ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf Informatik Computer Science

Probabilistic Foundations of Artificial Intelligence

Probabilistic Planning

Prof. Andreas Krause
Learning and Adaptive Systems (las.ethz.ch)

Markov Decision Processes

- An MDP is specified by
  - A set of states $X = \{1, \ldots, n\}$ …
  - A set of actions $A = \{1, \ldots, m\}$
  - Transition probabilities

    $P(x' \mid x, a) = \text{Prob(Next state} = x' \mid \text{Action } a \text{ in state } x)$
  - A reward function $r(x, a)$

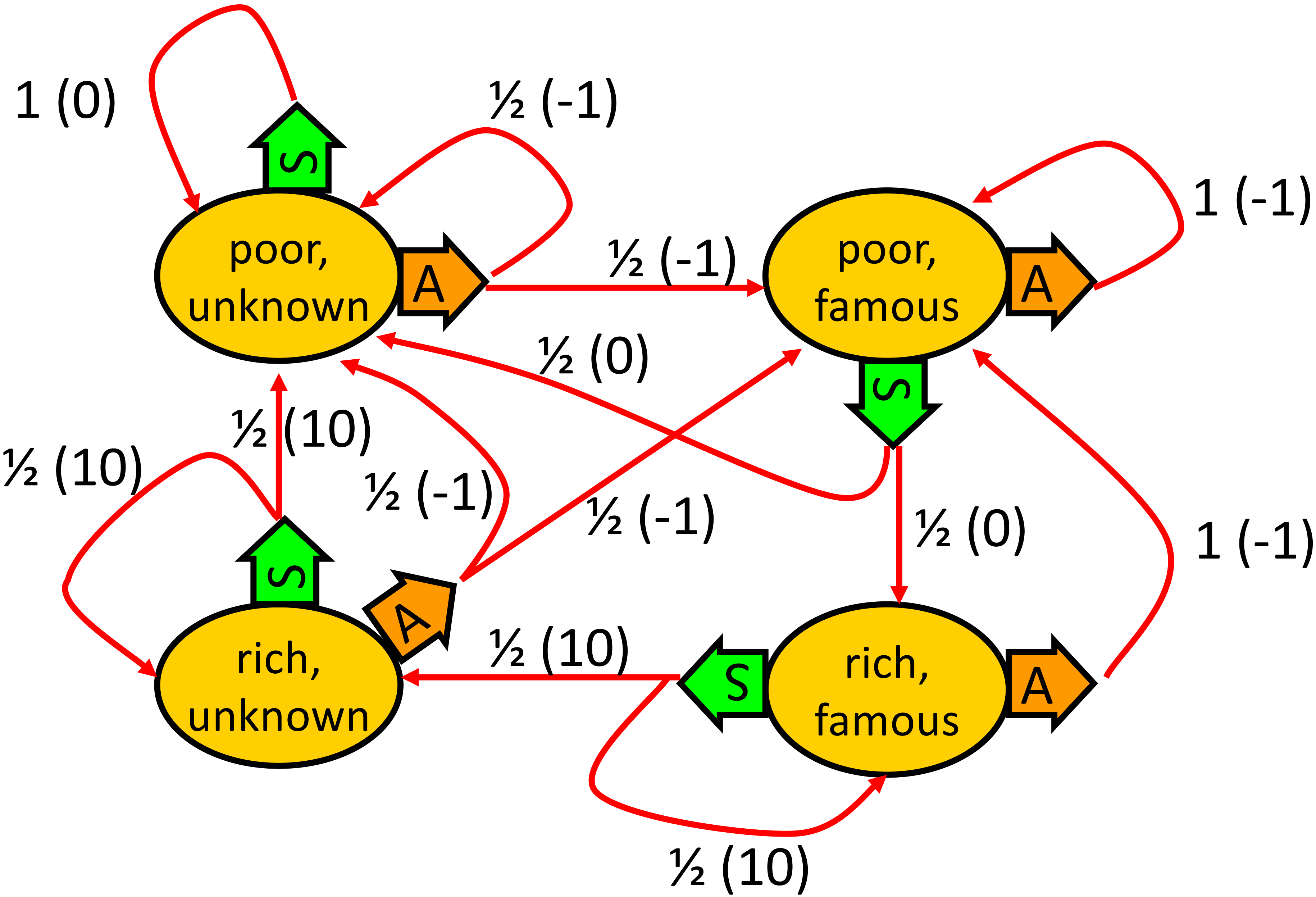
    Reward can be random with mean $r(x, a)$;

    Reward may depend on x only or (x, a, x') as well.
- For now assume r and P are known!
- Want to choose actions to maximize reward

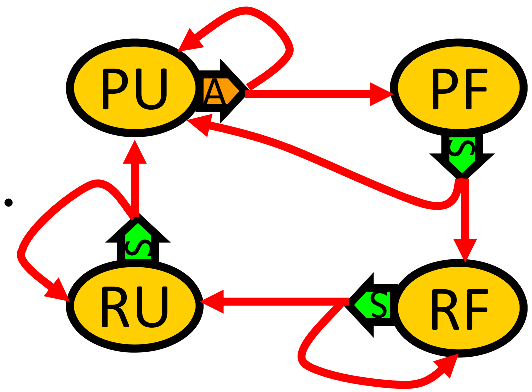
Applications of MDPs

- Robot action planning
- Elevator scheduling
- Manufactoring processes
- Network switching and routing
- AI in computer games
- Becoming rich and famous
- …

Becoming rich and famous



Planning in MDPs



- Deterministic policy $\pi: X \rightarrow A$
- Induces a **Markov chain**: $X_0, X_1, \ldots, X_t, \ldots$ with transition probabilities

$$P(X_{t+1}=x' \mid X_t=x) = P(x' \mid x, \pi(x))$$

- Expected value $J(\pi) = E[\ r(X_0,\pi(X_0)) + \gamma\, r(X_1,\pi(X_1)) + \gamma^2 r(X_2,\pi(X_2)) + \ldots\ ]$

Computing the value of a policy

For a fixed policy define value function

$$V^\pi(x) = J(\pi \mid X_0 = x) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(X_t, \pi(X_t)) \mid X_0 = x\Big]$$

Recursion:

$$V^\pi(x) = \mathbb{E}\Big[\gamma^0 r(X_0, \pi(X_0)) + \sum_{t=1}^{\infty} \gamma^t r(X_t, \pi(X_t)) \Big| X_0 = x\Big]$$

$$= r(x, \pi(x)) + \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^t r(X_t, \pi(X_t)) \mid X_0 = x\Big]$$

$$= r(x, \pi(x)) + \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^{t+1} r(X_{t+1}, \pi(X_{t+1})) \mid X_0 = x\Big]$$

$$= r(x, \pi(x)) + \gamma \sum_{x'} P(X_1 = x' \mid X_0 = x)\, \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(X_{t+1}, \pi(X_{t+1})) \mid X_1 = x'\Big]$$

$$\overset{\text{stat.}}{=} r(x, \pi(x)) + \gamma \sum_{x'} P(x' \mid x)\, \underbrace{\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(X_t, \pi(X_t)) \mid X_0 = x'\Big]}_{V^\pi(x')}$$

Solving for the value of a policy

$$V^\pi(x) = r(x,\pi(x)) + \gamma \sum_{x'} P(x' \mid x,\pi(x)) V^\pi(x')$$

$$V^\pi = \begin{bmatrix} V^\pi(1) \\ \vdots \\ V^\pi(n) \end{bmatrix}, \quad T^\pi = \begin{pmatrix} P(1 \mid 1,\pi(1)) & \cdots & P(n \mid 1,\pi(1)) \\ \vdots & & \\ P(1 \mid n,\pi(n)) & \cdots & P(n \mid n,\pi(n)) \end{pmatrix}, \quad r^\pi = \begin{bmatrix} r(1,\pi(1)) \\ \vdots \\ r(n,\pi(n)) \end{bmatrix}$$

$$V^\pi = r^\pi + \gamma T^\pi V^\pi \Rightarrow (I - \gamma T^\pi) V^\pi = r^\pi$$

$$\Rightarrow \boxed{V^\pi = (I - \gamma T^\pi)^{-1} r^\pi}$$

➔ **Can compute V^π exactly by solving linear system! ☺**

A simple algorithm

- For every policy π compute $J(\pi)$
- Pick $\pi^* = \text{argmax}\, J(\pi)$

Is this a good idea?

$\hookrightarrow$ m^n policies

Value functions and policies

Every value function induces a policy

Value function V^π

$$V^\pi(x) = r(x,\pi(x)) + \gamma \sum_{x'} P(x' \mid x,\pi(x))\, V^\pi(x')$$

Greedy policy w.r.t. V

$$\pi_V(x) = \arg\max_a r(x,a) + \gamma \sum_{x'} P(x' \mid x,a)\, V(x')$$

Every policy induces a value function

Theorem (Bellman):
Policy optimal ⇔ greedy w.r.t. its induced value function!

$$V^*(x) = \max_a \left[r(x,a) + \gamma \sum_{x'} P(x' \mid x, a)\, V^*(x') \right]$$

Policy iteration

- Start with an arbitrary (e.g., random) policy π
- Until converged do:

  Compute value function $V^{\pi}(x)$

  Compute greedy policy π_G w.r.t. V^{π}

  Set $\pi \leftarrow \pi_G$

- Guaranteed to
  - Monotonically improve $\forall x: V^{\pi_{t+1}}(x) \geq V^{\pi_t}(x)$
  - Converge to an optimal policy π^* in $O^*(n^2\, m\, /\, (1\text{-}\gamma))$ iterations! [Ye '10]

Alternative approach

- Recall (Bellman): For the optimal policy π^* it holds

$$\boxed{V^*(x) = \max_a r(x,a) + \gamma \sum_{x'} P(x' \mid x, a)\, V^*(x')}$$

- Compute V^* using fixed point / dynamic programming:

$V_t(x)$ = Max. expected reward when starting in state x and world ends in t time steps

$V_0(x) = \max_a r(x,a)$

$V_1(x) = \max_a r(x,a) + \gamma \sum_{x'} P(x' \mid x,a)\, V_0(x')$

$V_{t+1}(x) =$ ______ " ______ $V_t(x')$

Value iteration

- Initialize $V_0(x) = \max_a r(x, a)$
- For $t = 1$ to ∞

  For each x, a, let

  $$Q_t(x,a) = r(x,a) + \gamma \sum_{x'} P(x' \mid x,a)\, V_{t-1}(x')$$

  For each x let

  $$V_t(x) = \max_a Q_t(x,a)$$

  Break if $\|V_t - V_{t-1}\|_\infty = \max_x |V_t(x) - V_{t-1}(x)| \le \varepsilon'$

- Then choose greedy policy w.r.t. V_t

- **Guaranteed to converge to ε-optimal policy!**

Tradeoffs: Value vs Policy Iteration

- Policy iteration
  - Finds exact solution in polynomial # iterations!
  - Every iteration requires computing a value function
  - Complexity per iteration:
- Value iteration
  - Finds ε-optimal solution in polynomial # iterations
  - Complexity per iteration:
- In practice, which works better depends on application
- Can combine ideas of both algorithms

Recap: Ways for solving MDPs

- Policy iteration:
  - Start with random policy π
  - Compute exact value function V^{π} (matrix inversion)
  - Select greedy policy w.r.t. V^{π} and iterate
- Value iteration
  - Solve Bellman equation using dynamic programming
    $V_t(x) = \max_a r(x, a) + \gamma \sum_{x'} P(x' \mid x, a)\, V_{t-1}(x)$
- Linear programming