Introduction to Artificial Intelligence

Lecture 19 – Reinforcement Learning

CS/CNS/EE 154

Andreas Krause

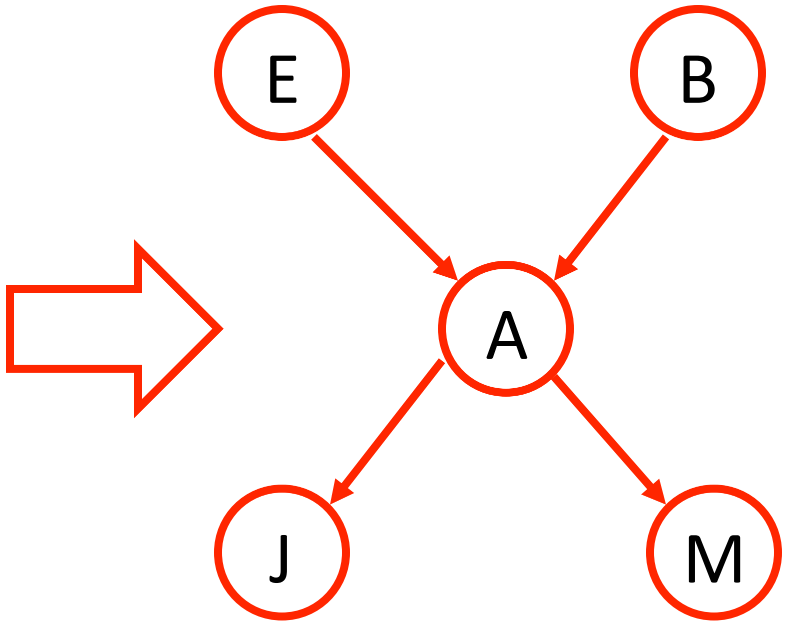
Announcements

- Exam:
  - December 8 10am till December 9 10am
  - Details posted on webpage
  - **Recitation this Thursday**
- Final project due December 7
- PLEASE fill out the course evaluation forms!
  Your feedback is extremely important!

Learning BN from Data

- Two main parts:
  - Learning structure (conditional independencies)
  - Learning parameters (CPDs)

E	B	A	M	J
0	0	0	0	0
1	0	1	1	0
1	0	1	1	1
0	1	1	0	1
0	1	0	0	1
…	…	…	…	…



Algorithm for Bayes net MLE

- Given:
  - Bayes Net structure G
  - Data set D of complete observations
- For each variable X_i estimate

$$\theta_{X_i|\mathbf{Pa}_i} = \frac{Count(X_i, \mathbf{Pa})}{Count(\mathbf{Pa}_i)}$$

- Results in globally optimal maximum likelihood estimate!

Pseudo-counts

- Make prior assumptions about parameters
- E.g.,: A priori, assume coin to be fair
- Practical approach: Assume we've seen a certain number of heads / tails:

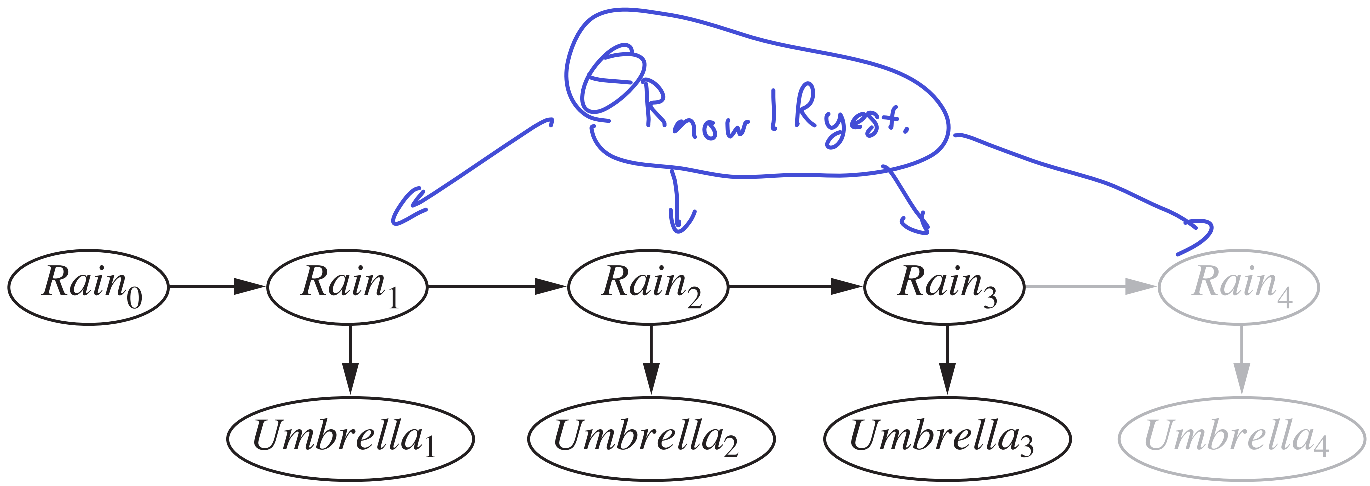
$$\theta_{F=c} = \frac{Count(F=c) + \alpha_c}{N + \alpha_c + \alpha_l}$$

"Pseudocounts"

- Looks like a hack.. In fact, this is equivalent to assuming a Beta prior

(Similar to the Gaussian prior for weights in regression)

Learning parameters for dynamical models

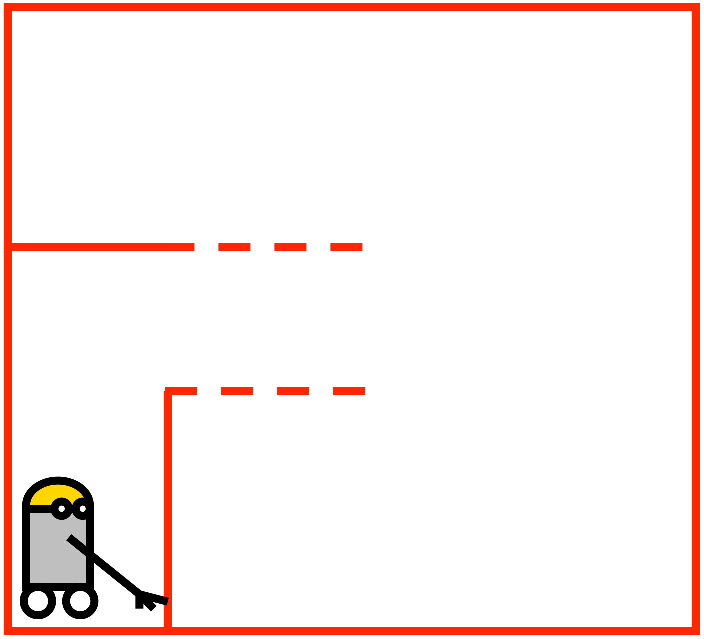


$$\Theta_{R_{now}=T \mid R_{yest}=T} = \frac{\sum_{t=1}^{T} \mathbb{I}(R_t=T, R_{t-1}=T)}{\sum_{t=1}^{T} \mathbb{I}(R_{t-1}=T)}$$

Summary

- To learn a Bayes net, need to
  - Learn structure
  - Learn parameters
- If all variables are observed
  - Get maximum likelihood parameter estimate by counting
  - Use pseudo-counts (Beta prior) to avoid overfitting
- Search for structure by maximizing score function
  - Score = Likelihood for best choice of parameters
- Can find optimal trees efficiently!

Learning from actions



Action	Reward
Forward	0
Left	0
Forward	0
Right	0
…	
Forward	**10**

- Want to learn a mapping from actions to rewards
- Credit assignment problem: which actions got me to the large reward??

Reinforcement learning

Agent actions *change* the state of the world (in contrast to supervised learning)

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

Agent: "I take a_3"

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

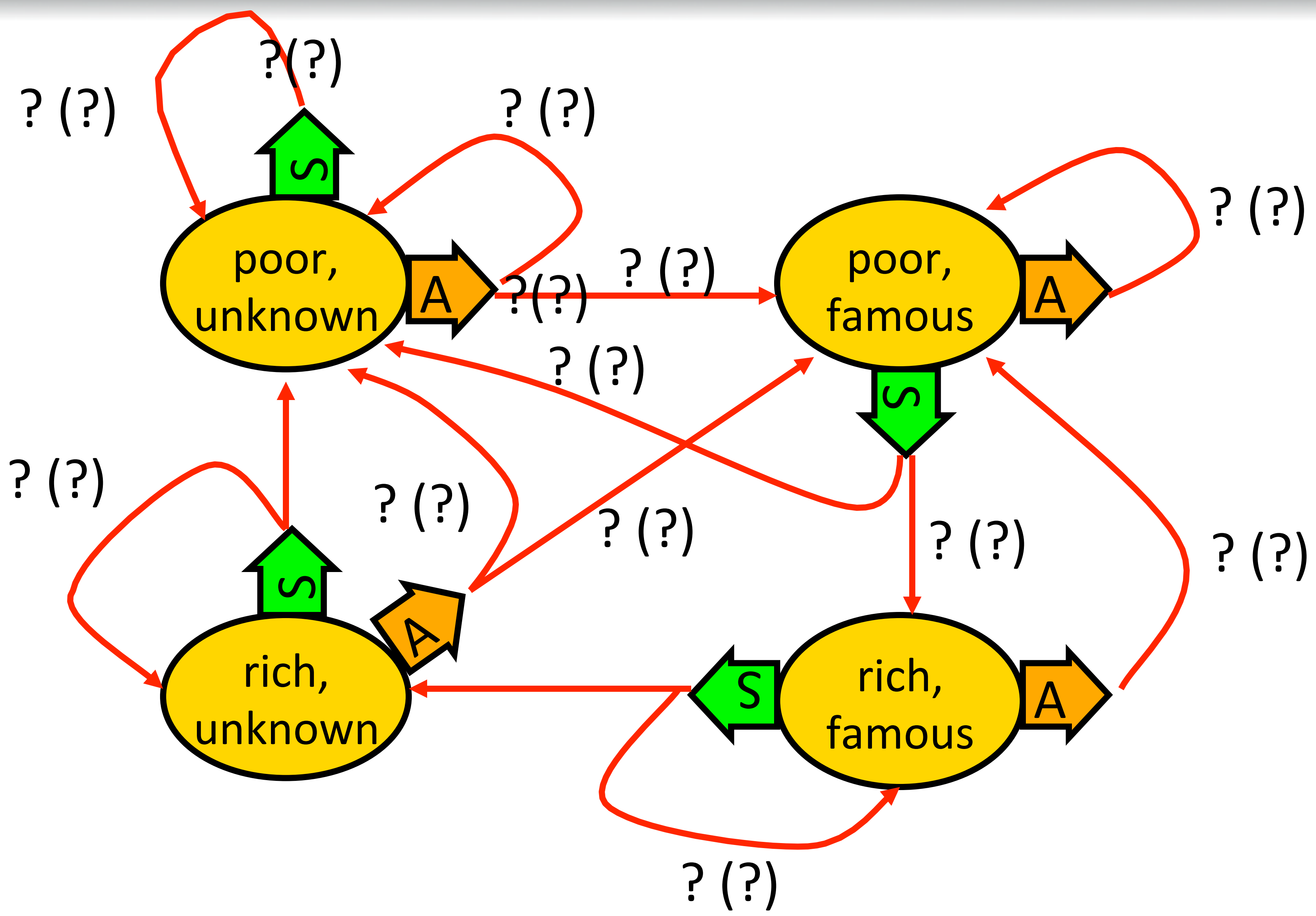
Agent: "I take a_9"

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

…

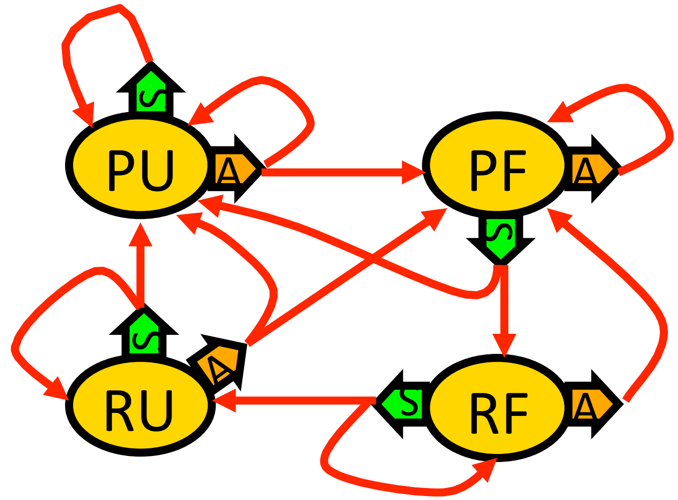
Assumption: States change according to some (unknown) MDP!

RL = Planning in unknown MDPs



Solving the Credit Assignment Problem

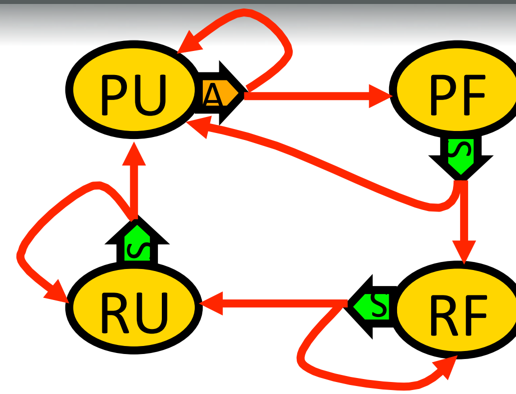
State	Action	Reward
PU	A	0
PU	S	0
PU	A	0
PF	S	0
PF	A	10
PF	A	10
…	…	…



Observed state transitions and rewards let you **learn** the underlying MDP!

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

Computing the value of a policy

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

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

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 $J(\pi) =$ $V^\pi(x_0)$ ← start state

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

$V^\pi = [V^\pi(1) \ldots V^\pi(n)]^T$, $r = [r(1,\pi(1)), \ldots r(n,\pi(n))]^T$

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

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

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

Value functions and policies

Every value function induces a policy

Value function V^π

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

Greedy policy w.r.t. V

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

Every policy induces a value function

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

Two basic approaches

1) Model-based RL ("Approximate dynamic programming")

- Learn the MDP

  Estimate transition probabilities P(s' | s,a)

  Estimate reward function r(s,a)

- Optimize policy based on estimated MDP

2) Model-free RL (later)

- Estimate the value function directly

Learning the MDP

- Need to estimate
  - transition probabilities $P(X_{t+1} \mid X_t, A)$
  - Reward function $r(X,A)$
- Can use techniques from last lecture (regularized maximum likelihood estimate)!
- Data set: $(x_1, a_1, r_1, x_2); (x_2, a_2, r_2, x_3); (x_3, a_3, r_3, x_4) \ldots$

$$P(X_{t+1} = x \mid X_t = x', A = a) = \frac{\text{Count}(X_{t+1} = x, X_t = x', A = a)}{\text{Count}(X_t = x', A = a)}$$

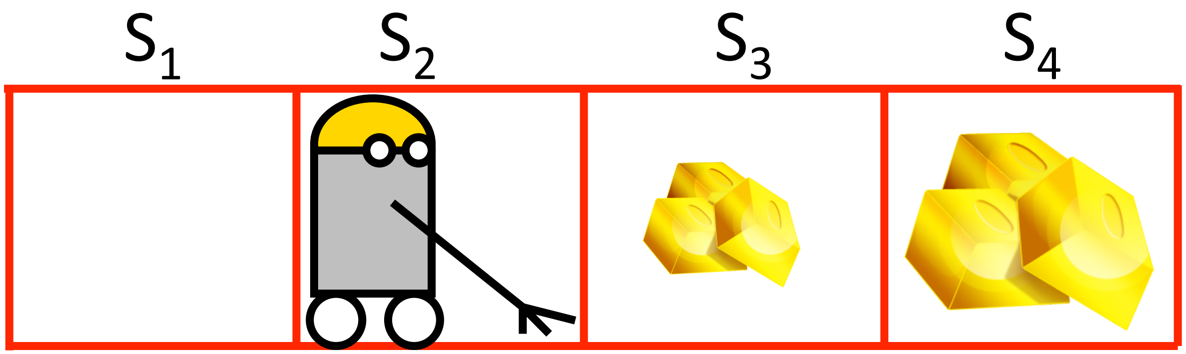
$$r(x, a) = \frac{1}{N_{x,a}} \sum_{t: X_t = x, A_t = a} r_t$$

RL is different from supervised learning

- So far, we have assumed we get i.i.d. data
- In reinforcement learning, the data we get *depends on our actions*!
- Some actions have higher rewards than others!

- *Dilemma*: Should we "collect more training data" or "choose high-reward actions"?

Exploration–Exploitation Dilemma in RL



- 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?

Possible approaches

- Always pick a random action?
  - Will eventually correctly estimate all probabilities and rewards ☺
  - May do extremely poorly! ☹

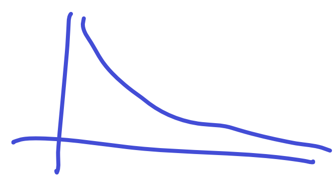
- Always pick the best action according to current knowledge (solve MDP with estimated parameters)?
  - Quickly get some reward
  - Can get stuck in suboptimal action! ☹

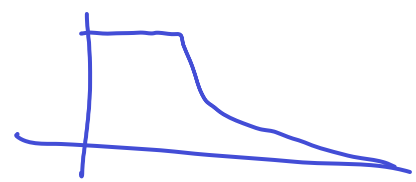
Possible approaches

- ε_n greedy
  - With probability ε_n: Pick random action
  - With probability $(1-\varepsilon_n)$: Pick best action

Want $\varepsilon_n \to 0$

Typically $\varepsilon_n = \frac{1}{n}$





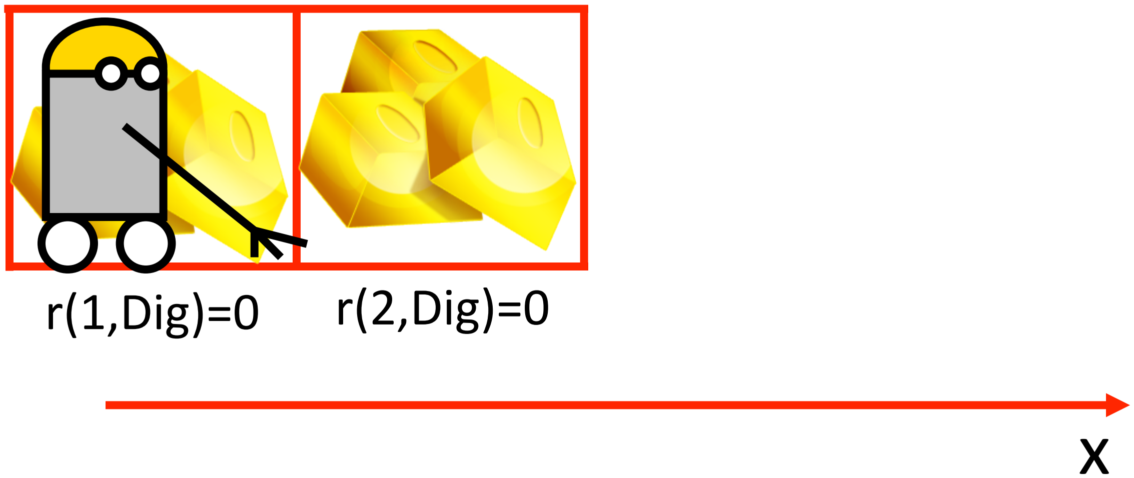
- Will converge to optimal policy with probability 1
- Often performs quite well
- Doesn't quickly eliminate clearly suboptimal actions

The R_{max} Algorithm [Brafman & Tennenholz '02]

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:

Implicit Exploration Exploitation in R_{max}



Three actions:

- Left
- Right
- Dig

r(i,Left) =0
r(i,Right)=0

Never need to explicitly choose whether we're exploring or exploiting!

Can rule out clearly suboptimal actions very quickly

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

The R_{max} algorithm

Input: Starting state x_0, discount factor γ

Initially:

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

Repeat:

- Execute policy π
- For each visited state action pair x, a, update r(x,a)
- Estimate transition probabilities $P(x' \mid x,a)$
- If observed "enough" transitions / rewards, recompute policy π according to current model P and r

How much is "enough"?

How many samples do we need to accurately estimate P (x' | x,a) or r(x,a)??

Hoeffding-Chernoff bound:

- $X_1, \ldots, X_n$ i.i.d. samples from Bernoulli distribution w. mean μ

$$P\left(\left|\mu - \frac{1}{n}\sum_i X_i\right| \geq \varepsilon\right) \leq 2\exp(-2n\varepsilon^2)$$

Sps want Error $\leq \varepsilon$ w. prob. $\geq 1-\delta$

Need $n \geq C \cdot \frac{1}{\varepsilon^2} \log \frac{1}{\delta}$

Performance of R_{max} [Brafman & Tennenholz]

Theorem:

With probability 1-δ, R_{max} will reach an ε-optimal policy polynomial in |S|, |A|, T, 1/ε and 1/δ

Problems of model-based RL?

- Memory required:

Need: $P(X_{t+1} | X_t, A_t) \rightarrow Count(X_{t+1}, X_t, A_t)$

$|S|^2 \cdot |A|$

$R(X_t, A_t) \rightarrow O(|S| \cdot |A|)$

- Computation time:

Need to compute opt. policy

E.g. policy iteration, need to compute value fn. $\Rightarrow O(|S|^3)$

Problems of model-based RL?

- Memory required: |A| |S|^2
- Computation time:
  - Need to frequently recompute optimal policy!

Model free RL

- Recall:
  1. Optimal value function $V^*(x)$ ➔ opt. policy π^*
  2. For optimal value function it holds:

$$V^*(x) = \max_a Q(x,a)$$

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

Key idea: Estimate Q(x,a) directly from samples!

Q-learning

- Estimate Q(x,a) = = r(x,a) + $\gamma \sum_{x`}$ P(x' | x,a) V*(x')
- Note that V*(x) = $\max_a$ Q(x,a)

- Suppose we
  - Have initial estimate of Q_0(x,a)
  - observe transition x, a, x' with reward r

$$Q_t(x,a) = r + \gamma V^*(x')$$
$$\approx r + \gamma \max_{a'} Q_{t-1}(x',a')$$

Unbiased estimate
Extremely high variance

Q-learning

Instead: $Q_t(x,a) = (1-\alpha_t)\underbrace{Q_{t-1}(x,a)}_{\text{prev. estimate}} + \underbrace{\alpha_t\left(r + \gamma \max_{a'} Q_{t-1}(x',a')\right)}_{\text{correction}}$

Theorem: If learning rate α_t satisfies

$\sum_t \alpha_t = \infty$

$\sum_t \alpha_t^2 < \infty$

} e.g. $\alpha_t = \frac{1}{t}$

and actions are chosen at random, then Q learning converges to optimal Q* with probability 1

How can we trade off exploration and exploitation?

Optimistic Q-learning

Similar to R_{max}:

Initialize $Q_0\,(x,a) = \prod_t (1-\alpha_t)^{-1}/(1-\gamma)\, R_{max}$

Theorem: With prob. $1-\delta$, optimistic Q-learning obtains an ε-optimal policy after a number of time steps that is polynomial in $|S|$, $|A|$, $1/\varepsilon$ and $1/\delta$

Properties of Q-learning

- Memory required: $O(|S| \cdot |A|)$
- Computation time: In every iteration: $O(|A|)$

  $a_t \in \arg\max_a Q(x, a)$

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