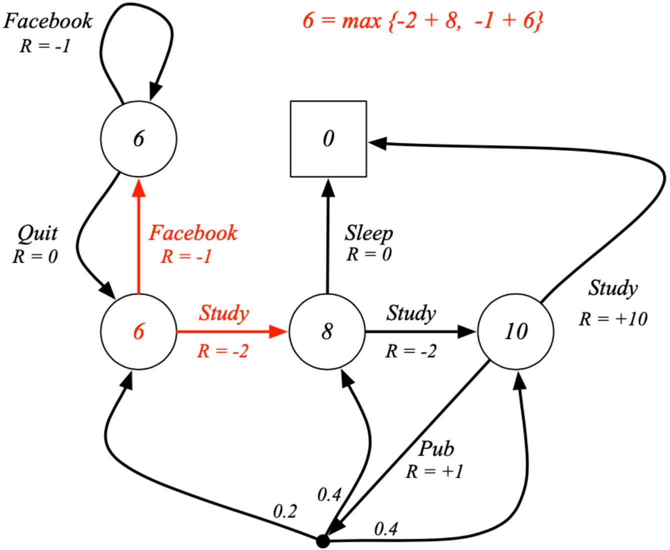
$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for q_* (2)



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

Example: Bellman Optimality Equation in Student MDP



Solving the Bellman Optimality Equation

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - Q-learning
  - Sarsa

Table of Contents

POMDPs

A Partially Observable Markov Decision Process is an MDP with hidden states. It is a hidden Markov model with actions.

Definition

A *POMDP* is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \gamma \rangle$

- $\mathcal{S}$ is a finite set of states
- $\mathcal{A}$ is a finite set of actions
- $\mathcal{O}$ is a finite set of observations
- $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}^a_{ss'} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
- $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
- $\mathcal{Z}$ is an observation function, $\mathcal{Z}^a_{s'o} = \mathbb{P}[O_{t+1} = o \mid S_{t+1} = s', A_t = a]$
- γ is a discount factor $\gamma \in [0, 1]$.

Belief States

Definition

A *history* H_t is a sequence of actions, observations and rewards,

$$H_t = A_0, O_1, R_1, ..., A_{t-1}, O_t, R_t$$

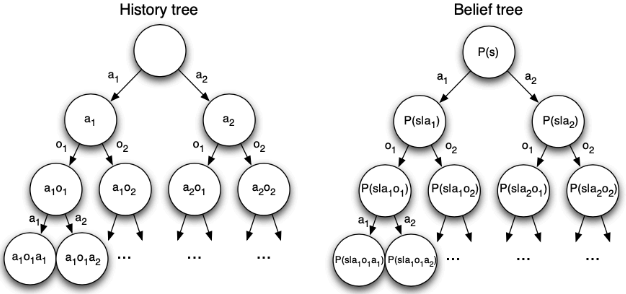
Definition

A *belief state* $b(h)$ is a probability distribution over states, conditioned on the history h

$$b(h) = (\mathbb{P}\left[S_t = s^1 \mid H_t = h\right], ..., \mathbb{P}\left[S_t = s^n \mid H_t = h\right])$$

Reductions of POMDPs

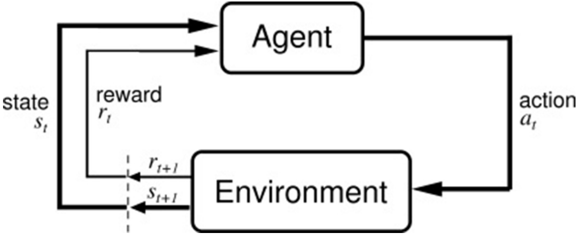
- The history H_t satisfies the Markov property
- The belief state $b(H_t)$ satisfies the Markov property



- A POMDP can be reduced to an (infinite) history tree
- A POMDP can be reduced to an (infinite) belief state tree

Table of Contents

MDP 基本框架



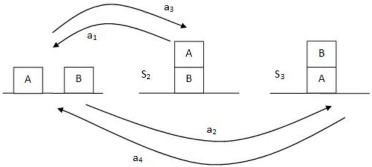
- MDPs 立足于 agent 与环境的直接交互
- 只考虑离散时间，假设 agent 与环境的交互过程可分解为一系列“阶段”，每个阶段由“感知–决策–行动”构成
- 已知环境模型

MDP 模型

- MDP 模型是一个四元组 (S, A, T, R)，其中
  - S 是一个有限集，其中每个元素 $s \in S$ 代表一个状态
  - A 是一个有限集，其中每个元素 $a \in A$ 代表一个行动
  - $T: S \times A \to \Pi(S)$ 称为状态转移函数，将每一对“状态　行动”映射为 S 上的一个概率分布，用记号 $T(s, a, s')$ 表示在状态 s 上执行 a 达到 s' 的概率
  - $R: S \times A \to \mathcal{R}$ 称为回报（reword）函数，$R(s, a)$ 表示在 s 上执行行动 a 所得到的即时回报（直接回报）
- 如何规定一个 MDP 模型是一个具体应用问题，不是 MDPs 本身研究的内容
- MDPs 研究的主题是，给定一个 MDP 模型，如何求最优策略，即期望回报最大的策略

积木世界

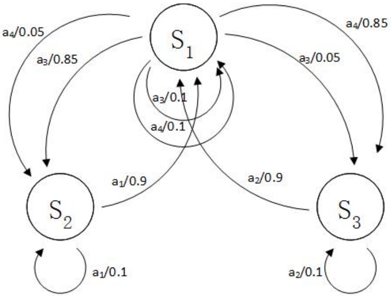
- 积木世界只有两块积木 A 和 B
- $S = \{s_1, s_2, s_3\}$
- $A = \{a_1, a_2, a_3, a_4\}$
    - $a_1 = move(A, B, F)$
    - $a_2 = move(B, A, F)$
    - $a_3 = move(A, F, B)$
    - $a_4 = move(B, F, A)$



积木世界 (con't)

- 规定 T

	a_1			a_2			a_3			a_4		
	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3	s_1	s_2	s_3
s_1	1	0	0	1	0	0	0.1	0.85	0.05	0.1	0.05	0.85
s_2	0.9	0.1	0	0	1	0	0	1	0	0	1	0
s_3	0	0	1	0.9	0	0.1	0	0	1	0	0	1



积木世界 (con't)

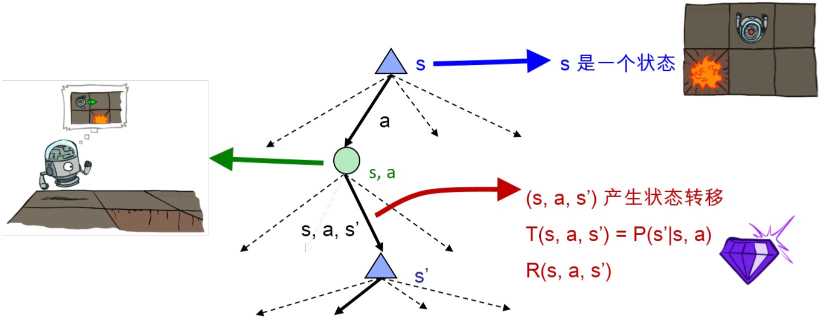
- 规定 R: “达到一个状态获得的回报”和“执行一个行动的代价”的综合
- 假设 s_2 是“最有益的”状态，s_3 是“最有害的”，s_1 是中性的，每个行动都有代价，而且“放上去”比“拿下来”代价大，R 的值仅仅反映“档次”

	a_1	a_2	a_3	a_4
s_1	-1	-1	1	-2
s_2	-2	-1	-1	-1
s_3	-1	0	-1	-1

最优策略（Optional Policy）

- “长期”分两类：
  - 无限阶段问题（例如，网络维护）
  - 有限阶段问题
- 策略分为“平稳的（stating）”与“非平稳的（non-stating）”
  - 平稳策略可表示为 $\pi : S \to A$
  - 非平稳策略 $\pi = \{\pi_t, \ldots, \pi_1\}$，每个 π_i 是一个 S 到 A 的映射。最后一次决策用 π_1，产生的行动是 $\pi_1(s)$，第一次用 π_t

MDP 的求解



问题：给定一个MDP模型，如何求解出最优策略？

值函数

- 任给策略 π，计算预期（长期）回报，用值函数 (value function) 度量预期回报
- 令 $V_{\pi,t}(s)$ 表示从 s 开始，执行策略 π 共 t 个阶段所获得的预期累计回报
  - 当 $t=1$ 时
  $$V_{\pi,1}(s) = R(s, \pi_1(s))$$
  - 当 $t=2$ 时
  $$V_{\pi,2}(s) = R(s, \pi_2(s)) + \Sigma_{s'} T(s, \pi_2(s), s') \cdot R(s', \pi_1(s'))$$
  - 多步情况
  $$V_{\pi,t}(s) = R(s, \pi_t(s)) + \Sigma_{s'} T(s, \pi_t(s), s') \cdot V_{\pi,t-1}(s')$$

值函数（con't）

- 有时采用“折扣准则”：折扣因子 $\gamma\ (<1)$ （在无限阶段中更有必要用，保证收敛）

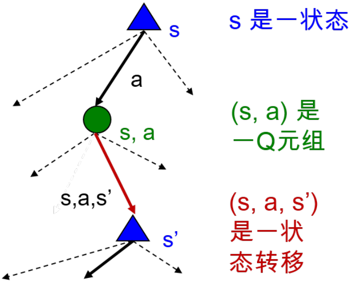
$$V_{\pi,t}(s) = R(s,\pi_t(s)) + \gamma\Sigma_{s'} T(s,\pi_t(s),s') \cdot V_{\pi,t-1}(s')$$

- 给定一个 MDP 模型和一个策略 π，由此可得 $V_{\pi,t}$，所有 π 中，$V_{\pi,t}$ 最大的 π 称为最优策略，记为 π^*，对应的值函数记为 V_t^*，反过来，如果知道了 V_t^*，也可得到 π^*

$$\pi_t^*(s) = argmax_a \left[R(s,a) + \gamma\Sigma_{s'} T(s,a,s') V_{t-1}^*(s')\right]$$

最优依据

- 定义1：状态s的**最优值函数** $V^*(s)$为，以s为初始状态，执行最优策略获得的期望收益值
- 定义2：元组(s, a)的**最优Q函数** $Q^*(s, a)$为，以s为初始状态，执行动作a后，按最优策略执行获得的期望收益值
- 定义3：**最优策略**π^*为执行过程中最大化期望收益值的策略



贝尔曼等式

- 利用贝尔曼等式，可以用递归的方式来计算状态的**期望收益值**：

$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

Richard Bellman
(1920 - 1984)

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

最优值函数的计算

值迭代算法

- 值迭代（value iteration）算法求 V_t^* 序列，借助于辅助 $Q_t^a(s)$，其直观含义为：在 s 执行 a，然后执行 $t-1$ 步的最优策略所产生的预期回报

对所有 $s \in S$, $V_0(s) := 0$; $t := 0$;

loop

$t := t + 1$;

loop 对所有 $s \in S$

loop 对所有 $a \in A$

$$Q_t^a(s) := R(s,a) + \gamma \Sigma_{s'} T(s,a,s') V_{t-1}(s')$$

end loop

$V_t(s) := max_a\, Q_t^a(s)$ (Bellman equation)

end loop

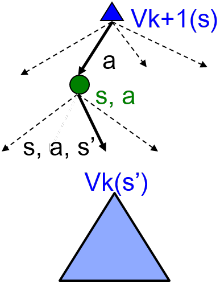
until $|V_t(s) - V_{t-1}(s)| < \epsilon$ 对所有 $s \in S$ 成立

值迭代算法 (con't)

1. 对所有状态s，初始化 $V_0(s) = 0$
2. 给定一组$V_k(s)$的值，对所有的状态进行下列迭代更新：

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

3. 不断重复步骤2，直到收敛



本质上是一种动态规划的方法

值迭代算法的一些结论

- 每一步值迭代的计算复杂度是 $O(|A||S|^2)$
- 根据不动点理论，算法将最终收敛到最优值
  - 收敛到最优值所需的迭代次数为：

    poly(|S|, |A|, 1/(1-γ))

策略迭代算法

- 策略迭代算法
  1. 先给定初始策略 π，求解线性方程，计算出 π 相应的值函数 V
  $$V(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s, a, s') V(s').$$
  2. 基于计算出的值函数 V，按下式更新策略
  $$\pi(s) \leftarrow argmax_a \left\{ R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s') \right\}$$
  3. 不断重复步骤 1，2，直到收敛
- 策略迭代算法最终收敛到最优值
  - 收敛到最优值所需的迭代不大于 $|A|^{|S|}$（确定性策略的总数目）

讨论

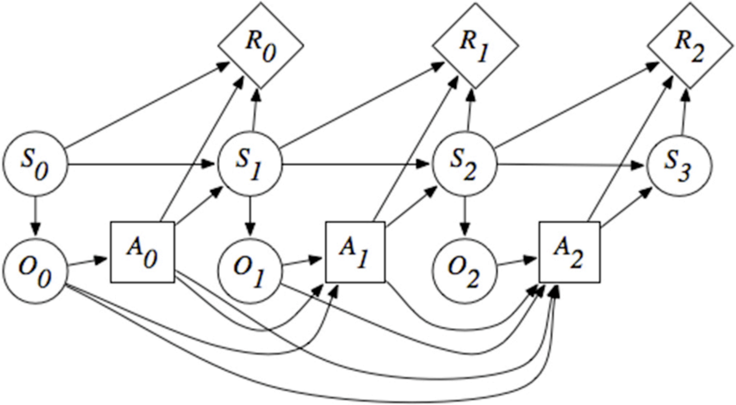
- 一个具体的 MDP 模型刻画了一个具体实际问题，即提供了该问题的一个数学模型
- MDPs 问题归结为最优策略的求解问题，三个要点：
  - 将 agents 的行动归结为策略的执行
  - 将 agents 的决策问题归结为求最优策略
  - 以“效用最大化”作为“最优”的基本标准
- 经典规划与 MDPs 的联系（异同）
  - 行动效果确定 vs. 不确定
  - 经典规划实际上假定环境是完全不可观察
  - 经典规划是以达到目标状态为“成功”标准，没有其他区别；而 MDPs 以效用最大化为“成功”标准，它的框架允许精华的，也可以通过优化途径求 π^*
- 一个经典规划问题可以描述为一个特殊的 NOMDP 问题，每个目标状态 s 上，定义一个“驻留行动”使得 $T(s, a, s) = 1$

Table of Contents

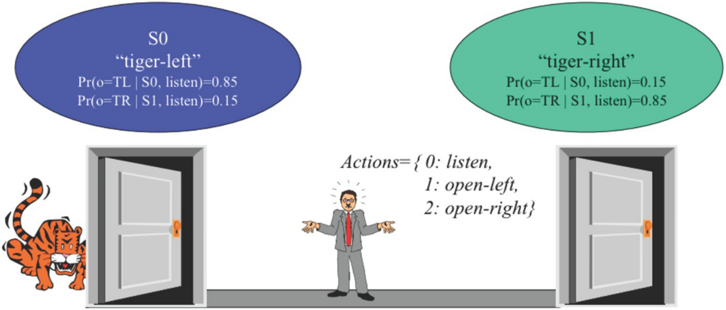
POMDPs 基本定义

- MDPs 刻画了行动的不确定性：事先不知道，事后知道；观察是确定的，知道行动的效果
- 一般情况下，行动和观察都是不确定的
- 一个 POMDP 模型是一个六元组 (S, A, T, R, Ω, O)，其中 S, A, T, R 的定义与 MDP 模型相同；Ω 是一个有限集，其中元素称为"观察"；$O: S \times A \to \Pi(\Omega)$ 称为观察函数，有时记为 $O(s', a, o)$，a 代表执行的行动，s' 是 a 达到的结果状态，o 在上述条件下出现的观察，$O(s', a, o)$ 是执行 a 达到 s' 观察到 o 的概率 $P_r(o \mid s', a)$
- 在 POMDP 问题中，"环境"被看成一个"黑箱"，一个状态是黑箱某个时期的内部情况，观察是这个黑箱的"输出"

POMDPs



老虎问题



Reward Function

- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

信念状态

- 令 $B = \Pi(S)$ 为 S 上所有状态分布的集合，$b \in B$ 称为一个“信念状态”（belief state）
- 假设 $S = \{s_1, \ldots, s_n\}$，则 $\forall i$：$0 \leq b(s_i) \leq 1$，$\Sigma_{i=1}^{n} b(s_i) = 1$
  $b = (b(s_1), \ldots, b(s_n))$，$b(s_i)$ 表示环境处于 s_i 状态的概率
- 当 $|S|$ 有穷时，$\pi(S)$ 是无穷的

信念状态 (con't)

- 给定一个 POMDP 模型，一个 $b \in B$，一个 $a \in A$ 和一个 $o \in \Omega$，可以计算（估计）在 b 下执行 a 得到 o 时，环境处于状态 $s' \in S$ 的概率：

$$
\begin{aligned}
b'(s') = P_r(s' \mid o, a, b) &= \frac{P_r(o \mid s', a, b)\, P_r(s' \mid a, b)}{P_r(o \mid a, b)} \\
&= \frac{P_r(o \mid s', a)\, \Sigma_s P_r(s' \mid a, b, s) P_r(s \mid a, b)}{P_r(o \mid a, b)} \\
&= \frac{O(s', a, o)\, \Sigma_s T(s, a, s') b(s)}{P_r(o \mid a, b)}
\end{aligned}
$$

b' 是 o, a, b 的函数，记为 $b' = SE(b, a, o)$，称为“状态估计函数”

信念状态 (con't)

- 对 $b'(s')$ 中分母推导：

$$
\begin{aligned}
P_r(o \mid a, b) &= \Sigma_{s'} O(s', a, o) P_r(s' \mid a, b) \\
&= \Sigma_{s'} O(s', a, o) \Sigma_s T(s, a, s') b(s)
\end{aligned}
$$

  是给定 a, b 下出现 o 的“平均概率”
- $b'(s')$ 中分子是 a, b, s' 下出现 o 的概率
- $P_r(o \mid a, b)$ 是保证 $\Sigma_{s'} b'(s') = 1$ 的“正规化因子”
- $\Sigma_s T(s, a, s') b(s) = P_r(s' \mid a, b)$ 是不考虑观察而得到的下一状态为 s' 的概率，而 $b'(s')$ 是考虑了观察 $O(s', a, o)$ 所得出的下一状态的概率。两者关系：

$$
b'(s') = \frac{O(s', a, o)}{P_r(o \mid a, b)} \cdot P_r(s' \mid a, b)
$$

- 若 $O(s', a, o) > P_r(o \mid a, b)$，则 $b'(s') > P_r(s' \mid a, b)$。在 s' 下出现 o 的概率，高于出现 o 的平均概率

信念状态 (con't)

- b' 包含了观察和模型的全部信息：
  1. 上一时刻的全部信息 b;
  2. 模型信息 T;
  3. 观察信息 $O(s', a, o)$, $s' \in S$
- 因此，在 POMDP 问题中，B 的作用相当于 MDP 中的 S

积木世界续

- 考虑积木世界的 POMDP 模型，其中 S, A, T, R 与前例相同，$\Omega = \{o_1, o_2\}$，假定 agent 的观察能力与行动无关，而且不能区分 s_2 与 s_3
- 定义 O: $\forall a$

$$O(s_1, a, o_1) = 1,\ O(s_2, a, o_1) = 0,\ O(s_3, a, o_1) = 0$$
$$O(s_1, a, o_2) = 0,\ O(s_2, a, o_2) = 1,\ O(s_3, a, o_2) = 1$$

- 设初始信念状态 $b = (0.9, 0, 0.1)$，执行 $a_3 = move(A, F, B)$，观察为 o_2

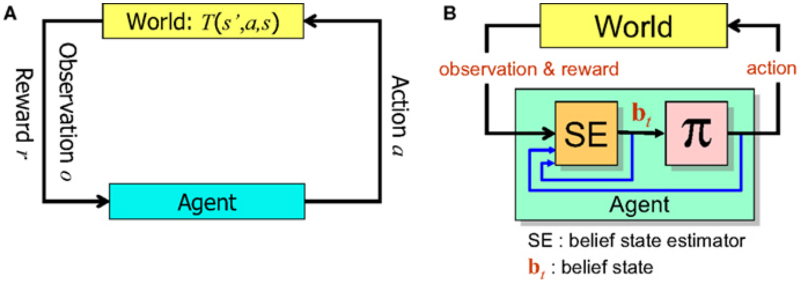
积木世界续 (con't)

计算下一时刻信念状态:

$$
\begin{aligned}
P_r(o_2 \mid a_3, b) &= \Sigma_{s'} O(s', a_3, o_2) P_r(s' \mid a_3, b) \\
&= P_r(s_2 \mid a_3, b) + P_r(s_3 \mid a_3, b) \\
&= \Sigma_s T(s, a_3, s_2) b(s) + \Sigma_s T(s, a_3, s_3) b(s) \\
&= (0.9 \times 0.85 + 0.1 \times 0) + (0.9 \times 0.05 + 1 \times 0.1) \\
&= 0.765 + 0.145 = 0.81
\end{aligned}
$$

$$
\begin{aligned}
b'(s_1) &= 0 \\
b'(s_2) &= \frac{0.765}{0.81} = 0.841 \\
b'(s_3) &= \frac{0.145}{0.81} = 0.159
\end{aligned}
$$

MDP vs. POMDP



A
World: $T(s',a,s)$
Reward r
Observation o
Action a
Agent
B
World
observation & reward
action
SE
$\mathbf{b}_t$
π
Agent
SE : belief state estimator
$\mathbf{b}_t$: belief state

POMDP to MDP

- B 在 POMDPs 中相当于 MDPs 中 S 的作用，设法将一个 POMDP 问题转化为一个变形的 MDP 问题，使得相应的 MDP 模型为 (B, A, τ, ρ)，$\tau : B \times A \to \Pi(B)$，$\rho : B \times A \to \mathcal{R}$

$$\tau(b, a, b') =_{df} P_r(b' \mid b, a) = \Sigma_o P_r(b' \mid a, b, o) P_r(o \mid a, b)$$

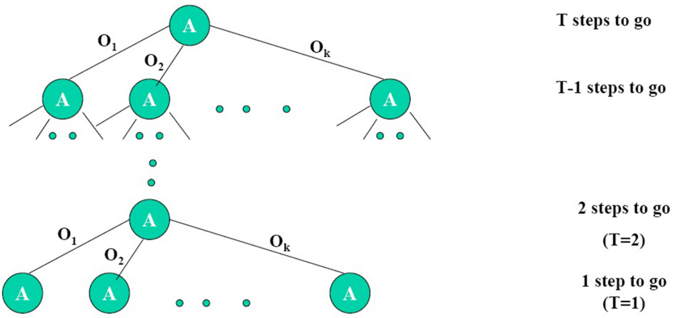
$$P_r(b' \mid a, b, o) = \begin{cases} 1, & \text{if } SE(b, a, o) = b' \\ 0, & \text{otherwise} \end{cases}$$

$$\rho(b, a) =_{df} \Sigma_s b(s) R(s, a)$$

- 由于 B 是无穷的，很难直接用值迭代算法求解

POMDPs 的策略表示

- POMDPs 的策略表示
  - $B \to A$（单步），多步的用 $\pi = (\pi_t, \ldots, \pi_1)$，但 B 是无穷的
  - 策略树：一个 t 层策略树（policy tree）共有 t 层节点，根节点上标记代表第一个行动，它的分枝代表执行该行动后得到的观察，...，是一种“有穷表示”



POMDPs 的值函数

- 采用策略树表示，令 p 是一个 t 步策略树，$V_p(s)$ 是在 s 上执行 p 的预期效用（扔采用折扣准则）:

$$
\begin{aligned}
V_p(s) &= R(s, a(p)) + \gamma \times \text{后 } t-1 \text{ 步的预期效用} \\
&= R(s, a(p)) + \gamma \Sigma_{s'} P_r(s' \mid s, a(p)) \Sigma_{o_i} P_r(o_i \mid s', a(p)) V_{o_i(P)}(s') \\
&= R(s, a(p)) + \gamma \Sigma_{s'} T(s, a(p), s') \Sigma_{o_i} O(s', a(p), o_i) V_{o_i(P)}(s')
\end{aligned}
$$

  $o_i(p)$ 是 p 的由分枝 o_i 引出的 $t-1$ 步子树
- 对任意 $b \in B$，定义 $V_p(b) = \Sigma_s b(s) V_p(s)$ 为在 b 上执行 p 的预期效用
- 令 P_t 为所有 t 步策略树的集合，P_t 有穷，b 上最优 t 阶段效用定义为

$$V_t(b) = max_{p \in P_t} V_p(b)$$

  使 $V_p(b) = V_t(b)$ 的 p 是 b 上的最佳策略

POMDPs 的值函数 (con't)

- 考虑在整个 B 上的最佳策略，一般要用策略树集来表示（因为，B 无穷、连续分布）
- $P \subseteq P_t$，满足 $\forall b \in B, \exists p \in P$ 使得 $V_p(b) \geq V_{p'}(b), \forall p' \in P_t$
- $|P_t|$ 是有穷的，

$$|P_t| = |A|^{\frac{|\Omega|^t - 1}{|\Omega| - 1}} = |A|^{|\Omega|^{t-1}}$$

- 若 $|A| = |\Omega| = t = 10$, $|P_t| \approx 10^{10^9}$，而深蓝才 10^{22}.

POMDPs 的值函数 (con't)

- S 是固定的、有穷的，B 是无穷的
- 设 $S = \{s_1, \ldots, s_n\}$，$b = \{b(s_1), \ldots, b(s_n)\}$，$0 \leq b(s_i) \leq 1$，$\Sigma_i b(s_i) = 1$，“变化的”不是 s，而是 b

$$V_p(b) = \Sigma_s b(s) V_p(s) = b(s_1) V_p(s_1) + \cdots + b(s_n) V_p(s_n)$$

$V_p(b)$ 是 $b(s_1), \ldots, b(s_n)$ 的一个线性函数。$V_p(s_i)$ 与 b 无关

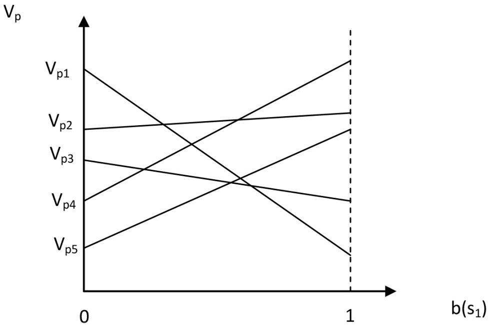
- 例：考虑 $|S| = 2$

$$\begin{aligned} V_p(b) &= b(s_1) V_p(s_1) + b(s_2) V_p(s_2) \\ &= V_p(s_1) b(s_1) + V_p(s_2)(1 - b(s_1)) \\ &= (V_p(s_1) - V_p(s_2)) b(s_1) + V_p(s_2) \end{aligned}$$

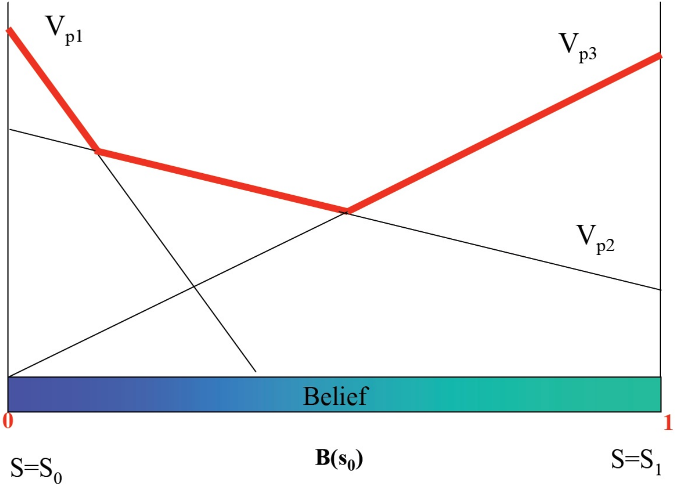
对不同的 p 有不同的 $V_p(s_1)$，$V_p(s_2)$，对应不同的线段

最大期望效用函数

- 由所有 $p \in P_t$ 产生的 $V_p(b)$ 的“上表面”可以得到 $V_t(b)$，整个 B 上最优策略集合，以及每个最优策略的“最优区域”，用线性规划可确定
- 对任意 $|S|$，情况类似，$V_t(b)$ 是一个“分块线性的凸函数”



最大期望效用函数 (con't)



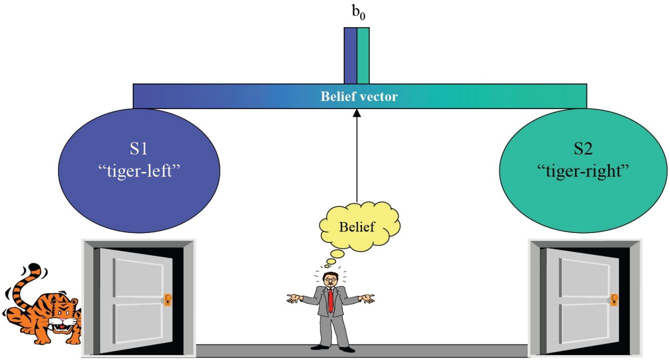
最大期望效用函数 (con't)

- 由 $V_p(b)$ 构成“上表面”的所有 p 组成的集合（代表 $V_t(b)$），称为最大期望效用函数/最佳值函数 $V_t(b)$ 的“最简表示”，记为 PV_t，每个 $p \in PV_t$ 是 B 的某个区域上的最优策略树，p 的“最优区域”是

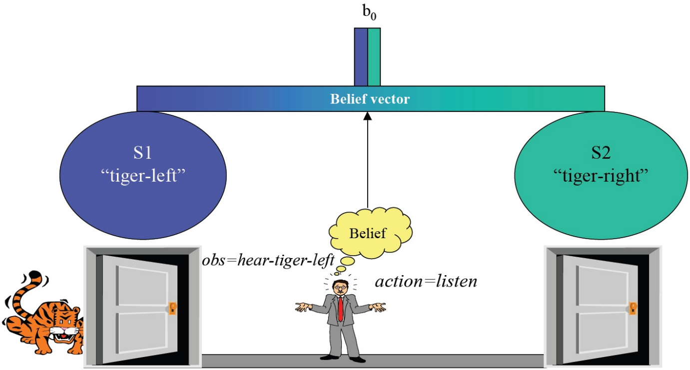
$$\{b \in B \mid \forall p' \in PV_t \setminus \{p\}, \Sigma_s b(s) V_p(s) > \Sigma_s b(s) V_{p'}(s)\}$$

  可以用线性规划方便求出
- PV_t 的任何真子集都无法表达 $V_t(b)$，称为不完备的，与“上表面”有共同点的所有 $p \in P_t$ 的集合记为 P^+V_t，称为“次简表示”
  - $PV_t \subseteq P^+V_t$;
  - $\forall p \in P^+V_t, \exists b \in B, \forall p' \in P_t$: $V_p(b) \geq V_{p'}(b)$;
  - 使用 PV_t/P^+V_t 的动机是，通常它们远远小于 P_t

老虎问题 (con't)

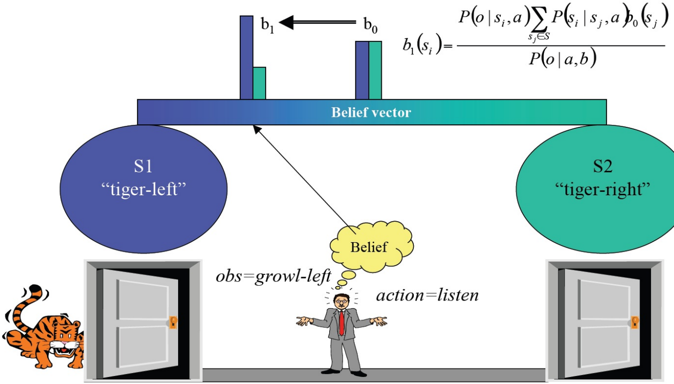


老虎问题 (con't)



b0
Belief vector
S1
"tiger-left"
S2
"tiger-right"
Belief
obs=hear-tiger-left
action=listen

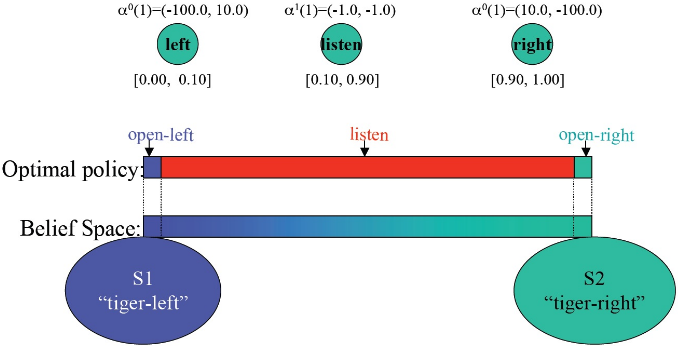
老虎问题 (con't)



b_1
b_0
$b_1(s_i) = \frac{P(o \mid s_i, a) \sum_{s_j \in S} P(s_i \mid s_j, a) b_0(s_j)}{P(o \mid a, b)}$
Belief vector
S1
"tiger-left"
S2
"tiger-right"
Belief
obs=growl-left
action=listen

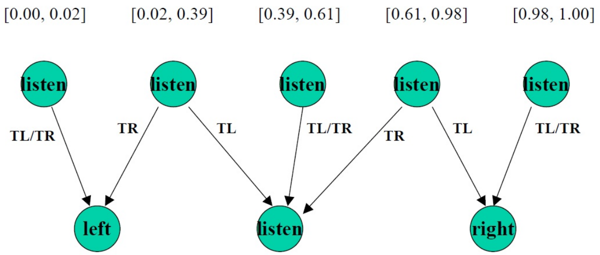
老虎问题 (con't)

- Optimal Policy for t=1



老虎问题 (con’t)

- For t=2



小结

- MDP 刻画了行动的不确定性，考虑概率和效用结合的最大期望效用
- MDP 模型为四元组 (S, A, T, R)
- MDP 可以用值迭代算法计算最优策略，计算复杂性为 P-complete
- POMDP 在 MDP 基础上刻画观察的不确定性，同样计算最大期望效用
- POMDP 模型为六元组 (S, A, T, R, Ω, O)
- POMDP 可以看做一个基于无穷信念状态的 MDP
- POMDP 的策略可以通过策略树表示
- POMDP 的最优策略的计算复杂性为 PSPACE-complete