Active Learning and Optimized Information Gathering

Lecture 3 – Reinforcement Learning

CS 101.2
Andreas Krause

Announcements

- Homework 1: out tomorrow
  - Due Thu Jan 22
- Project
  - Proposal due Tue Jan 27 (start soon!)
- Office hours
  - Come to office hours before your presentation!
  - Andreas: Friday 12:30-2pm, 260 Jorgensen
  - Ryan: Wednesday 4:00-6:00pm, 109 Moore

2

Course outline

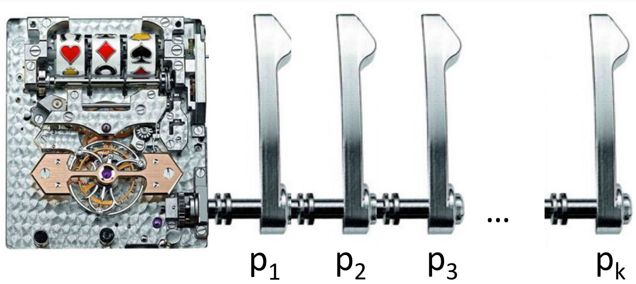
1. Online decision making

2. Statistical active learning

3. Combinatorial approaches

3

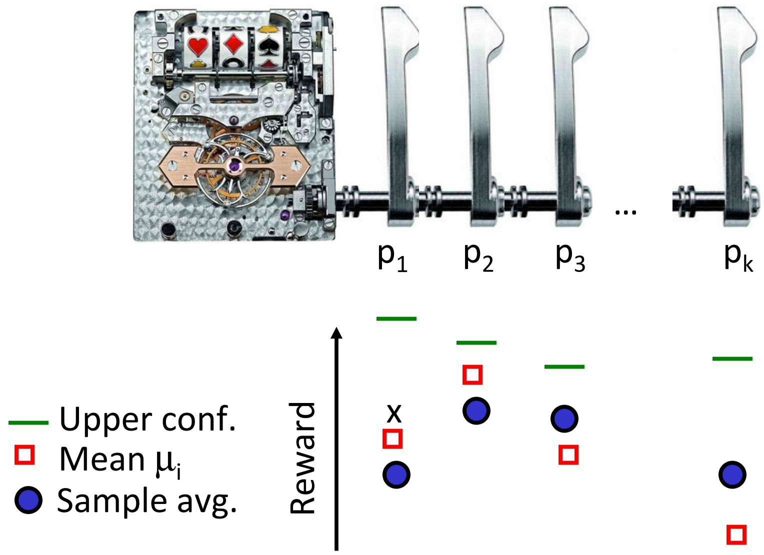
k-armed bandits



- Each arm i gives reward $X_{i,t}$ with mean μ_i

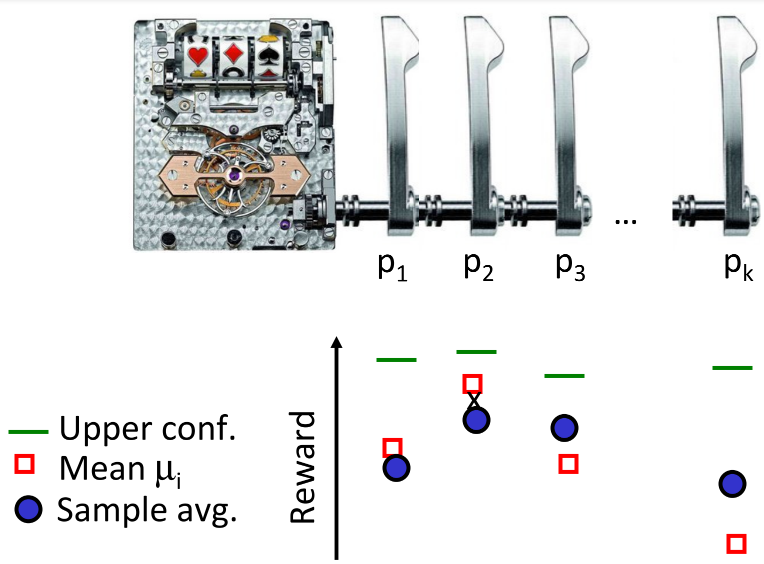
4

UCB 1 algorithm: Implicit exploration



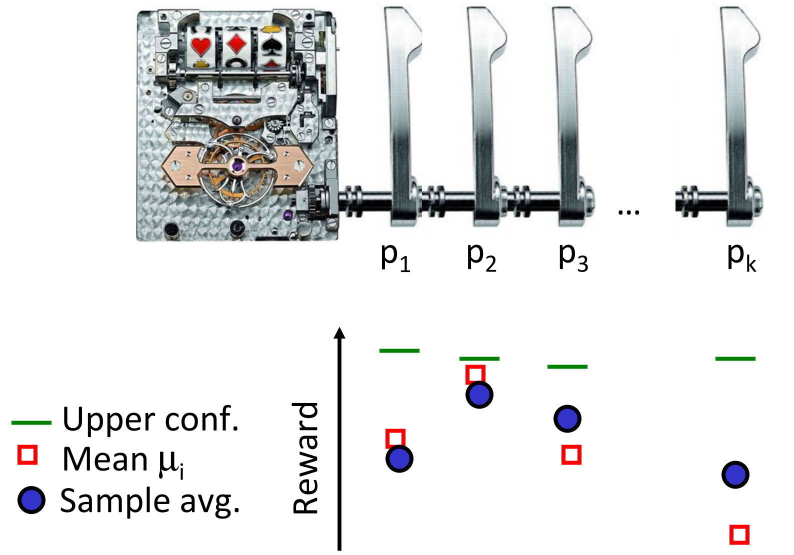
5

UCB 1 algorithm: Implicit exploration



6

UCB 1 algorithm: Implicit exploration



7

Performance of UCB 1

Last lecture:

For each suboptimal arm j: $E[T_j] = O(\log n/\Delta_j)$

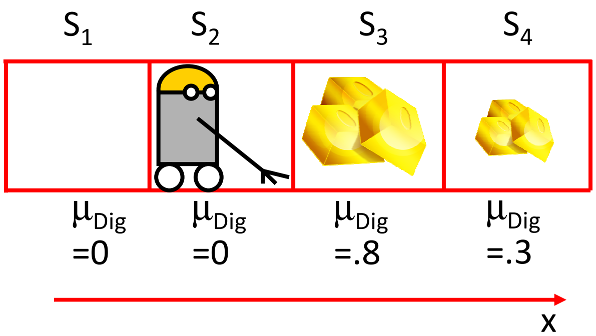
See notes on course webpage

This lecture:

What if our actions change the expected reward μ_i??

8

Searching for gold (oil, water, …)



Three actions:

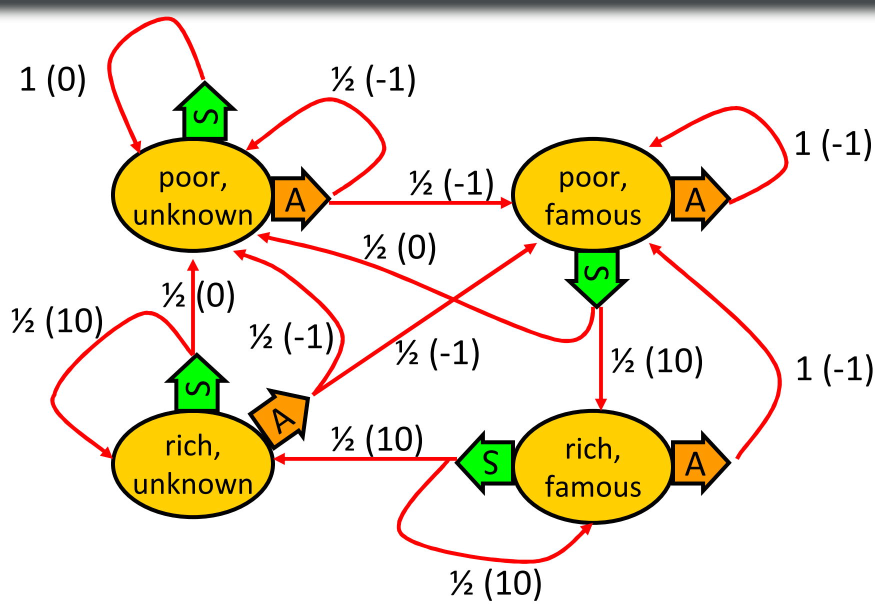
- Left
- Right
- Dig

$\mu_{Left} = 0$
$\mu_{Right} = 0$

- Mean reward depends on internal state!
- State changes by performing actions

9

Becoming rich and famous



10

Markov Decision Processes

- An MDP has
  - A set of states $S = \{s_1,\ldots,s_n\}$ …
  - with reward function r(s,a) [random var. with mean $\mu_s = r(s,a)$]
  - A set of actions $A = \{a_1,\ldots,a_m\}$
  - Transition probabilities
    $P(s'|s,a)$ = Prob(Next state = s' | Action a in state s)
- For now assume r and P are known!
- Want to choose actions to maximize reward
  - Finite horizon
  - Discounted rewards

11

Finite horizon MDP Decision model

- Reward R = 0
- Start in state s
- For t = 0 to n
  - Choose action a
  - Obtain reward $R \leftarrow R + r(s,a)$
  - End up in state s' according to $P(s'|s,a)$
  - Repeat with $s \leftarrow s'$
- Corresponds to rewards in bandit problems we've seen

12

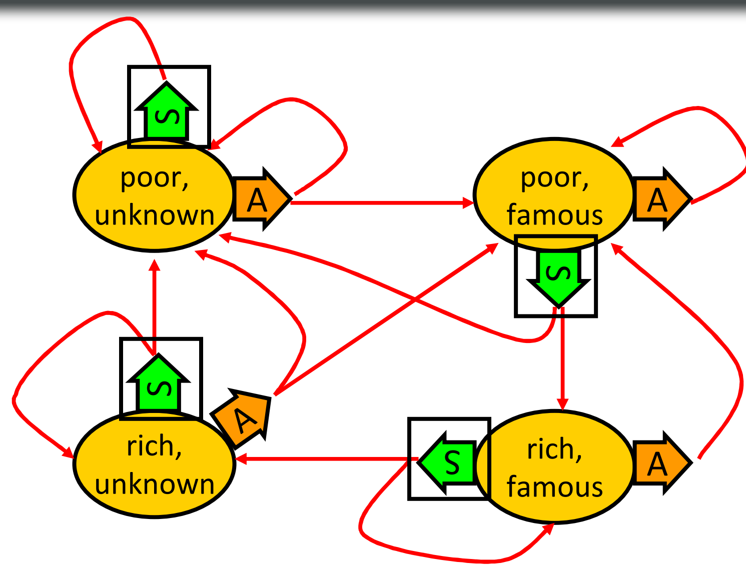
Discounted MDP Decision model

- Reward R = 0
- Start in state s
- For t = 0 to ∞
  - Choose action a
  - Obtain **discounted** reward $R = R + \gamma^t \, r(s,a)$ $0<\gamma<1$
  - End up in state s' according to $P(s'|s,a)$
  - Repeat with $s \leftarrow s'$

This lecture: Discounted rewards

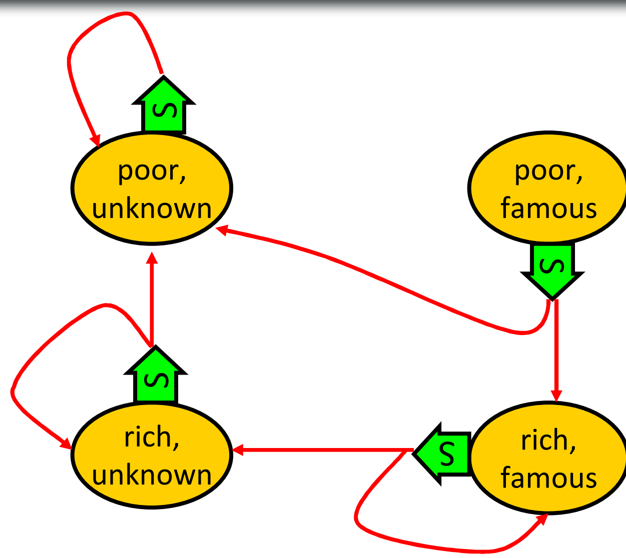
- Fixed probability $(1-\gamma)$ of "obliteration" (inflation, running out of battery, …)

Policies



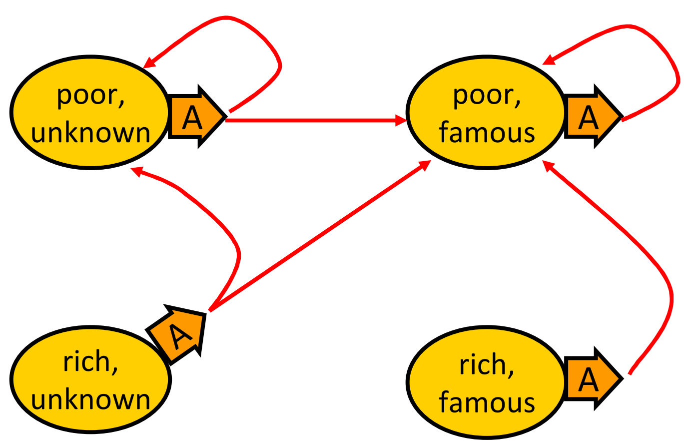
Policy: Pick one fixed action for each state

Policies: Always save?



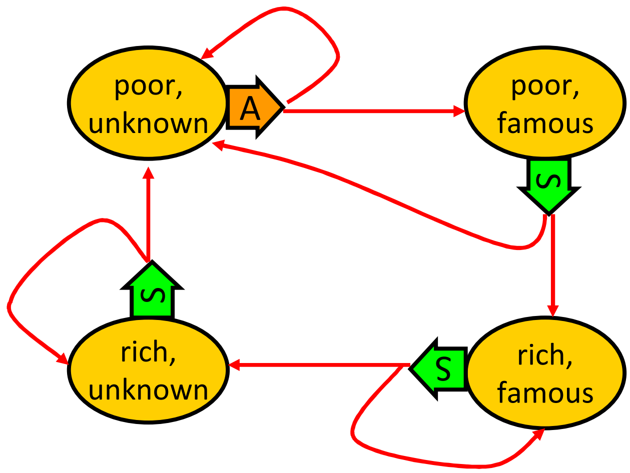
15

Policies: Always advertise?



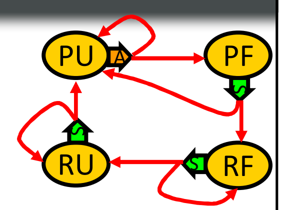
16

Policies: How about this one?



17

Planning in MDPs



- Deterministic policy $\pi: S \rightarrow A$
- Induces a **Markov chain**: $S_1, S_2, \ldots, S_t, \ldots$
  with transition probabilities

$$P(S_{t+1}=s' \mid S_t=s) = P(s' \mid s, \pi(s))$$

- Expected value $J(\pi) = E[\ r(S_1,\pi(S_1)) + \gamma\, r(S_2,\pi(S_2)) + \gamma^2\, r(S_3,\pi(S_3)) + \ldots\ ]$

18

Computing the value of a policy

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

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

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

and $J(\pi) = V^\pi(\text{start state})$

In matrix notation: $V^\pi = r + \gamma P V^\pi$

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

How can we find the optimal policy?

19

A simple algorithm

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

Is this a good idea??

20

Value functions and policies

Every value function induces a policy

Value function V^π

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

Greedy policy w.r.t. V

$\pi_V(s) = \text{argmax}_a\, r(s,a) + \gamma \sum_{s'} P(s' \mid s,a)\, V(s')$

Every policy induces a value function

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

21

Policy iteration

- Start with a random policy π
- Until converged do:
  - Compute value function $V_\pi(s)$
  - Compute greedy policy π_G w.r.t. V_π
  - Set $\pi \leftarrow \pi_G$
- Guaranteed to
  - Monotonically improve
  - Converge to an optimal policy π^*
- Often performs really well!
- Not known whether it's polynomial in |S| and |A|!

22

Alternative approach

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

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

- Compute V^* using dynamic programming:

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

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

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

  $V_{t+1}(s) =$ $V_t(s')$

23

Value iteration

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

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

  For each s let $V_t(s) = \max_a r(s,a) + \gamma Q_t(s,a)$

  Break if $\max_s | V_t(s) - V_{t-1}(s) | \le \varepsilon$

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

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

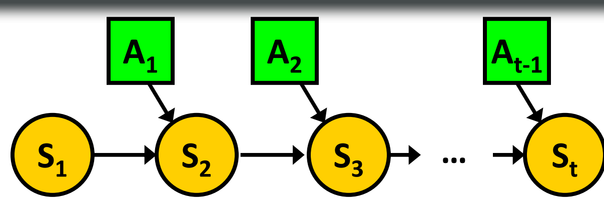
24

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
    $V_t(s) = \max_a r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) V_{t-1}(s)$
- Linear programming

25

MDP = controlled Markov chain

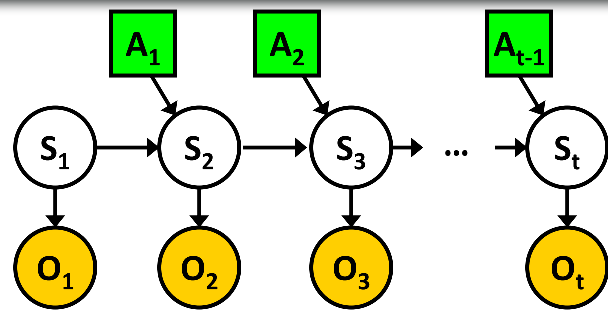


Specify $\mathbf{P(S_{t+1} \mid S_t,a)}$

- State fully observed at every time step
- Action A_t controls transition to S_{t+1}

26

POMDP = controlled HMM



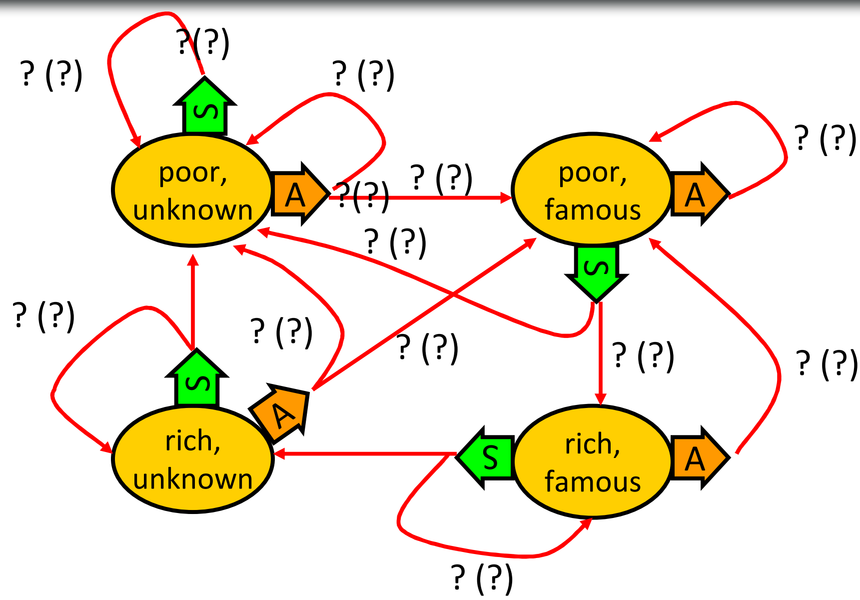
Specify **$P(S_{t+1} \mid S_t, a_t)$**
$P(O_t \mid S_t)$

- Only obtain noisy observations O_t of the hidden state S_t
- **Very powerful model! ☺**
- **Typically extremely intractable ☹**

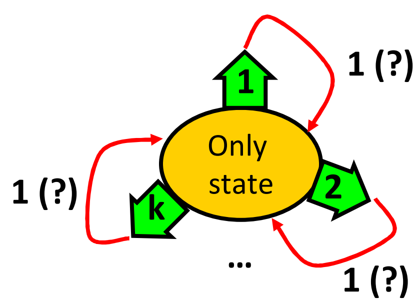
Applications of MDPs

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

What if the MDP is not known??



Bandit problems as unknown MDP



Special case with only 1 state, unknown rewards

Reinforcement learning

World: "You are in state s_{17}. You can take actions a_3 and a_9"

Robot: "I take a_3"

World: "You get reward -4 and are now in state s_{279}. You can take actions a_7 and a_9"

Robot: "I take a_9"

World: "You get reward 27 and are now in state s_{279}… You can take actions a_2 and a_{17}"

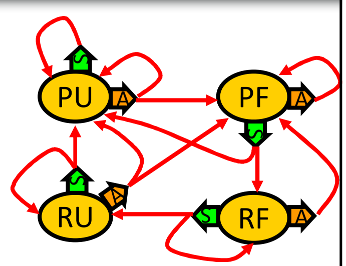
…

Assumption: States change according to some (unknown) MDP!

31

Credit Assignment Problem

State	Action	Reward
PU	A	0
PU	S	0
PU	A	0
PF	S	0
PF	A	**10**
…	…	…



"Wow, I won! How the heck did I do that??"

Which actions got me to the state with high reward??

32

Two basic approaches

1) Model-based RL

- Learn the MDP
  - Estimate transition probabilities P(s' | s,a)
  - Estimate reward function r(s,a)
- Optimize policy based on estimated MDP

Does not suffer from credit assignment problem! ☺

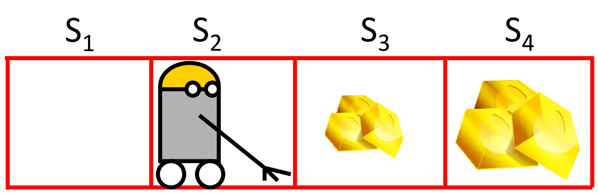
2) Model-free RL (later)

- Estimate the value function directly

33

Exploration–Exploitation Tradeoff in RL

- We have seen part of the state space and received a reward of 97.



- Should we
  - **Exploit:** stick with our current knowledge and build an optimal policy for the data we've seen?
  - **Explore:** gather more data to avoid missing out on a potentially large reward?

34

Possible approaches

- Always pick a random action?
  - Will eventually converge to optimal policy ☺
  - Can take very long to find it! ☹
- Always pick the best action according to current knowledge?
  - Quickly get some reward
  - Can get stuck in suboptimal action! ☹

Possible approaches

- ε_n greedy
  - With probability ε_n: Pick random action
  - With probability $(1-\varepsilon_n)$: Pick best action
  - Will converge to optimal policy with probability 1 ☺
  - Often performs quite well ☺
  - Doesn't quickly eliminate clearly suboptimal actions ☹
- What about an analogy to UCB1 for bandit problems?

The R_{max} Algorithm [Brafman & Tennenholz]

Optimism in the face of uncertainty!

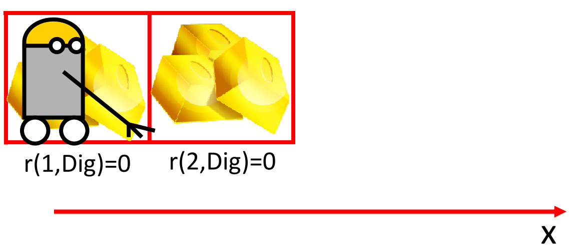
- If you don't know r(s,a):
  - Set it to R_{max}!
- If you don't know P(s' | s,a):
  - Set P(s* | s,a) = 1 where s* is a "**fairy tale**" state:

$r(s^*,a) = R_{max}$

$P(s^* | s^*,a) = 1$

37

Implicit Exploration Exploitation in R_{max}



Three actions:

- Left
- Right
- Dig

r(i,Left) =0

r(i,Right)=0

Like UCB1:
Never know whether we're exploring or exploiting! ☺

38

Exploration—Exploitation Lemma

Theorem: Every T timesteps, w.h.p., R_{max} either
- Obtains near-optimal reward, or
- Visits at least one unknown state-action pair

- T is related to the mixing time of the Markov chain of the MDP induced by the optimal policy

39

The R_{max} algorithm

Input: Starting state s_0, discount factor γ

Initially:
- Add fairy tale state s^* to MDP
- Set $r(s,a) = R_{max}$ for all states s and actions a
- Set $P(s^* \mid s,a) = 1$ for all states s and actions a

Repeat:
- Solve for optimal policy π according to current model P and R
- Execute policy π
- For each visited state action pair s, a, update r(s,a)
- Estimate transition probabilities P(s' | s,a)
- If observed "enough" transitions / rewards, recompute policy π

40

How much is "enough"?

How many samples do we need to accurately estimate $P(s' \mid s,a)$ or $r(s,a)$??

Hoeffding-Chernoff bound (from last lecture!):

- $X_1, \ldots, X_n$ i.i.d. samples from Bernoulli distribution w. mean μ
- $P(|1/n \sum_i X_i - \mu| \geq \varepsilon) \leq 2\, e^{-2n\varepsilon^2}$

Performance of R_{max} [Brafman & Tennenholz]

Theorem:

With probability $1-\delta$, R_{max} will reach an ε-optimal policy in $O(|S|\ |A|\ T / (\varepsilon\ \delta))$

Proof sketch:

Every T timesteps
- get ε-opt. rew. or
- Accurately identify $P(s'|s,a)$, $r(s,a)$

Theorem: Can get logarithmic regret bounds using slight modification of R_{max} (Auer et al, NIPS '06)

Challenges of RL

- Curse of dimensionality
  - MDP and RL polynomial in |A| and |S|
  - Structured domains (chess, multiagent planning, …): |S|, |A| exponential in #agents, state variables, …
  - ➔ Learning / approximating value functions (regression)
  - ➔ Approximate planning using factored representations
- Risk in exploration
  - Random exploration can be disastrous
  - ➔ Learn from "safe" examples: Apprenticeship learning

43

What you need to know

- MDPs
  - Policies
  - value- and Q-functions
- Techniques for solving MDPs
  - Policy iteration
  - Value iteration
- Reinforcement learning = learning in MDPs
- Model-based / model-free RL
- Different strategies for trading off exploration and exploitation
  - Implicit: R_{max}, like UCB1, optimism in the face of uncertainty
  - Explicit: ε_n greedy

44

Acknowledgments

- Some material used from Andrew Moore's MDP / RL tutorials: http://www.cs.cmu.edu/~awm/
- Presentation of R_{max} based on material from CMU 10-701 (Carlos Guestrin)

45