reinforcement learning : intro + MDP

terminologies

actions, probabilities, costs (of actions), utilities (of results)

decision-theoretic agent

an agent is rational exactly when it chooses the action with the maximum expected utility taken over all results of actions

utility-theoretic agent

Expected Utility:

$$\boxed{EU(A \mid E) = \sum_i P(Result_i(A) \mid Do(A), E)\, U(Result_i(A))}$$

The **principle of maximum expected utility (MEU)** says that a rational agent should choose an action that maximizes $EU(A \mid E)$.

axioms of utility

a lottery:

Example:
Lottery L with two outcomes, C_1 and C_2:

$$L = [p, C_1; 1-p, C_2]$$

Preference between lotteries:

$L_1 \succ L_2$	The agent prefers L_1 over L_2
$L_1 \sim L_2$	The agent is indifferent between L_1 and L_2
$L_1 \succsim L_2$	The agent prefers L_1 or is indifferent between L_1 and L_2

the axioms of utility

- **Orderability**
  $(A \succ B) \vee (B \succ A) \vee (A \sim B)$

- **Transitivity**
  $(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$

- **Continuity**
  $A \succ B \succ C \Rightarrow \exists p[p, A; 1-p, C] \sim B$

- **Substitutability**
  $A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$

- **Monotonicity**
  $A \succ B \Rightarrow (p > q \Leftrightarrow [p, A; 1-p, B] \succ [q, A; 1-q, B])$

- **Decomposability**
  $[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$

utility functions and axioms

- **Utility Principle** If an agent's preferences obey the axioms, then there exists a function $U : S \mapsto R$ with
  $U(A) > U(B) \Leftrightarrow A \succ B$
  $U(A) = U(B) \Leftrightarrow A \sim B$

- **Expected Utility of a Lottery:**
  $U([p_1, S_1; \ldots; p_n, S_n]) = \sum_i p_i \, U(S_i)$
  $\rightarrow$ Since the outcome of a nondeterministic action is a lottery, an agent can act rationally only by following the Maximum Expected Utility (MEU) principle.

assessign utilities

- "Best possible prize" $U(S) = u_{max} = 1$
- "Worst catastrophe" $U(S) = u_{min} = 0$

Given a utility scale between u_{min} and u_{max} we can asses the utility of any particular outcome S by asking the agent to choose between S and a standard lottery $[p, u_{max}; 1-p, u_{min}]$. We adjust p until they are equally preferred.
Then, p is the utility of S. This is done for each outcome S to determine $U(S)$.

human judgement and irrationality

sequential decision problems

deterministic ~: all actions always lead to the next selected direction

stochastic ~: each action achieves the intended effect with a probability

- an MDP:
  - **Set of states** S
  - **Set of actions** A
  - **Transition model** $P(s' \mid s, a)$, with $s, s' \in S$ and $a \in A$
  - **Reward function** $R(s)$, with $s \in S$ $\boxed{\text{ALTERNATIVE is to use } R(s,a,s')}$
- transition model, policy, optimal policy
- performance: sum of rewards for the states visited
- finite / infinite horizon

  finite horizon: 有限时间范围，到达一个 s_N 之后的utility就不再重要

  - Finite horizon: $U_h([s_0, s_1, \ldots, s_{N+k}]) = U_h([s_0, s_1, \ldots, s_N])$ for all $k > 0$.

  optimal policy depends on **current state** & **remaining steps to go**

  (nonstationary)

  for infinite horizon:

  optimal policy only depends on current state (stationary)
- assign utilities:
  - **Additive rewards**:
    $U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$
  - **Discounted rewards**:
    $U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$
  - The term $\gamma \in [0, 1[$ is called the **discount factor**.
  - With **discounted rewards** the **utility of an infinite state sequence is always finite**. The **discount factor expresses** that **future rewards have less value than current rewards**.

for finite horizon, additive rewards are reasonable;

for infinite horizon, discount -> reward value converges to a finite value

- the utility of a state depends on the utility of the state sequences that follow it

$$U^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s\right]$$

Note:

R(s) is the short-term reward for being in s

U(s) is the long-term total reward from s onwards

- choosing actions using MEU

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)\, U(s')$$

Bellman-equation:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} P(s' \mid s, a)\, U(s')$$

utility of a state =

immediate reward

```
+ expected discounted utility of the next state
```

应用bellman eauation的例子：

Bellman-Equation: Example

- In our 4×3 world the equation for the state (1,1) is

$$
\begin{aligned}
U(1,1) = -0.04 + \gamma \max\{ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && (Up)\\
& 0.9U(1,1) + 0.1U(1,2), && (Left)\\
& 0.9U(1,1) + 0.1U(2,1), && (Down)\\
& 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && (Right)\\
= -0.04 + \gamma \max\{ & 0.8 \cdot 0.762 + 0.1 \cdot 0.655 + 0.1 \cdot 0.705, && (Up)\\
& 0.9 \cdot 0.705 + 0.1 \cdot 0.762, && (Left)\\
& 0.9 \cdot 0.705 + 0.1 \cdot 0.655, && (Down)\\
& 0.8 \cdot 0.655 + 0.1 \cdot 0.762 + 0.1 \cdot 0.705\} && (Right)\\
= -0.04 + 1.0\,(& 0.6096 + 0.0655 + 0.0705), && (Up)\\
= -0.04 + 0.7456 = 0.7056 &&&
\end{aligned}
$$

for these utilities

3	0.812	0.868	0.918	+1
2	0.762		0.660	–1
1	0.705	0.655	0.611	0.388
	1	2	3	4

- Up is the optimal action in (1,1).

注意stochastic agent的特点：预期方向和预期方向的正交方向！

- q-equation: generalization of the Bellman-equation

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s' \mid s, a)\, U(s')$$

$$Q(s,a) = R(s) + \gamma \sum_{s'} P(s' \mid s, a)\, \max_{a'} Q(s', a')$$

is the corresponding equation for the **Q-function -the action utility function** - the utility after taking action a in s

$$U(s) = \max_{a} Q(s,a)$$

---

value iteration: calculating optimal policies

histories, separable,

A utility function on histories U_h is **separable** iff there exists a function f such that

$$U_h([s_0, s_1, \ldots, s_n]) = f(s_0, U_h([s_1, \ldots, s_n]))$$

The simplest form is an additive reward function R:

$$U_h([s_0, s_1, \ldots, s_n]) = R(s_0) + U_h([s_1, \ldots, s_n]))$$

-> 递归迭代（如果计算过程有环，history的长度可能是无限长；而无限长也可能最终值是收敛的converge）

非线性（max运算）

算法：（多次迭代直到收敛）

```
function VALUE-ITERATION(mdp, ε) returns a utility function
  inputs: mdp, an MDP with states S, actions A(s), transition model P(s′ | s, a),
              rewards R(s, a, s′), discount γ
          ε, the maximum error allowed in the utility of any state
  local variables: U, U′, vectors of utilities for states in S, initially zero
                   δ, the maximum relative change in the utility of any state

  repeat
     U ← U′; δ ← 0
     for each state s in S do
        U′[s] ← max_{a ∈ A(s)} Q-VALUE(mdp, s, a, U)
        if |U′[s] − U[s]| > δ then δ ← |U′[s] − U[s]|
  until δ ≤ ε(1 − γ)/γ
  return U
```

为什么这样能收敛？可证明：

- It can be shown that **value iteration converges** and that

  if $\|U_{t+1} - U_t\| < \epsilon(1-\gamma)/\gamma$ then $\|U_{t+1} - U\| < \epsilon$

  if $\|U_t - U\| < \epsilon$ then $\|U^{\pi_t} - U\| < 2\epsilon\gamma/(1-\gamma)$

---

policy iteration

- policy evaluation

- **Policy evaluation**: given a policy π_t, calculate $U_t = U^{\pi_t}$, the utility of each state if π_t were executed.

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$

- policy improvement

$$\pi_{t+1}(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)\, U_t(s')$$

算法伪代码：

function POLICY-ITERATION(mdp) **returns** a policy
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$
  local variables: U, a vector of utilities for states in S, initially zero
    π, a policy vector indexed by state, initially random

  repeat
    $U \leftarrow$ POLICY-EVALUATION(π, U, mdp)
    $unchanged? \leftarrow$ true
    for each state s **in** S **do**
      $a^* \leftarrow \underset{a \in A(s)}{\operatorname{argmax}}$ Q-VALUE(mdp, s, a, U)
      if Q-VALUE(mdp, s, a^*, U) $>$ Q-VALUE($mdp, s, \pi[s], U$) **then**
        $\pi[s] \leftarrow a^*$; $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

多次迭代对每一个state都找到最佳action（每个state的最佳action都left unchanged）

---

value iteration 和 policy iteration 两个算法的区别

policy iteration算法实际上是value iteration算法的一部分，value iteration直接计算value