Introduction to Artificial Intelligence

Lecture 16 – Markov Decision Processes

CS/CNS/EE 154

Andreas Krause

Announcements

- Homework 3 out, due Wed Nov 24
- Code for project final released; due Dec 1

HMMs / Kalman Filters

$$\begin{tikzcd}
X_1 \arrow[r] \arrow[d] & X_2 \arrow[r] \arrow[d] & X_3 \arrow[r] \arrow[d] & X_4 \arrow[r] \arrow[d] & X_5 \arrow[r] \arrow[d] & X_6 \arrow[d] \\
Y_1 & Y_2 & Y_3 & Y_4 & Y_5 & Y_6
\end{tikzcd}$$

- $X_1,\ldots,X_T$: Unobserved (hidden) variables
- $Y_1,\ldots,Y_T$: Observations
- HMMs: X_i Multinomial, Y_i multinomial (or arbitrary)
- Kalman Filters: X_i, Y_i Gaussian distributions

Bayesian filtering

- Start with $P(X_1)$
- At time t
  - Assume we have $P(X_t \mid y_{1 \ldots t-1})$
  - Conditioning: $P(X_t \mid y_{1 \ldots t})$

$$\begin{tikzcd}
X_1 \arrow[r] \arrow[d] & X_2 \arrow[r] \arrow[d] & X_3 \arrow[r] \arrow[d] & X_4 \arrow[r] \arrow[d] & X_5 \arrow[r] \arrow[d] & X_6 \arrow[d] \\
Y_1 & Y_2 & Y_3 & Y_4 & Y_5 & Y_6
\end{tikzcd}$$

Have $P(X_1)$

want $P(X_1 \mid Y_1)$

$= \frac{1}{Z} P(Y_1 \mid X_1) \cdot P(X_1)$

Bayes' rule

$$P(X_t \mid y_{1:t}) = \frac{1}{Z} P(x_t \mid y_{1:t-1}) \cdot P(y_t \mid X_t)$$

  - Prediction: $P(X_{t+1} \mid y_{1 \ldots t})$

$$P(X_{t+1} \mid y_{1:t}) = \sum_{x_t} P(X_{t+1}, x_t \mid y_{1:t})$$

$$= \sum_{x_t} P(X_t \mid y_{1:t}) \underbrace{P(X_{t+1} \mid X_t, y_{1:t})}_{P(X_{t+1} \mid X_t)}$$

For k states, can do filtering in $O(k^2)$

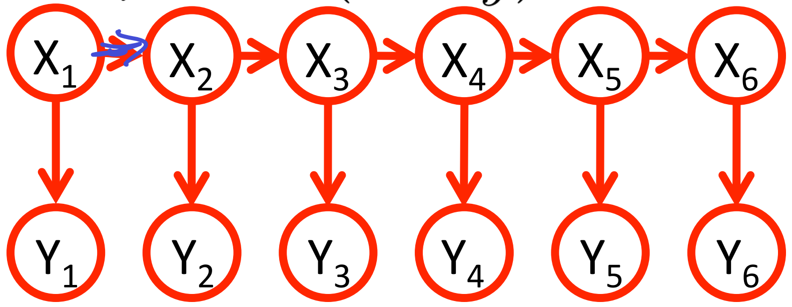
Kalman Filters (Gaussian HMMs)

- $X_1,\ldots,X_T$: Location of object being tracked $\in \mathbb{R}^d$
- $Y_1,\ldots,Y_T$: Observations $\in \mathbb{R}^{d'}$
- $P(X_1)$: Prior belief about location at time 1
- $P(X_{t+1}|X_t)$: "Motion model"
  - How do I expect my target to move in the environment?

$$\mathbf{X}_{t+1} = \mathbf{F}\mathbf{X}_t + \varepsilon_t \text{ where } \varepsilon_t \in \mathcal{N}(0, \Sigma_x)$$

- $P(Y_t \mid X_t)$: "Sensor model"
  - What do I observe if target is at location X_t?

$$\mathbf{Y}_t = \mathbf{H}\mathbf{X}_t + \eta_t \text{ where } \eta_t \in \mathcal{N}(0, \Sigma_y)$$



General Kalman update

- Transition model $P(\mathbf{x}_{t+1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t+1}; \mathbf{F}\mathbf{x}_t, \Sigma_x)$
- Sensor model $P(\mathbf{y}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t; \mathbf{H}\mathbf{x}_t, \Sigma_y)$

- Kalman Update:

$$\mu_{t+1} = \mathbf{F}\mu_t + \mathbf{K}_{t+1}(\mathbf{y}_{t+1} - \mathbf{H}\mathbf{F}\mu_t)$$

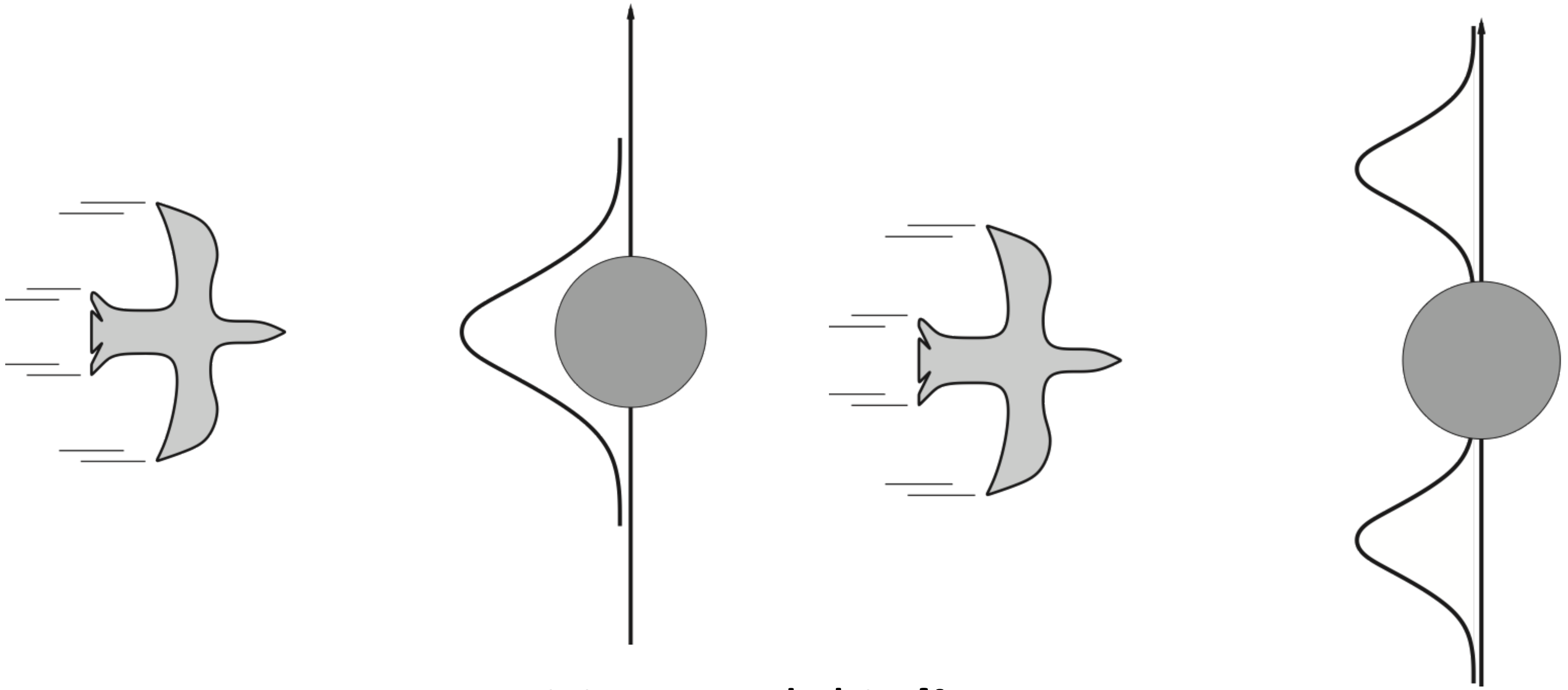
$$\Sigma_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1})(\mathbf{F}\Sigma_t\mathbf{F}^T + \Sigma_x)$$

- Kalman gain:

$$\mathbf{K}_{t+1} = (\mathbf{F}\Sigma_t\mathbf{F}^T + \Sigma_x)\mathbf{H}^T(\mathbf{H}(\mathbf{F}\Sigma_t\mathbf{F}^T + \Sigma_x)\mathbf{H}^T + \Sigma_y)^{-1}$$

- Can compute Σ_t and $\mathbf{K}_t$ offline

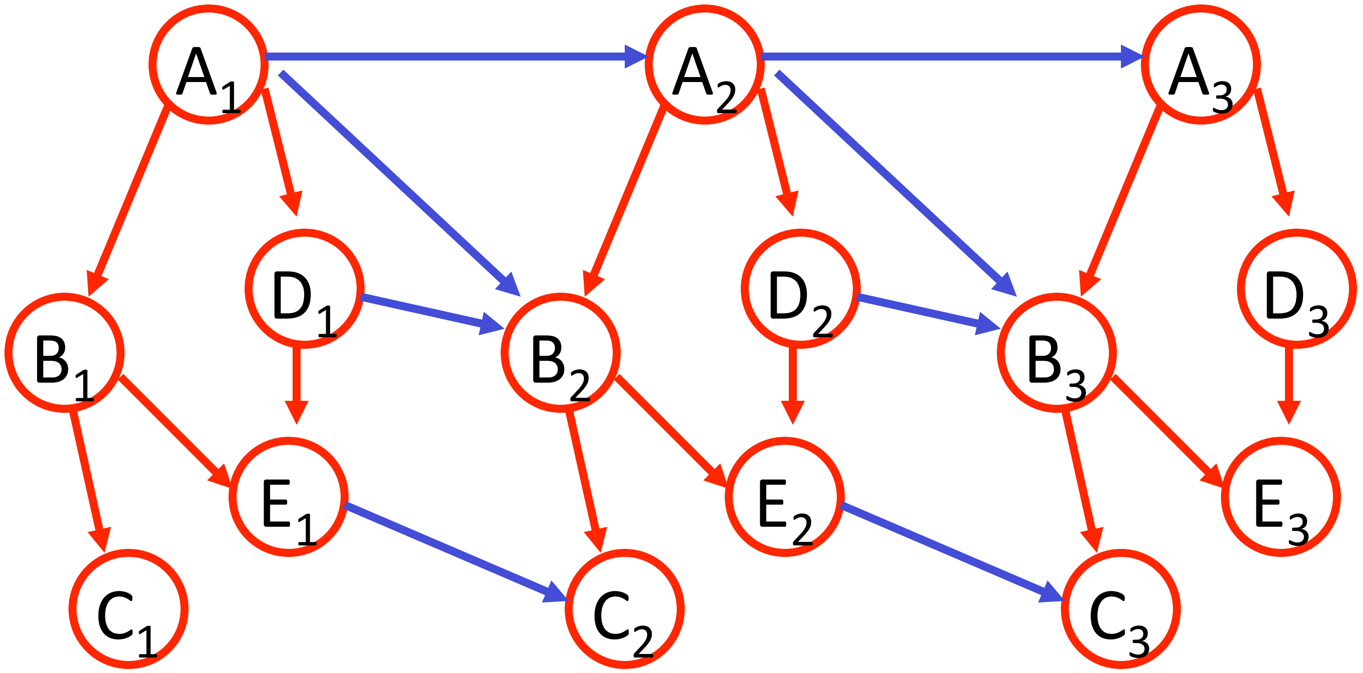
When KFs fail



- KFs assume transition model is **linear**
  - Implies that predictive distribution is Gaussian (unimodal)
- Need approximate inference to capture nonlinearities!

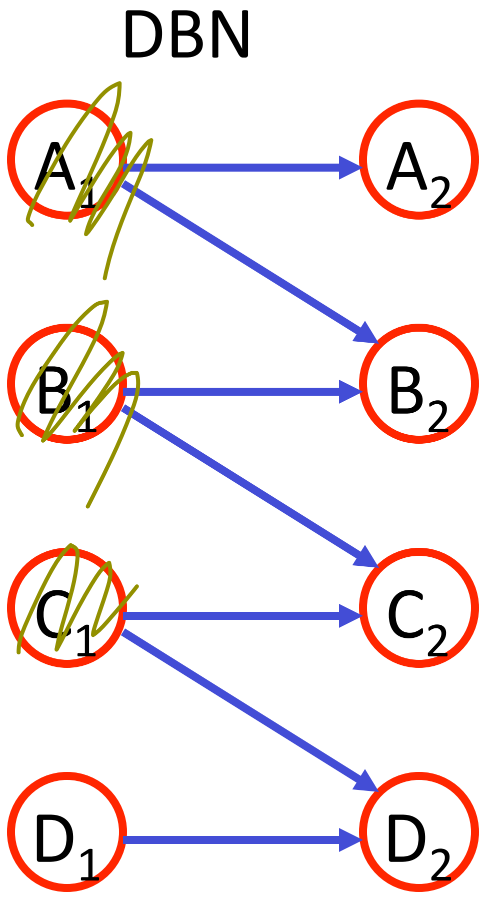
Dynamic Bayesian Networks

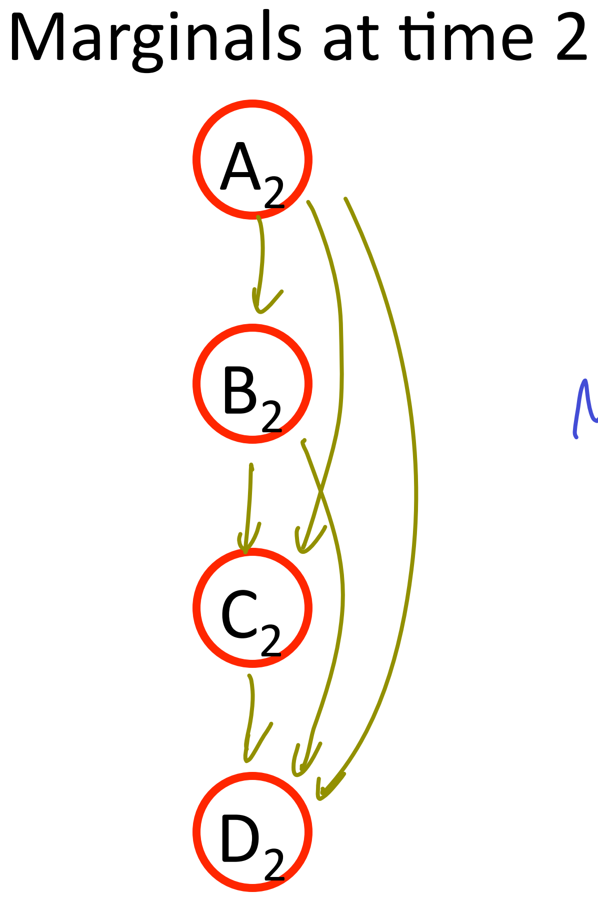
- At every timestep have a *Bayesian Network*



- Variables at each time step t called a slice $\mathbf{S_t}$
- "Temporal" edges connecting S_{t+1} with S_t

Inference in DBNs?





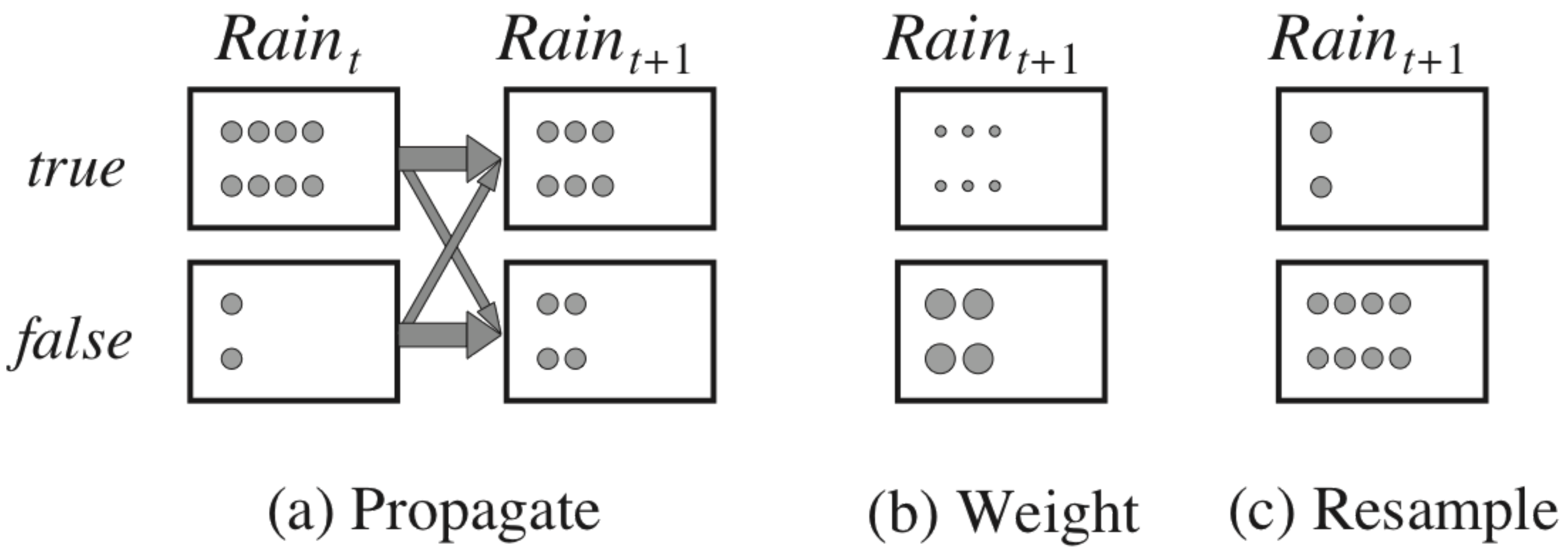
Need approximate inference!

Particle filtering

- Very useful approximate inference technique for dynamical models
  - Nonlinear Kalman filters
  - Dynamic Bayesian networks

- **Basic idea**: Approximate the posterior at each time by samples (particles), which are propagated and reweighted over time

Particle filtering example



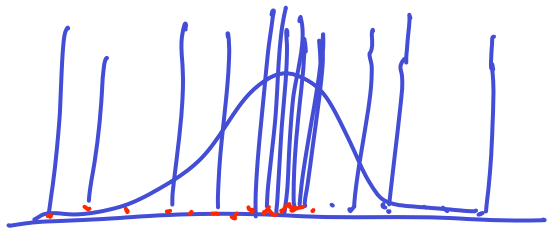
"No umbrella"

Representing distributions by particles

- True distribution (possibly continuous): $\underline{P(x)}$
- N i.i.d. samples: $\underline{x_1, \ldots, x_N}$

$$\delta_{x_i}(x) = \begin{cases} 1 & \text{if } x = x_i \\ 0 & \text{otw} \end{cases}$$

- Represent: $\underline{P(x)} \approx \frac{1}{N}\delta_{x_i}(x)$
- Get expectations: $\mathbb{E}_P[f(X)] \approx \frac{1}{N}\sum_i f(x_i)$
- E.g., mean: $\mathbb{E}_P[X] \approx \frac{1}{N}\sum_i x_i$



Particle filtering

- Suppose $$\underline{P(X_t \mid y_{1:t})} \approx \frac{1}{N}\sum_{i=1}^{N} \delta_{x_{i,t}}$$

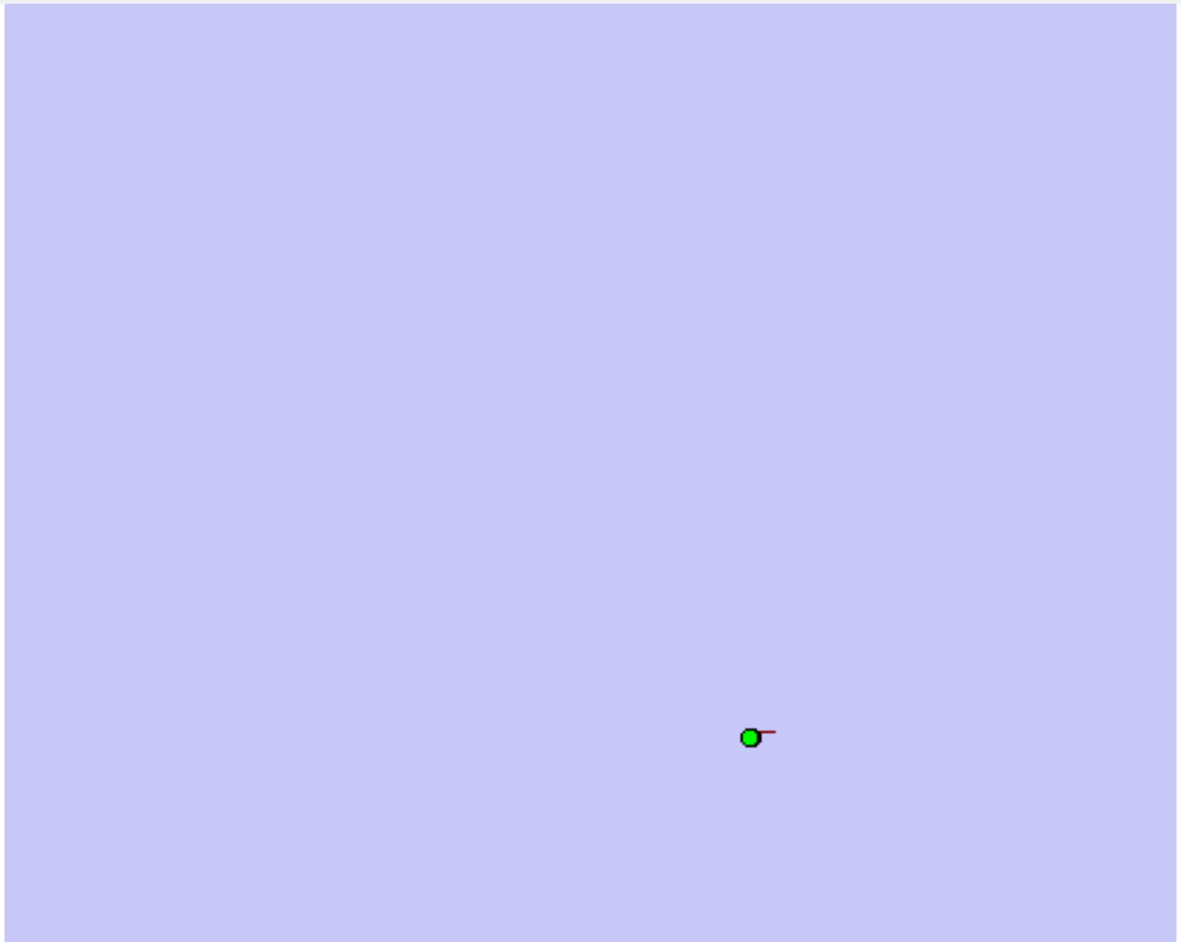
- For each particle: $$x_i' \sim P(X_{t+1} \mid x_{i,t})$$

$Z = \sum_{i=1}^{N} P(y_{t+1} \mid x_i')$

- Weigh particles: $$w_i = \frac{1}{Z} P(y_{t+1} \mid x_i')$$

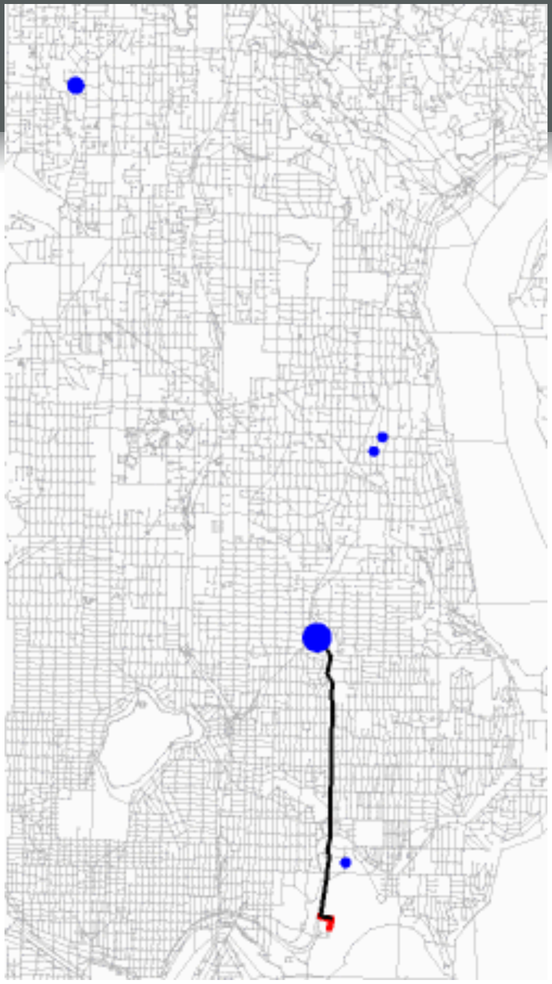
- Resample N particles $$x_{i,t+1} \sim \frac{1}{N}\sum_{i=1}^{N} w_i \delta_{x_i'}$$

Robot localization & mapping



D. Haehnel,
W. Burgard,
D. Fox, and
S. Thrun.
IROS-03.

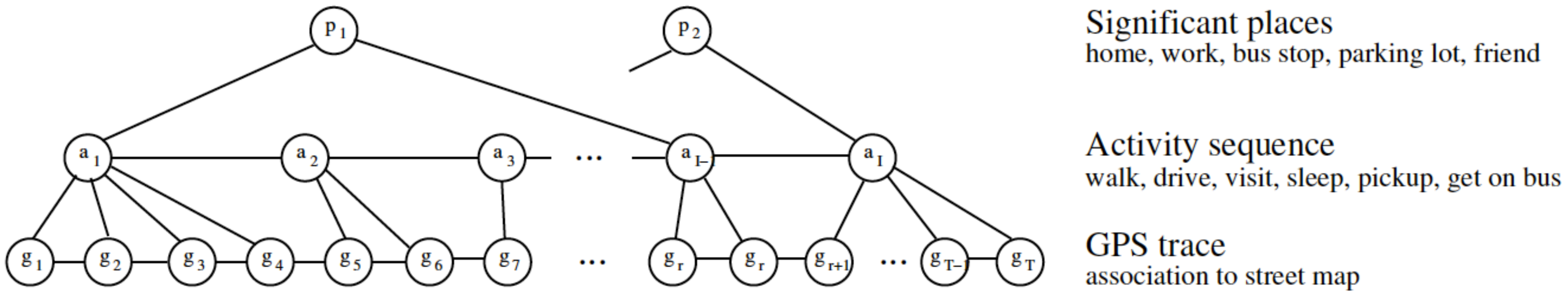
- Infer both location and map from noisy sensor data
- Particle filters



Activity recognition

L. Liao, D. Fox, and H. Kautz. *AAAI-04*

Predict "goals" from raw GPS data
"Hierarchical Dynamical Bayesian networks"



p₁
p₂
a₁
a₂
a₃
a_{I-1}
a_I
g₁
g₂
g₃
g₄
g₅
g₆
g₇
g_r
g_r
g_{r+1}
g_{T-1}
g_T
Significant places
home, work, bus stop, parking lot, friend
Activity sequence
walk, drive, visit, sleep, pickup, get on bus
GPS trace
association to street map

Summary

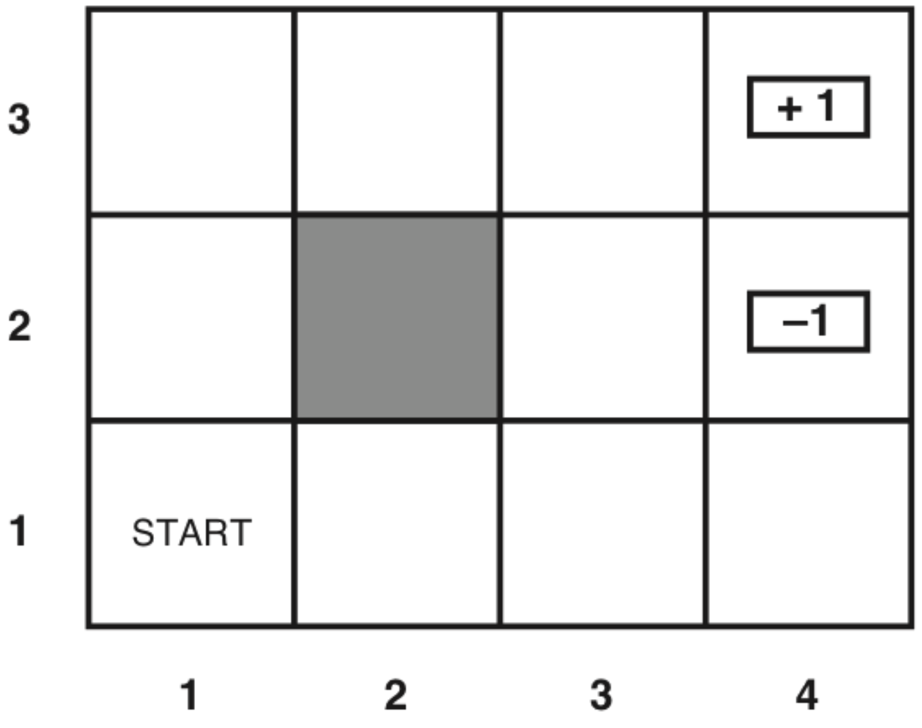
- Dynamical models
  - Multiple copies of static models, one per time step
- Examples:
  - HMM
  - Kalman Filter
  - Dynamic Bayesian networks
- Inference tasks
  - Filtering/prediction: Can do recursively!
  - Smoothing
  - MPE
- Particle filtering for approximate inference

Probabilistic planning

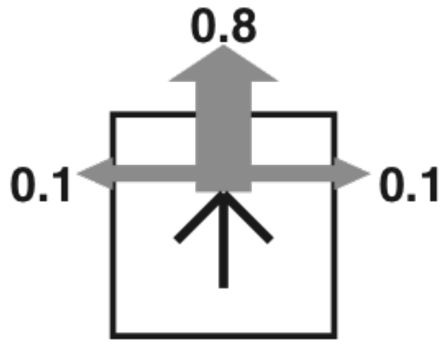
- So far: Probabilistic inference in dynamical models
  - E.g.: Tracking a robot based on noisy measurements

- Next: How should we control the robot to maximize reward?

Probabilistic planning

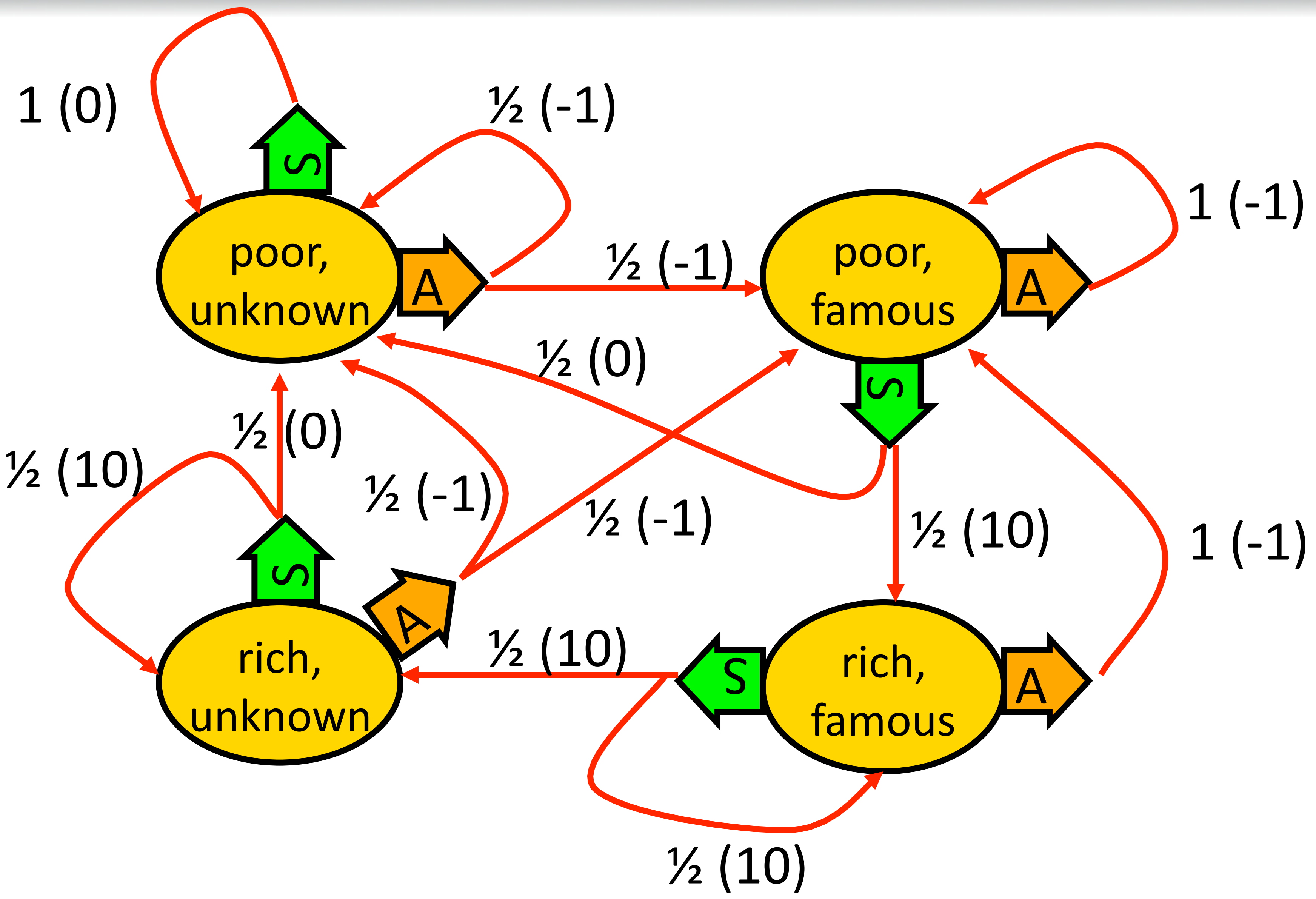


(a)



(b)

Becoming rich and famous



Markov Decision Processes

- An MDP has
  - A set of states $X = \{x_1,\ldots,x_n\}$ …
  - A set of actions $A = \{a_1,\ldots,a_m\}$
  - A reward function $r(x,a)$ [or random var. with mean $r(x,a)$]
  - Transition probabilities

    $P(x'|x,a)$ = Prob(Next state = x' | Action a in state x)

- For now assume r and P are known!

- Want to choose actions to maximize reward

Utility over time

- Finite horizon

$$\begin{tikzcd} S_1 \arrow[r, "a_1"] \arrow[d] & S_2 \arrow[r, "a_2"] \arrow[d] & S_3 \arrow[r] \arrow[d] & \cdots \arrow[r] & S_T \\ r_1 & r_2 & r_3 & \cdots & \end{tikzcd}$$

$$R_T = \sum_{t=1}^{T} r_t$$

- Discounted rewards

$$\begin{tikzcd} S_1 \arrow[r, "a_1"] \arrow[d] & S_2 \arrow[r, "a_2"] \arrow[d] & \cdots \\ r_1 & r & \end{tikzcd}$$

$$R_T = \sum_{t=0}^{\infty} \gamma^t r_t$$

$$\gamma \in (0,1)$$

In practice: $\gamma = 0.95$

Finite horizon MDP Decision model

- Reward R = 0
- Start in state x
- For t = 0 to T
    - Choose action a
    - Obtain reward R = R + r(x,a)
    - End up in state x’ according to P(x'|x,a)
    - Repeat with x ← x’

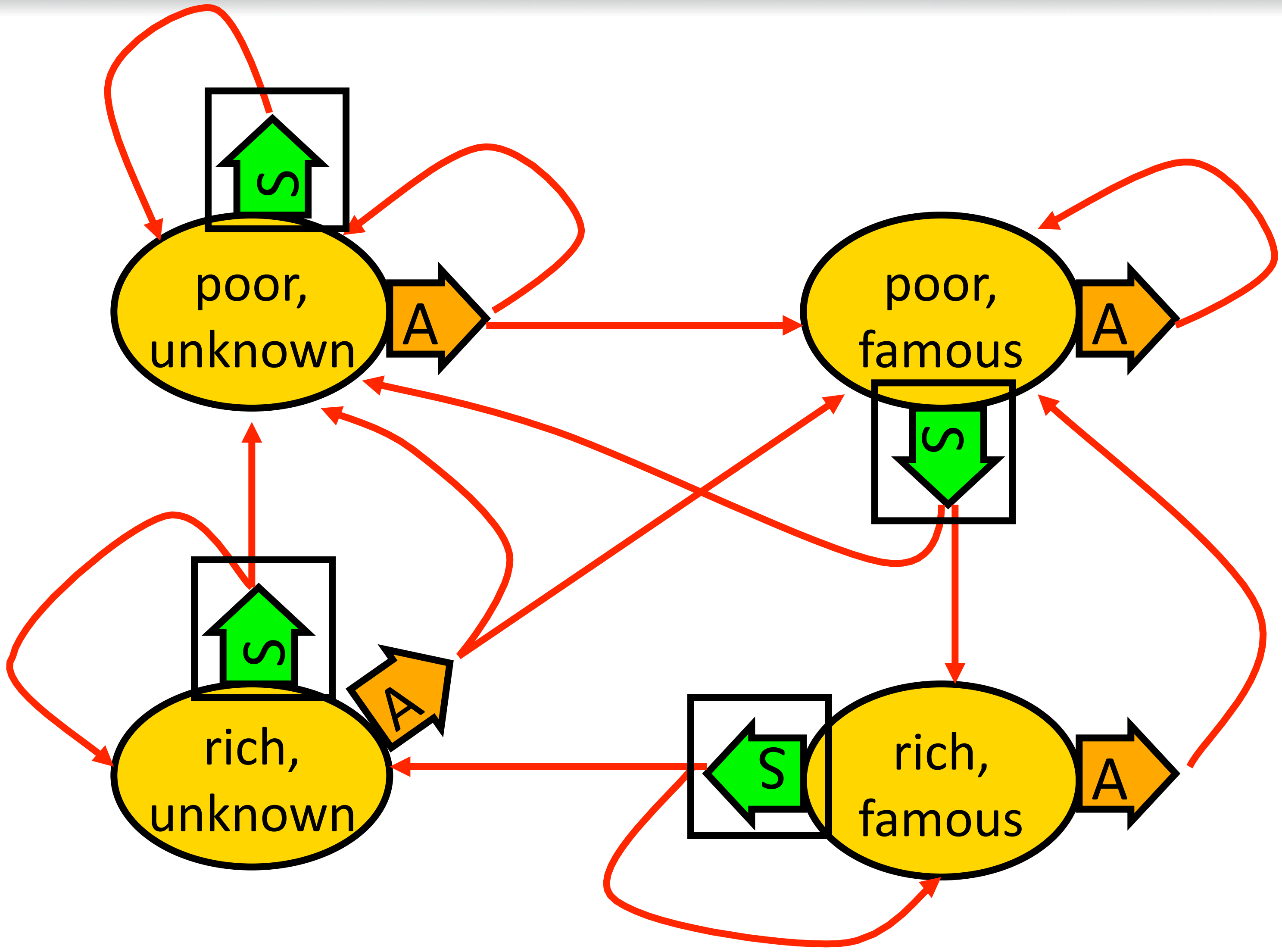
Discounted MDP Decision model

- Reward R = 0
- Start in state x
- For t = 0 to $\boxed{\infty}$
  - Choose action a
  - Obtain **discounted** reward R = R + $\boxed{\gamma^t}$ r(x,a)
  - End up in state x' according to P(x'|x,a)
  - Repeat with x ← x'

This lecture: Discounted rewards

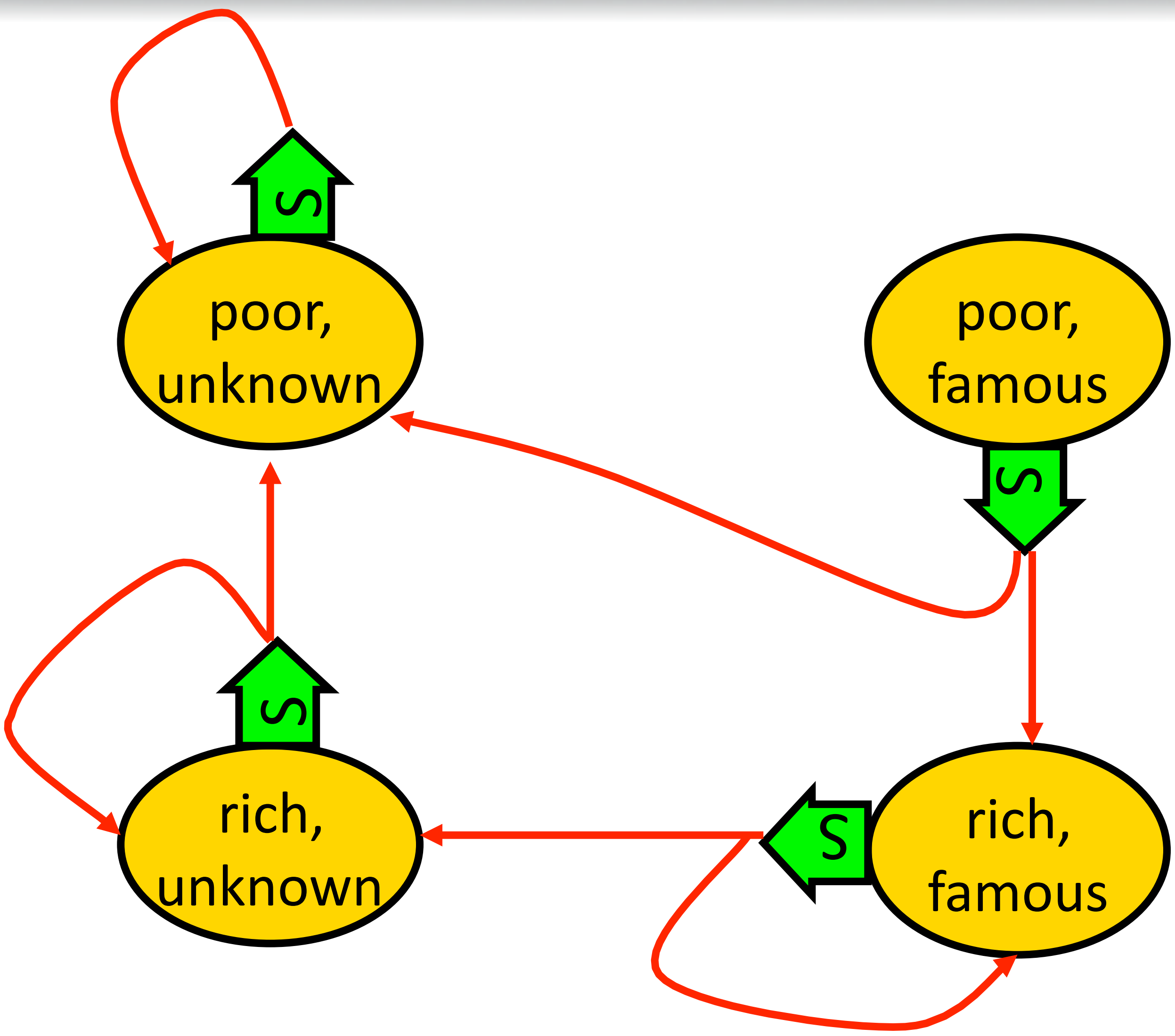
- Fixed probability (1-γ) of "obliteration" (inflation, running out of battery, …)

Policies

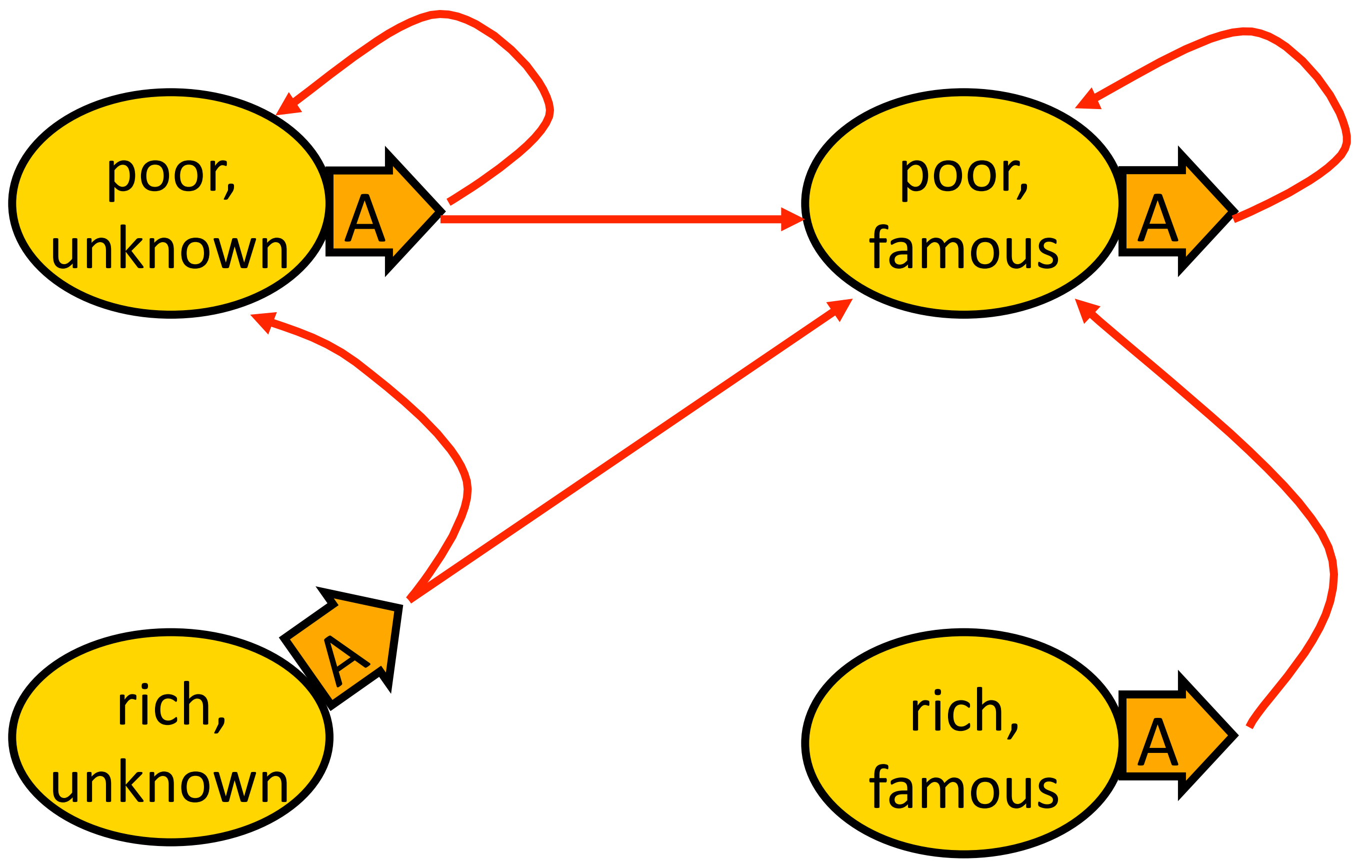


Policy: Pick one *fixed* action for each state

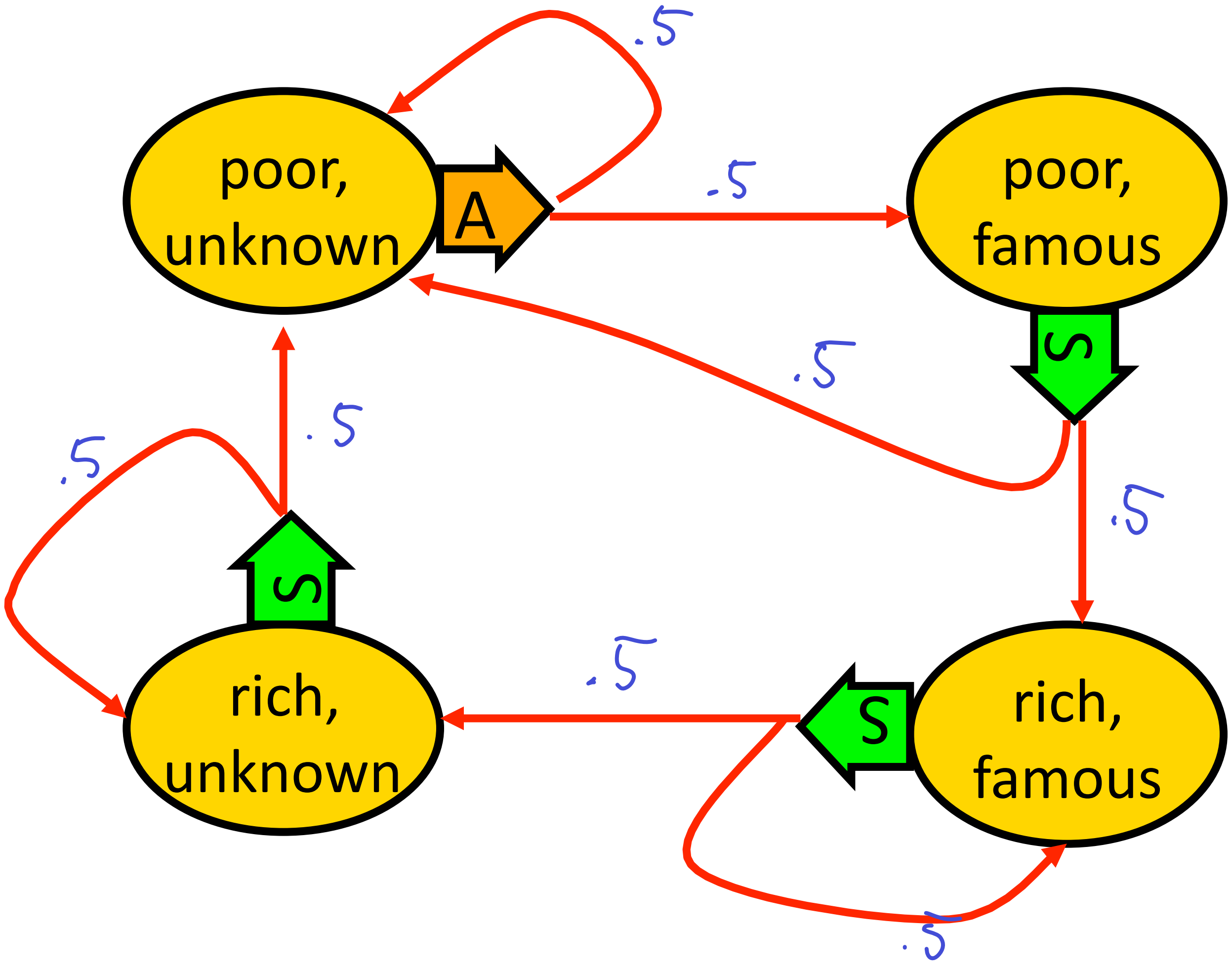
Policies: Always save?



Policies: Always advertise?

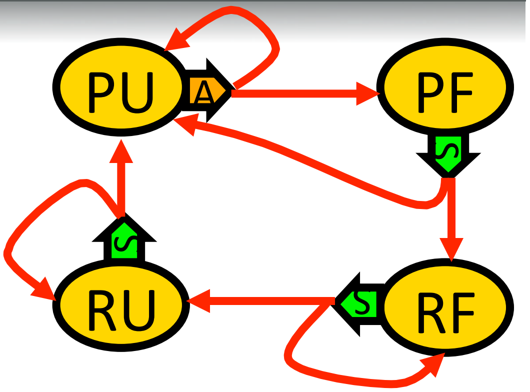


Policies: How about this one?



Planning in MDPs

- Deterministic policy $\pi: X \rightarrow A$
- Induces a **Markov chain**: $X_1, X_2, \ldots, X_t, \ldots$ with transition probabilities



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

- Expected value $J(\pi) = E[\ r(X_1,\pi(X_1)) + \gamma\, r(X_2,\pi(X_2)) + \gamma^2 r(X_3,\pi(X_3)) + \ldots\ ]$

Computing the value of a policy

- For fixed policy π and each state x, define **value function**

$V^{\pi}(x) = J(\pi \mid \text{start in state } x) = \underline{r(x,\pi(x))} + E[\sum_t \gamma^t\, r(X_t,\pi(X_t))]$

Recursion: $V^{\pi}(x) = r(x,\pi(x)) + \gamma\, E[\sum_t \gamma^{t-1}\, r(X_t,\pi(X_t))]$

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

and J(π) = $V^{\pi}(x_0)$

$\uparrow$ start state

$[V^{\pi}(1) \dots V^{\pi}(n)]^T$ $\quad [r(1,\pi(1)), \dots r(n,\pi(n))]^T$

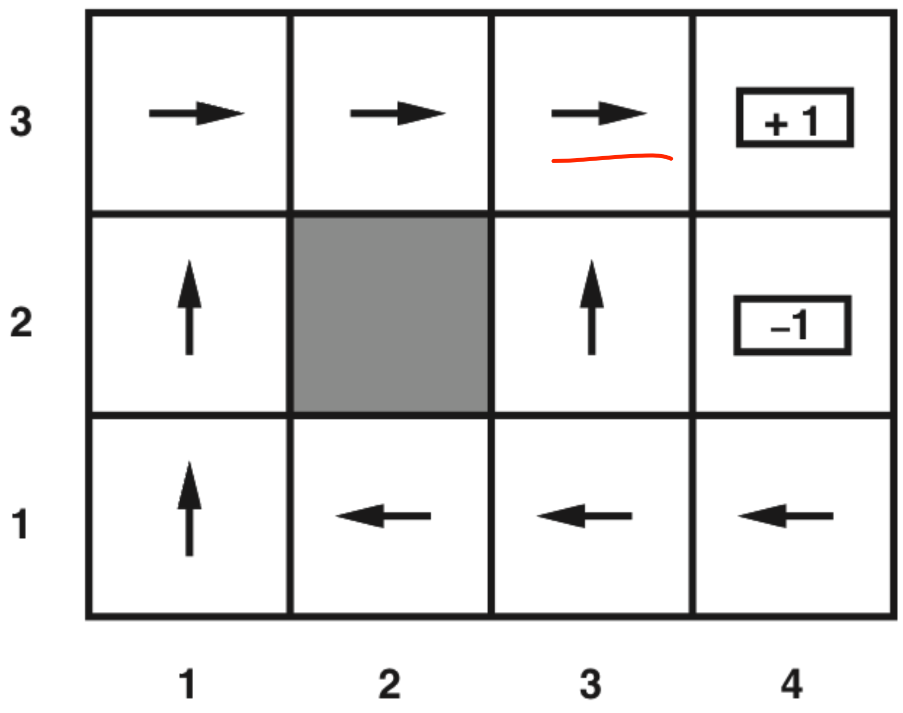
$$\begin{pmatrix} P(1 \mid 1,\pi(1)) & \dots & P(n \mid 1,\pi(1)) \\ \vdots & & \\ P(1 \mid n,\pi(n)) & \dots & P(n \mid n,\pi(n)) \end{pmatrix}$$

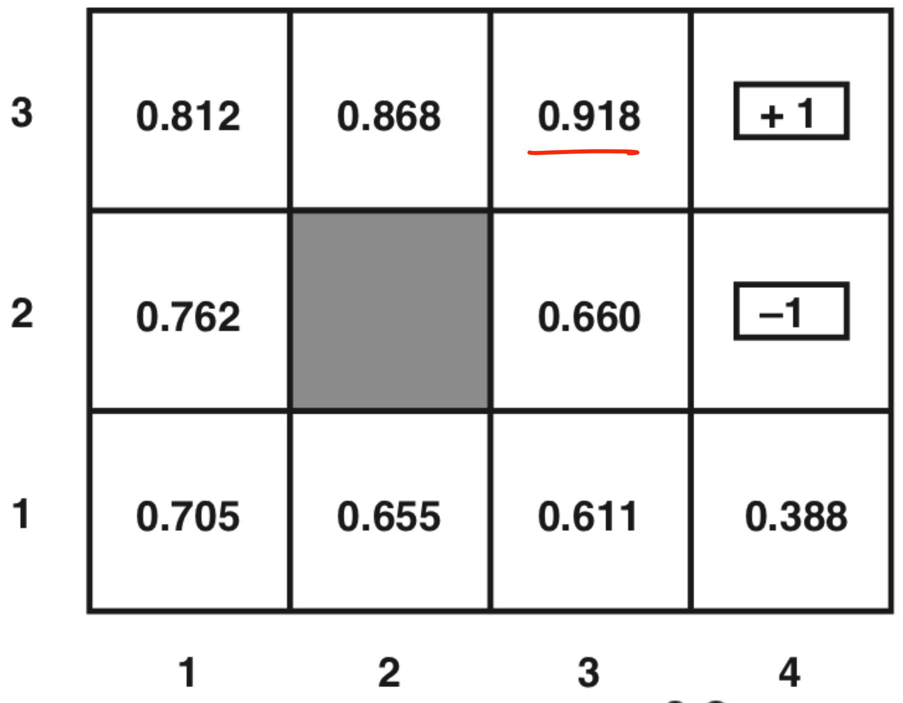
In matrix notation: $V^{\pi} = r + \gamma T V^{\pi}$

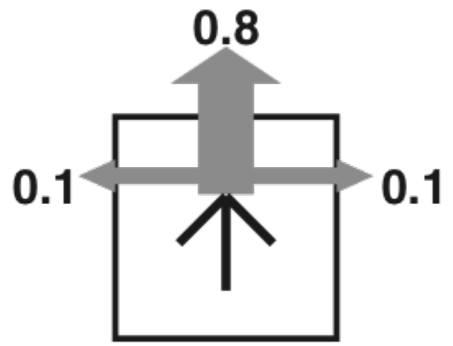
$\Rightarrow V^{\pi} = (I-\gamma T)^{-1} r$

➔ **Can compute V^π analytically, by matrix inversion! ☺**

Policies







How can we find the optimal policy?

A simple algorithm

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

Is this a good idea??

No!!

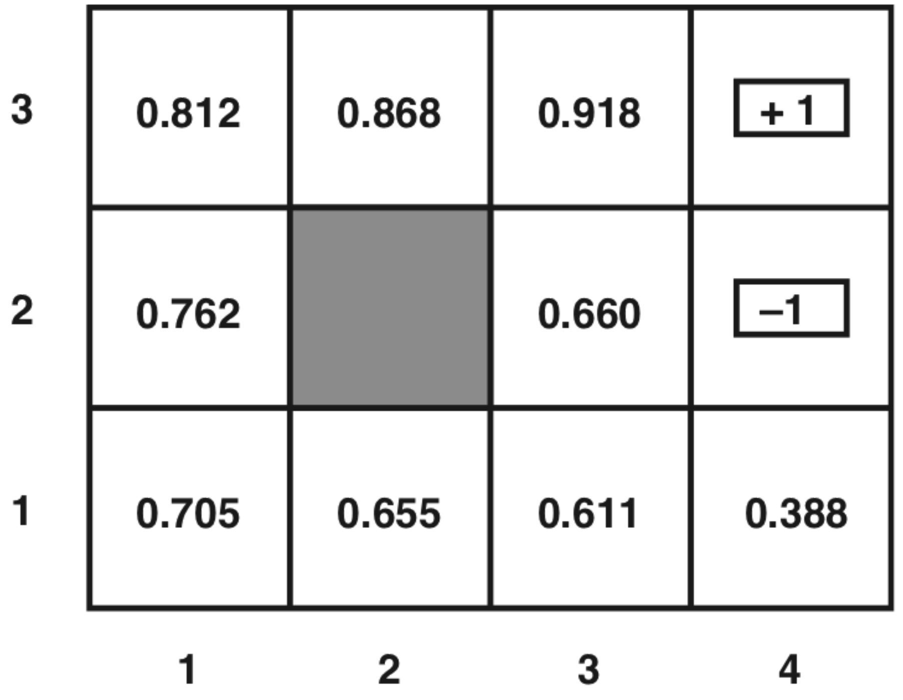
$\#$ policies $= |A|^{|S|}$

Suppose I give you the values

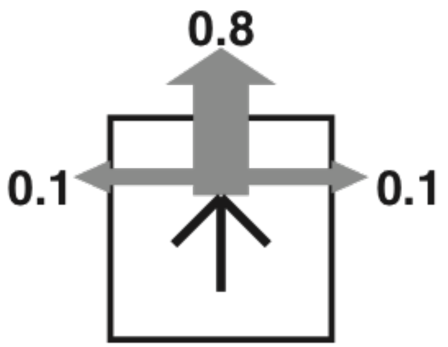
Sps we start in state x

$$Q(x,a) = r(x,a) + \sum_{x'} P(x'|x,a) \cdot V(x')$$

$$\Rightarrow a^* \in \arg\max_a Q(x,a)$$



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



Value functions and policies

$$\begin{tikzcd}
\boxed{\begin{array}{l}\text{Value function } V^{\pi} \\ V^{\pi}(x) = r(x,\pi(x)) + \\ \gamma \sum_{x'} P(x' | x,\pi(x))\, V^{\pi}(x')\end{array}} \arrow[r, bend left, "\text{Every value function induces a policy}"] & \boxed{\begin{array}{l}\text{Greedy policy w.r.t. } V \\ \pi_V(x) = \text{argmax}_a\, r(x,a) + \\ \qquad \gamma \sum_{x'} P(x' \mid x,a)\, V(x)\end{array}} \arrow[l, bend left, "\text{Every policy induces a value function}"]
\end{tikzcd}$$

Thm: Policy optimal ⇔ greedy w.r.t. its induced value function!

Policy iteration

- Start with a random policy π
- Until converged do:

```
        Compute value function $V_\pi(x)$
        Compute greedy policy $\pi_G$ w.r.t. $V_\pi$
        Set $\pi \leftarrow \pi_G$
```

- Guaranteed to
  - Monotonically improve $\forall t, x : V^{\pi_{t+1}}(x) \geq V^{\pi_t}(x)$
  - Converge to an optimal policy π^*
- Often performs really well!
- Not known whether it's polynomial in |X| and |A|!

Alternative approach

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

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

- Compute V^* using dynamic programming:

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

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

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

$V_{t+1}(x) =$ ——— " ——— $V_t(x')$

Value iteration

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

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

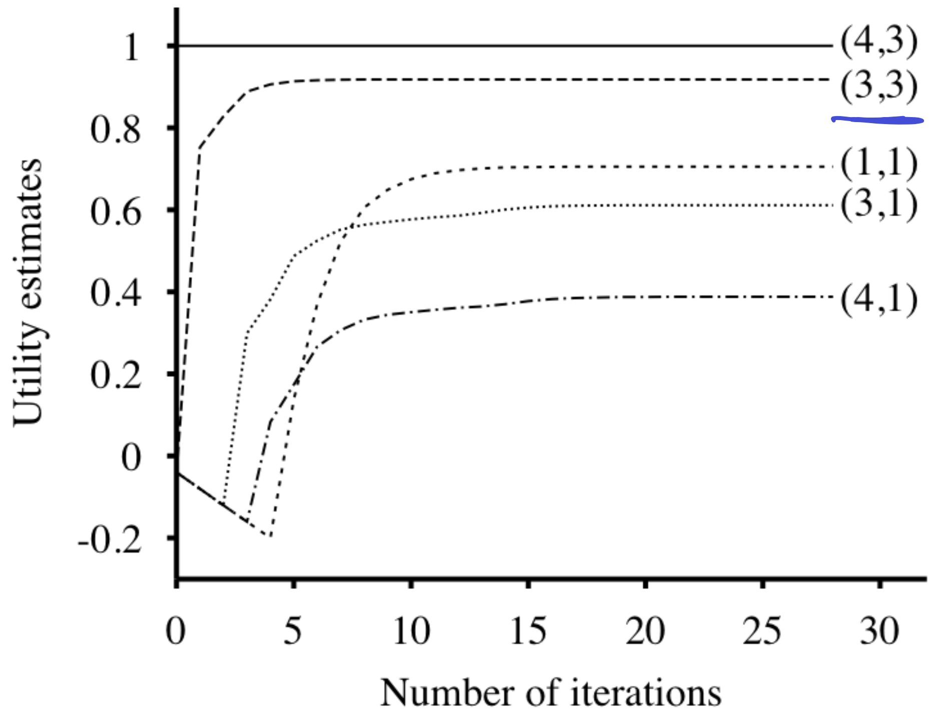
  For each x let $V_t(x) = \max_a Q_t(x,a)$

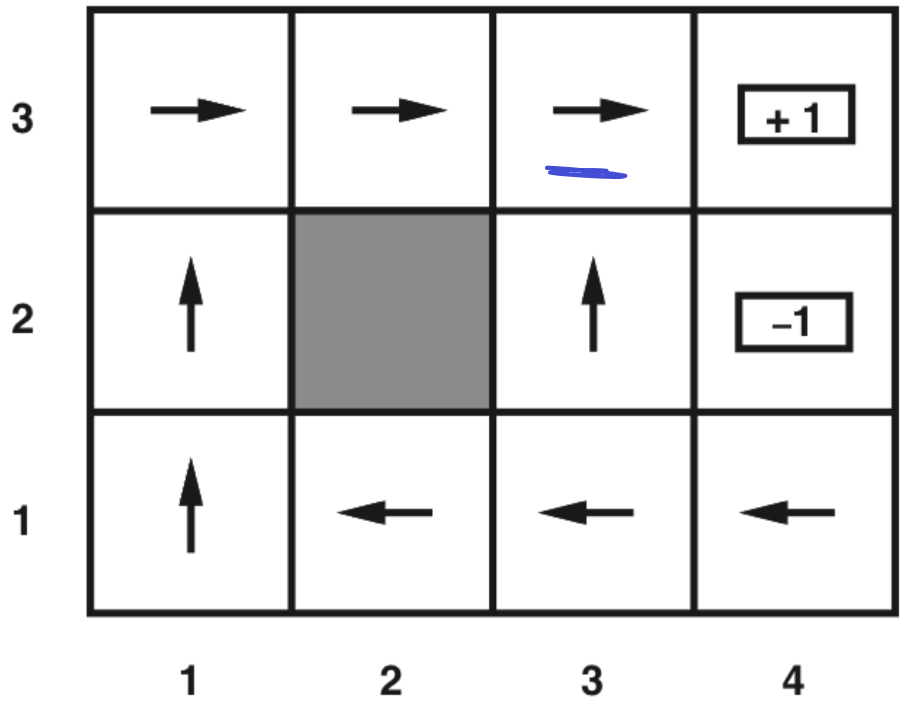
  Break if $\max_x \left| V_t(x) - V_{t-1}(x) \right| \leq \varepsilon$

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

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

Value iteration





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

Applications of MDPs

- Robot path planning (noisy actions)
- Elevator scheduling
- Manufactoring processes
- Network switching and routing
- AI in computer games
- …