Lecture5 Markov Decision Processes

- Problem — Many applications are non-deterministic! Go to next states with a probability
- Hidden Factors
  - No need to consider all of them
  - Too complicated to consider (Too Many)
  - Not everything can be clearly understood
- Applications
  - Robotics — Decide where to move, but actuators can fail etc
  - Resource Allocation — Decide what to produce,don't know customer demand
  - Agriculture — Decide what to plant, don't know weather
- Markov Process
  - Markov — Given the present state, the future and the past are independent
  - Type
    - Markov Reward Process (No Control)

$$P(\overbrace{s_{t+1}}^{\text{future}} \mid \overbrace{s_t}^{\text{current}}, \overbrace{s_{t-1}, \dots, s_0}^{\text{past}}) = P(s_{t+1} \mid s_t)$$

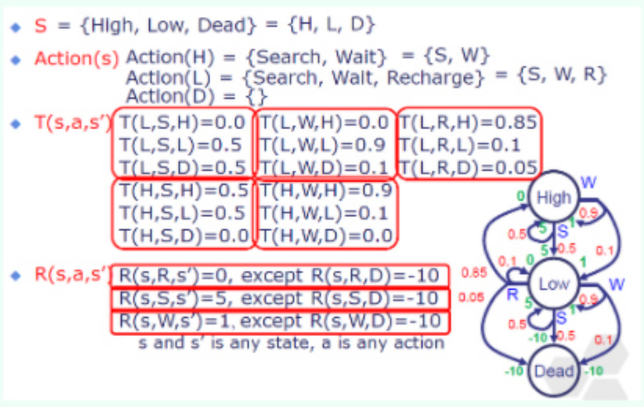
where t is the time

    - Markov Decision Process (Control)

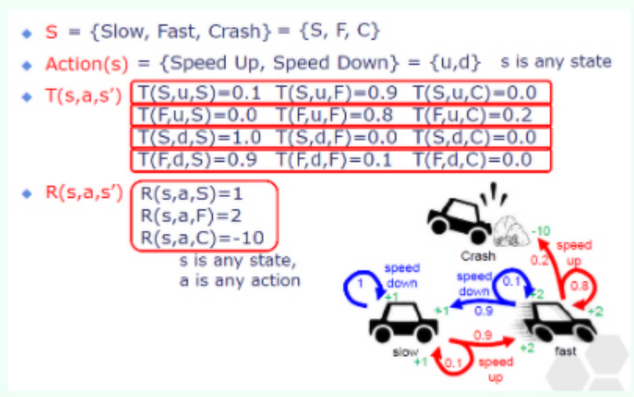
$$P(\overbrace{s_{t+1}}^{\text{future}} \mid \overbrace{a_t, s_t}^{\text{current}}, \overbrace{s_{t-1}, \dots, s_0}^{\text{past}}) = P(s_{t+1} \mid a_t, s_t)$$

where t is the time

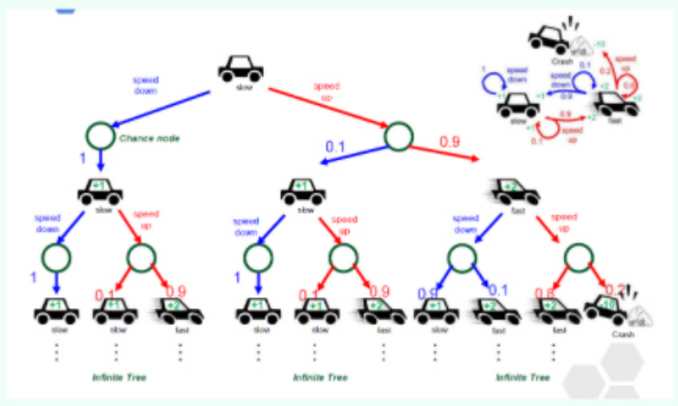
- Markov Decision Process
  - S State & S' New State After Action
  - Action(s)
  - T(s, a, s') — Transition Function P(s' | s, a)
  - R(s, a, s') — Reward Function E(e | s, a, s')
- Recycling Robot



- Example
  - Car Driving



- Tree Representation



- Reward Estimation
  - Problem: Tree can be extended forever and Actions may have infinite rewards
  - Time-Limited Values T — Limit the maximum steps in a episode and end
  - Discount Factor
    - Rewards decay exponentially

$$V(s_t) = \underbrace{R_{t+1}}_{\text{Next}} + \gamma \underbrace{R_{t+2}}_{\text{Next Next}} + \gamma^2 \underbrace{R_{t+3}}_{\text{Next Next Next}} + \gamma^3 \underbrace{R_{t+4}}_{\text{Next Next Next Next}} + \cdots$$

($V(s_t)$: now)

    - γ^∞ = 0 means no influence to the result
    - Bellman Equation

$$v_\pi(s) = \sum_a \pi(s,a) \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma v_\pi(s')]$$

- Markov Decision Process Policy
  - Prediction — Given policy Π, estimate state value $v_\pi(s)$
  - Control — Estimate the optimal state value $v_{\pi^*}(s)$ — pi * implicitly obtained
- State Value Estimation
  - Value Iteration — 根据policy迭代计算V(s)从0-n，直到收敛converge
  - Policy Iteration
    - Calculate utilities for the fixed policy until converge
    - Policy Evaluation

$$V_{k+1}^{\pi_i}(s) = \sum_{s'} T(s, \pi_i(s), s')[R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')]$$

    - Update the policy
    - Policy Improvement

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')]$$

    - Loop until no update on policy
  - Comparison
    - Value Iteration
      - Update the value (and the policy implicitly)
      - Don't track the policy explicitly
    - Policy Iteration
      - Update utility in a fixed policy
      - After one policy is evaluated, a new policy is chosen
    - Both of them can provide the optimal solution for MDPs

Pollion_Wang