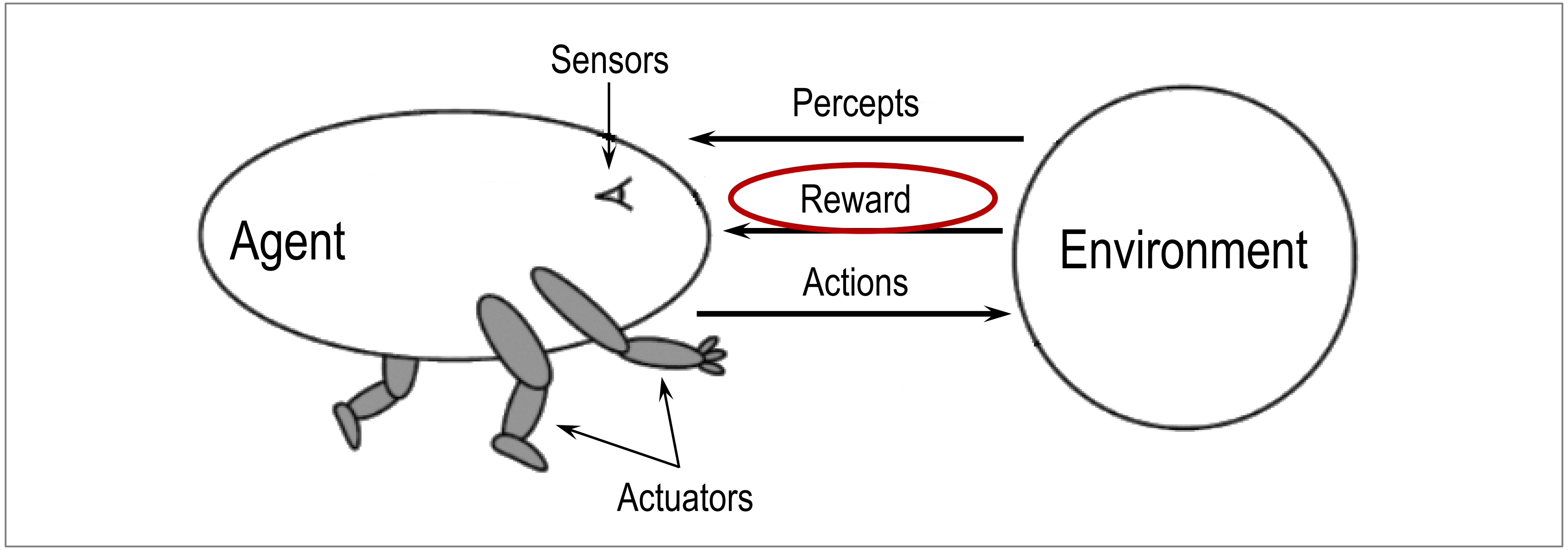
Decision-theoretic Planning

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

What is Decision-theoretic Planning 什么是决策理论规划

- Classic planning is to find a plan to achieve its goals with lowest cost.
  经典规划是寻找一个以最小代价到达其目标的计划。
- Decision-theoretic Planning is to find a plan to achieve its goals with maximum expected utility (MEU).
  决策理论规划是寻找一个以最大期望效用 (MEU) 到达其目标的规划。

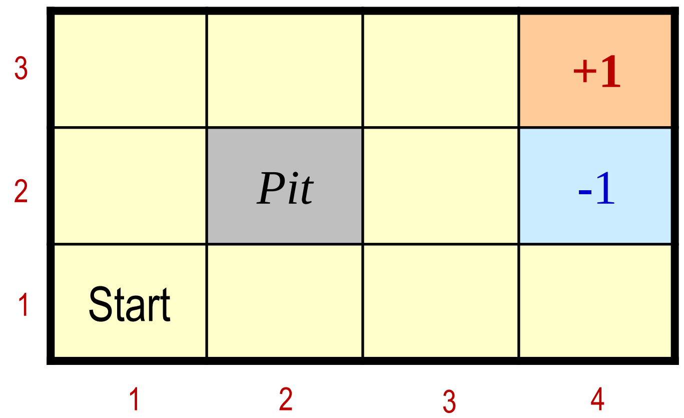


Example: Grid World 方格世界

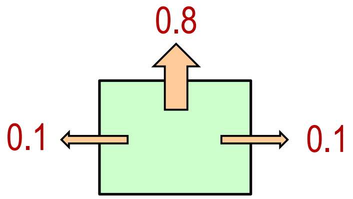
- Agent lives in a grid, walls block agent's path. Stochastic movement.
  智能体在格子中，围墙挡住了智能体的去路。随机移动。

- *Transition model*: 转换模型
  - probability 0.8: agent moves up;
    概率0.8：智能体上移；
  - probability 0.1: agent moves right or left;
    概率0.1：智能体左移、右移；
  - no movement: if a wall in the direction;
    不移：若前方是堵墙；
  - reward +1 and –1: two terminal states;
    回报+1和−1：两个终点状态；
  - reward –0.04: other no-terminal states.
    回报−0.04：其它非终点状态。

- Goal: maximize sum of rewards. 目标：回报值最大化。



(a)



(b)

How to Formulize and Solve 如何形式化与求解

- How to formalize the problems of Decision-theoretic Planning?
  如何对决策理论规划问题进行形式化？
  - Markov Decision Process (MDP)
    马尔科夫决策过程 (MDP)
- How to solve the problems of Markov Decision Process?
  如何对马尔科夫决策过程进行求解？
  - Dynamic Programming
    动态规划

Contents

Markov Decision Process (MDP) 马柯夫决策过程 (MDP)

- It is a *discrete time stochastic control process*, means action outcomes depend only on the current state.

  是一种离散时间随机控制过程，意味着动作结果仅仅依赖于当前状态。

- A Markov Decision Process (MDP) is a 5-tuple (S, A, T, R, γ), where

  一个马柯夫决策过程是一个5元组 (S, A, P, R, γ)，其中

  - a set of states, $s \in S$ 一个状态集，$s \in S$
  - a set actions, $a \in A$ 一个动作集，$a \in A$
  - a transition model, $T(s, a, s')$ 一个迁移模型，$T(s, a, s')$

    Probability that a from s leads to s', i.e., $P(s' | s, a)$

    从s导出s'的概率，即：$P(s' | s, a)$
  - a reward function, $R(s, a, s')$ 一个回报函数，$R(s, a, s')$
  - discount, $\gamma \in [0, 1]$ 衰减，$\gamma \in [0, 1]$

Core Problem 核心问题

- The core problem of classical planning：经典规划的核心问题
  - agent is in a *deterministic environment,*
    智能体是在一个确定性的环境，
  - solving the problem is to find a plan to achieve its goal.
    求解该问题是找到到一个达其目标的计划。
- The core problem of Markov Decision Process (MDP)：马尔科夫决策过程的核心问题
  - agent is in a *discrete time stochastic environment,*
    智能体处于一个离散时间随机环境，
  - solving the problem is to find a policy to control his process.
    求解该问题是找到一个控制其过程的策略。

Finding policy is the core problem to solve MDPs

Core Problem 核心问题

- Given a MDP(S, A, T, R, γ), a policy is a computable function π that outputs for each state s an action a.

  给定一个MDP(S, A, T, R, γ)，一个策略是一个计算函数π，它对每个状态s生成一个动作a.

  - A *deterministic policy* π is defined as: 一个确定性策略被定义为：

$$\pi : S \rightarrow A$$

  - A *stochastic policy* π can also be defined as: 一个随机策略也可以被定义为：

$$\pi : S \times A \rightarrow [0,1]$$

where $\pi(s, a) \geq 0$ and $\sum_a \pi(s, a) = 1$

- Goal is to choose a policy π that will maximize some cumulative function of the random rewards.

  目标是选择一个策略π，使随机回报值的一些累积函数最大化。

Utilities and Optimal Policies 效用和优化策略

- In sequential decision problems, preferences are expressed between sequences of states.
  在顺序决策问题中，偏好由状态顺序之间的顺序来表示。
- Usually use an additive utility functions:
  通常采用一个累加效用函数：

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots = \sum_i R(s_i)$$

- Utility of a *state* (a.k.a. its value) is defined to be:
  一个状态（亦称其值）的效用被定义为：

  $U(s_i)$ = expected sum of rewards until termination assuming optimal actions.

  假设最佳动作结束之前的预期回报值的总和
- Two optimal policies: Value Iteration and Policy Iteration.
  两个优化策略：值迭代和策略迭代。

1) Value Iteration 值迭代

- Basic idea: 基本思想
  - calculate the utility of each state, and then use the state utilities to select an optimal action in each state.
    计算每个状态的效用，然后使用该状态效用在每个状态中选择一个最佳动作。
  - π function is not used; instead the value of π is calculated within $U(s)$.
    不使用π函数；而π值在$U(s)$中计算。
- Bellman equation for utilities:
  贝尔曼效用等式:

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

Bellman equation is the basis of value iteration algorithm.

1) Value Iteration 值迭代

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, 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 change in the utility of any state in an iteration

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s **in** S **do**
      $\boxed{U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']}$
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$
  return U

The value iteration algorithm for calculating utilities of states.

计算状态效用的值迭代算法

2) Policy Iteration 策略迭代

- Basic idea: alternate the two phases.

  基本思想：交替执行如下两个阶段：

  - Policy evaluation: 策略迭代

    given a policy π_i, calculate utility U_i of each state if π_i were to be executed.

    给定一个策略π_i，如果π_i被执行的话，计算每个状态的效用U_i。

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

  - Policy improvement: 策略改善

    calculate a new MEU (maximum expected utility) policy π_{i+1}, using one-step look-ahead based on U_i.

    使用基于U_i的提前看一步法，计算一个新的MEU（最大期待效用）策略π_{i+1}。

$$\pi^*(s) = \gamma \operatorname*{argmax}_{\alpha \in A(s)} \sum_{s'} P(s' \mid s, a) U(s')$$

2) Policy Iteration 策略迭代

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**
      if $\boxed{\max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s'] > \sum_{s'} P(s' \mid s, \pi[s])\ U[s']}$ **then do**
        $\boxed{\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\ U[s']}$
        $unchanged? \leftarrow$ false
  until $unchanged?$
  return π

The policy iteration algorithm for calculating an optimal policy.

计算最佳策略的值迭代算法

Thank you for your attention!

AI