ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

inf | Informatik
Computer Science

Probabilistic Artificial Intelligence

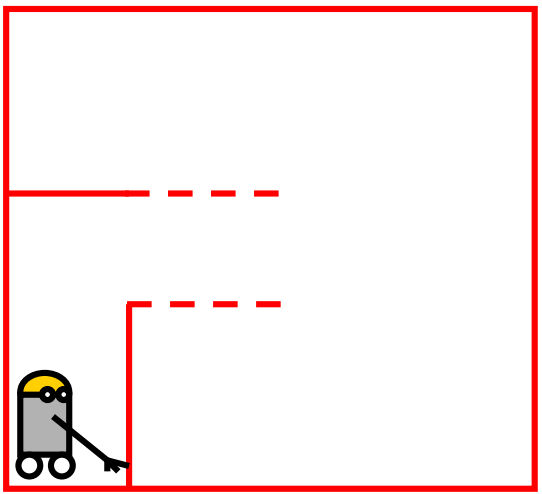
Introduction to Reinforcement Learning

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

Announcements

- No lecture next week (13.12.), only exercises

Learning to Act in Unknown Environments



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

- Learn a mapping from (seq. of) 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"

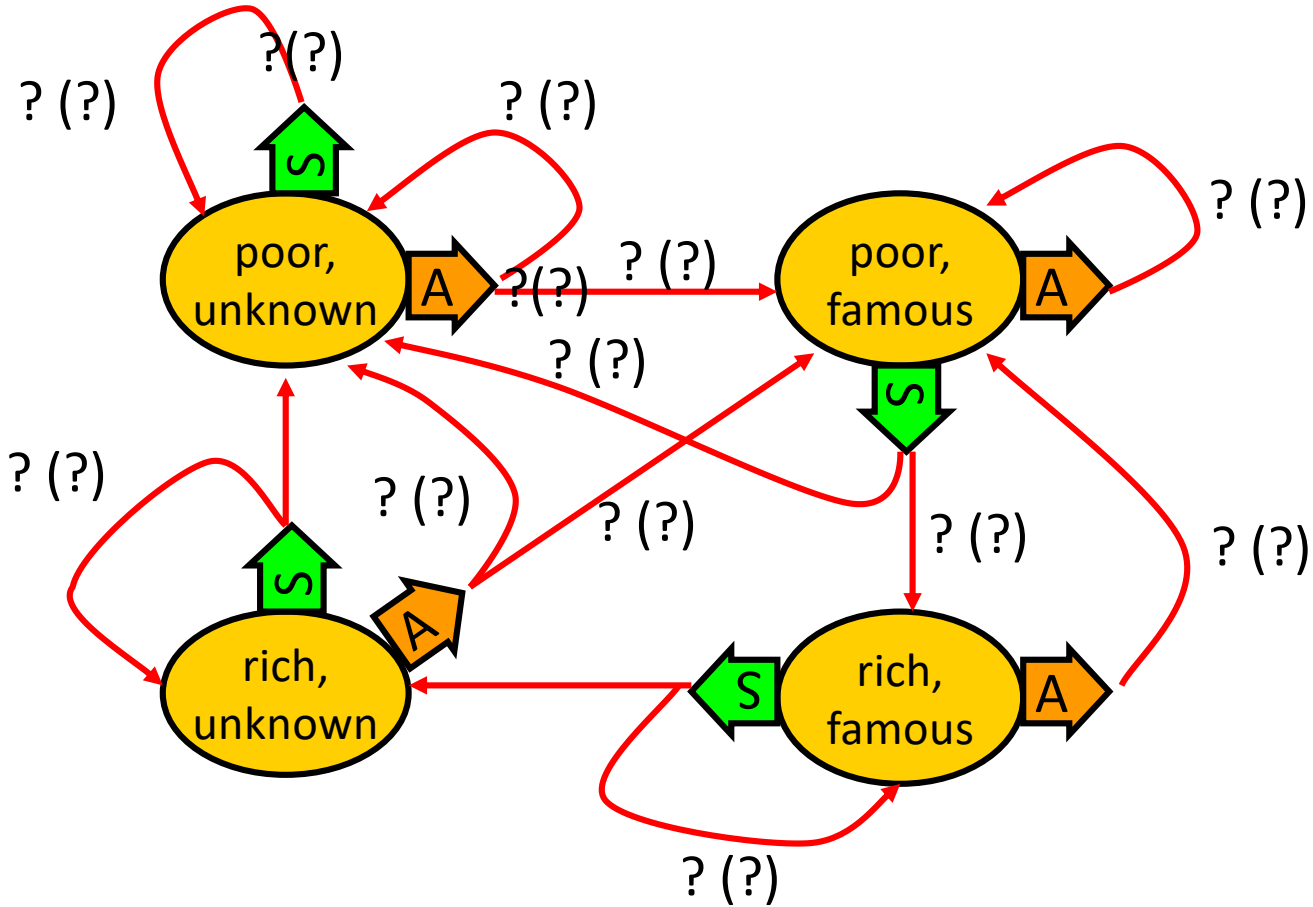
Agent: "I take a_9"

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

…

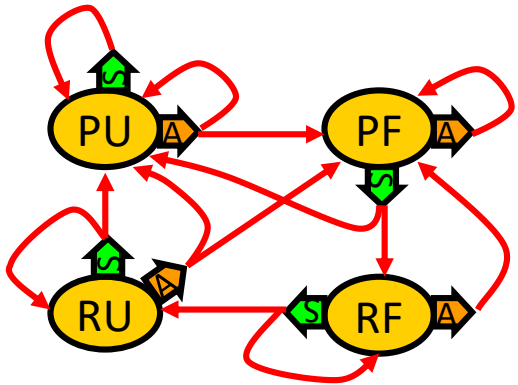
Assumption: States change according to some (unknown) MDP!

RL = Planning in unknown MDPs



Solving the Credit Assignment Problem

State x	Action a	Reward r
PU	A	0
PU	S	0
PU	A	0
PF	S	0
PF	A	10
PF	A	10
…	…	…

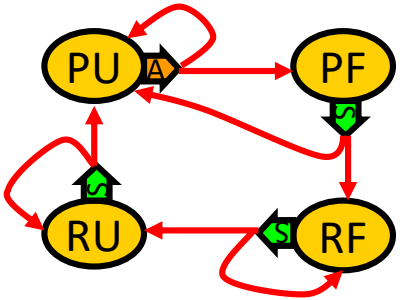


Observed state transitions and rewards let you **learn** the underlying MDP!

Recall: Planning in MDPs

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

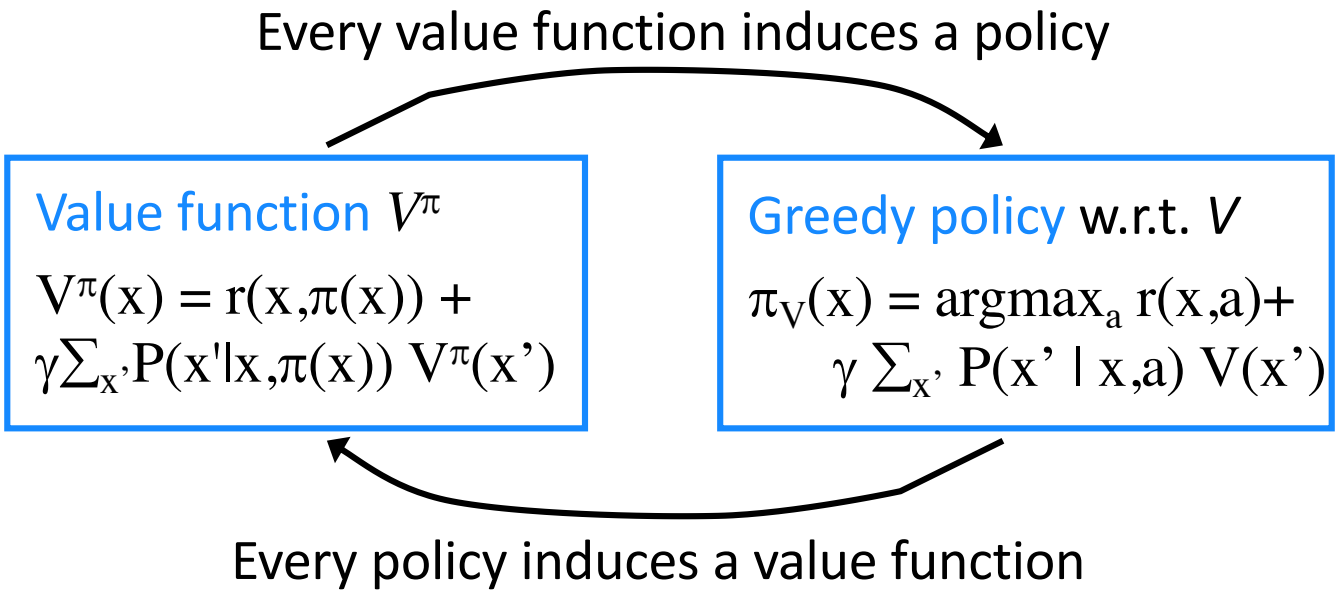
$$P(X_{t+1}=x' \mid X_t=x) = P(x' \mid x, \pi(x))$$



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

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

Value Functions and Policies



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

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

Reinforcement Learning

- RL is different from supervised learning
  - The data we get is not i.i.d.
  - In reinforcement learning, the data we get *depends on our actions*!
  - Some actions have higher rewards than others!

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

Two basic approaches to RL

1) Model-based RL

- Learn the MDP

  Estimate transition probabilities $P(x' \mid x, a)$

  Estimate reward function $r(x, a)$

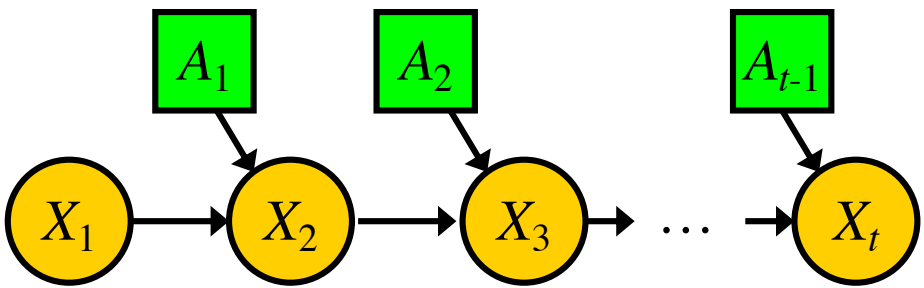
- Optimize policy based on estimated MDP

2) Model-free RL

- Estimate the value function directly;
- Policy gradient methods;
- Actor-critic methods.

Off-policy vs on-policy RL

- On-policy RL
  - Agent has full control over which actions to pick
  - Can choose how to trade exploration and exploitation
- Off-policy RL
  - Agent has no control over actions, only gets observational data (e.g., demonstrations, data collected by applying a different policy, …)



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 learning Bayes Nets: (regularized) maximum likelihood estimation
- Data set: $x_1, a_1, r_1, x_2, a_2, r_2, x_3, a_3, r_3, x_4 \ldots$

  $\hookrightarrow D = \{(x_1, a_1, r_1, x_2), (x_2, a_2, r_2, x_3), (x_3, a_3, r_3, x_4), \ldots\}$
- Estimate transitions:

$$P(X_{t+1} \mid X_t, A) \approx \frac{Count(X_{t+1}, X_t, A)}{Count(X_t, A)}$$

- Estimate rewards:

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

→ rewards experienced

Exploration-Exploitation Dilemma

- Always pick a random action?
  - Will eventually correctly estimate all probabilities and rewards ☺
  - May do extremely poorly in terms of rewards! ☹
- Always pick the best action according to current knowledge?
  - Quickly get some reward
  - Can get stuck in suboptimal action! ☹
- Balance exploration and exploitation (more later)

Trading Explorationand Exploitation

- ε_t greedy
  - With probability ε_t: Pick random action
  - With probability $(1\text{-}\varepsilon_t)$: Pick best action
- If sequence ε_t satisfies Robbins Monro (RM) conditions then will converge to optimal policy with probability 1

$$\sum_t \varepsilon_t = \infty \quad ; \quad \sum_t \varepsilon_t^2 < \infty$$

- Simple, often performs fairly 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 $r(x, a)$:
  - Set it to $R_{\max}$!

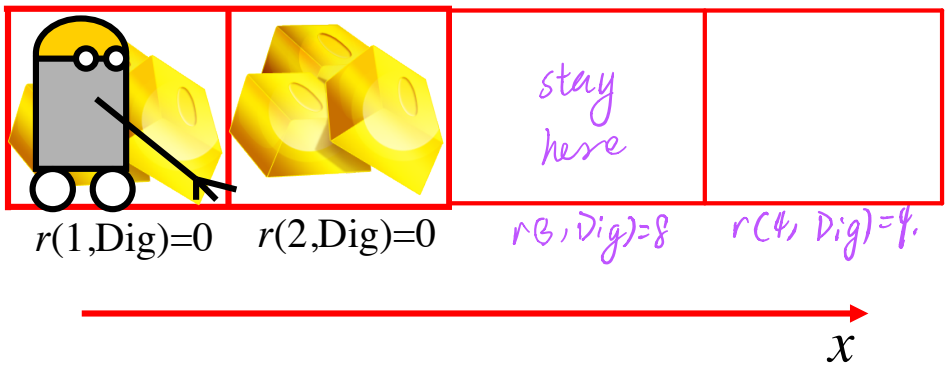
- If you don't know $P(x' \mid x, a)$:
  - Set $P(x^* \mid x, a) = 1$ where x^* is a **"fairy tale"** state:

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

$P(x^* \mid x^*, a) = 1 \quad \forall a$

stay at that state

Implicit Exploration Exploitation in R_{max}



Three actions:

- Left
- Right
- Dig

$$r(i, \text{Left}) = 0$$
$$r(i, \text{Right}) = 0$$

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

Can rule out clearly suboptimal actions very quickly

Exploration—Exploitation Lemma

with high probability

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 $P(x' \mid x, a)$ or $r(x,a)$?

Hoeffding-Chernoff bound:

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

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

Performance of R_{max} [Brafman & Tennenholz]

Theorem:

With probability $1\text{-}\delta$, $R_{\max}$ will reach an ε-optimal policy in a number of steps that is polynomial in $|X|$, $|A|$, T, $1/\varepsilon$ and $\log(1/\delta)$

Problems of model-based RL?

- Memory required: store $P(x'|x,a) \Rightarrow O(|X|^2 \cdot |A|)$
  $r(x,a) \Rightarrow O(|X| \cdot |A|)$
- Computation time:

Solving MDP once: $\text{poly}(|X|, |A|, \frac{1}{\varepsilon}, \log\frac{1}{\delta})$
Need to do this often!

Two basic approaches

1) Model-based RL

- Learn the MDP

  Estimate transition probabilities $P(x' \mid x, a)$

  Estimate reward function $r(x, a)$

- Optimize policy based on estimated MDP

2) Model-free RL

- Estimate the value function directly;
- Policy gradient methods;
- Actor-critic methods

Model free RL

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

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

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

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

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

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

$$Q(x,a) \leftarrow (1-\alpha_t) Q(x,a) + \alpha_t \left(r + \gamma \max_{a'} Q(x',a') \right)$$

$V(x')$

Q-learning

$$Q(x,a) \leftarrow (1-\alpha_t)Q(x,a) + \alpha_t\left(r + \gamma \max_{a'} Q(x',a')\right)$$

Theorem: If learning rate α_t satisfies

$\sum_t \alpha_t = \infty$

$\sum_t \alpha_t^2 < \infty$

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

How can we trade off exploration and exploitation?

Convergence of Optimistic Q-learning

[Even-dar & Mansour '02]

Similar to $R_{\max}$:

Initialize $Q(x,a) = \frac{R_{\max}}{1-\gamma} \prod_{t=1}^{T_{\text{init}}} (1-\alpha_t)^{-1}$

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

Properties of Q-learning

- Memory required: store $Q(x,a)$ $\Rightarrow O(|X||A|)$
- Computation time: At each iter.:

  pick $a_t = \arg\max_a Q(x,a)$ $\Rightarrow O(|A|)$

  update $Q(x,a_t)$ $\Rightarrow O(|A|)$

Challenges of RL

- MDP and RL polynomial in $|A|$ and $|X|$. Problem in:
  - Structured domains (chess, multiagent planning, …): $|X|$, $|A|$ exponential in #agents, state variables, …
  - Continuous domains ($|A|$ and $|X|$ infinite)

➔ Learning / approximating value functions (regression)

➔ Approximate planning using factored representations

- Risk in exploration
  - Random exploration can be disastrous

  ➔ Learn from "safe" examples: Apprenticeship learning

Acknowledgments

- Slides based on material accompanying the textbook "AI: A Modern Approach" (3rd edition) by S. Russell and P. Norvig and material by Carlos Guestrin