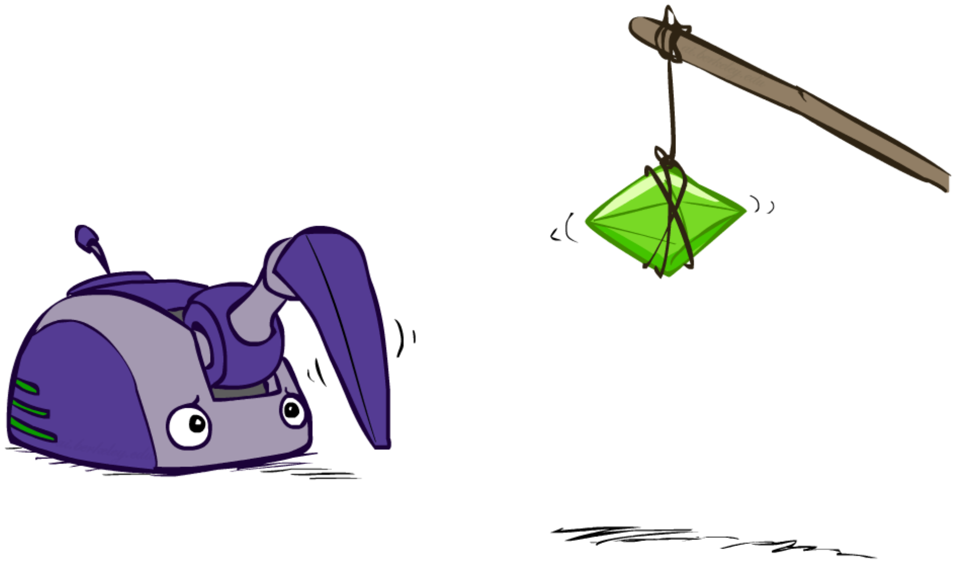
15-281 AI: Representation and Problem Solving

Reinforcement Learning I



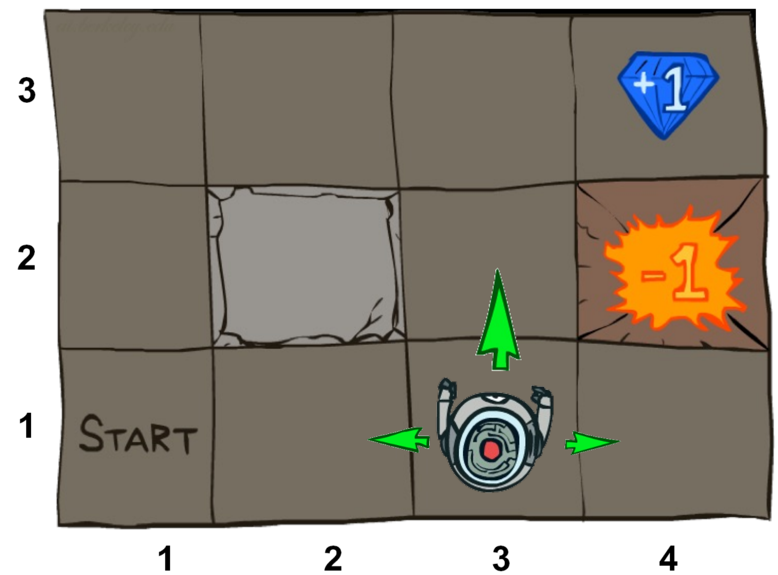
Instructors: Nihar Shah and Tuomas Sandholm

Carnegie Mellon University

Slide credits: CMU AI and ai.berkeley.edu

Recall Markov Decision Processes

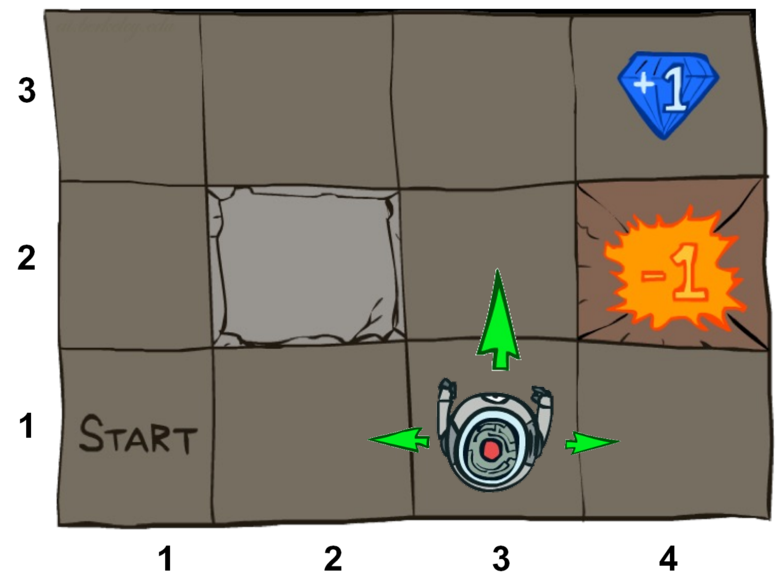
- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function $T(s, a, s')$
    - Probability that a from s leads to s', i.e., $P(s' \mid s, a)$
  - A reward function $R(s, a, s')$



All of this information is known beforehand

Reinforcement Learning (RL)

- RL is defined by:
  - A set of states s ∈ S
  - A set of actions a ∈ A
  - A transition function T(s, a, s’)
    - Probability that a from s leads to s’, i.e., P(s’ | s, a)
  - A reward function R(s, a, s’)



Then what is the difference with MDPs?

MDPs vs. Reinforcement Learning

MDPs

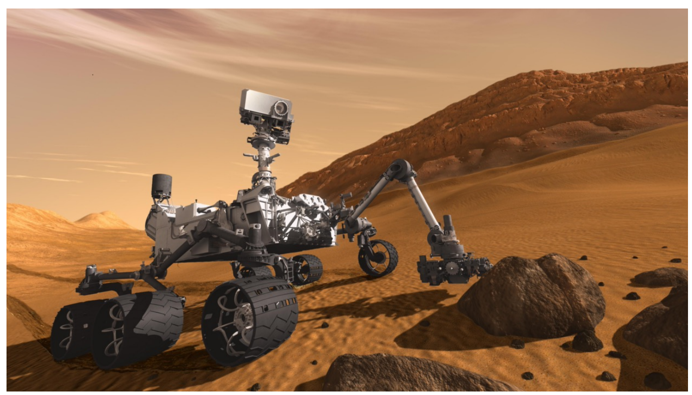
- You know the rewards and transitions (i.e., the model of the world) beforehand

- E.g., a prior rover etc. mapped the entire terrain which you can use now

Reinforcement Learning

- You don't know rewards and transitions beforehand

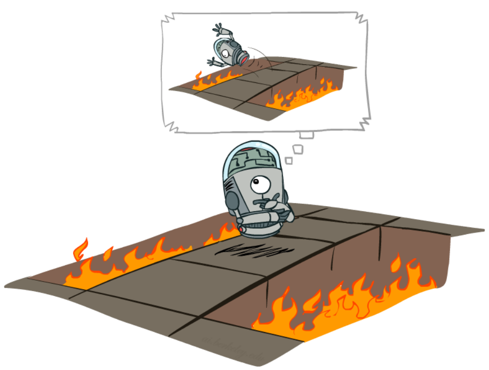
- E.g., the robot is deployed in an environment that is completely unseen before



MDPs vs. Reinforcement Learning

MDPs

- You know the rewards and transitions (i.e., the model of the world) beforehand
- Find policy to maximize total reward
- **Run our MDP solvers offline**
- **Only deploy the optimal policy**



Reinforcement Learning

- You don't know rewards and transitions beforehand
- Still assume the world is an MDP
- Do not know how the world works
- Find policy to maximize total reward
- **Need to take actions in the world to understand the world**

DeepMind Atari (©Two Minute Lectures)

Example: Learning to Walk

Initial

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – initial

Example: Learning to Walk

Training

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – traini

Example: Learning to Walk

Finished

[Kohl and Stone, ICRA 2004]

[Video: AIBO WALK – finish

Reinforcement Learning

We don't know T and R! How do we find policies?

We need to estimate quantities by trying out

At time step t,

- Take action a_t
- End up in new state s_{t+1}
- Observe reward = $R(s_t, a_t, s_{t+1})$

Data: $s_0, a_1, R_1, s_1, a_2, R_2, s_2, a_3, R_3, s_3, \ldots$

Passive vs Active RL

- Passive RL
  - Suppose policy π being used is already given to you
  - Only worry about how to learn from experience
  - Agent does **not** control the policy π of taking actions

- Active RL
  - Agent has to decide how to collect experience

Today: Passive RL

- Let π denote the policy being used
- One may like to learn the value obtained from this policy:
  - $V^{\pi}(s)$ = expected reward starting at state s and always following π
  - $Q^{\pi}(s, a)$=expected reward stating at s, taking action a now, and following π thereafter
  - One may like to learn the optimal policy: π^* or $V^*(s)$ or $Q^*(s, a)$

Data: $s_0, a_1, R_1, s_1, a_2, R_2, s_2, a_3, R_3, s_3, \ldots$

- Data collected via a policy π

Today: Policy evaluation

- For the policy π being used in the passive RL setting, how to estimate V^{π}?

- (Next lecture: How do we compute an optimal policy)

Monte-Carlo estimation

Collect a bunch of samples and take their average value

- Suppose we want to compute the expected age of people in the US

- Strategy: Sample N people at random. Suppose their ages are $a_1, a_2, \dots a_N$. Now "estimate" the expected age to be
$$E[A] \approx \frac{1}{N}(a_1 + a_2 + \ \dots a_N)$$

Reinforcement learning approaches

- **Model based RL**
  - First estimate T(s, a, s′) and R(s, a, s′)
  - Then use methods from MDP

- Data: $s_0, a_1, R_1, s_1, a_2, R_2, s_2, a_3, R_3, s_3, \ldots$
- $\hat{T}(s, a, s') = \frac{\#\ times\ (s,a,s')occurs}{\#\ times\ (s,a)\ occurs}$
- $\hat{R}(s, a, s')$ = mean(rewards obtained from state s taking action a and moving to s')

Reinforcement learning approaches

- **Model based RL**
  - First estimate T(s, a, s′) and R(s, a, s′)
  - Then use methods from MDP

- **Model free RL**
  - Could be wasteful to estimate T(s, a, s′) and R(s, a, s′)
  - Estimate V^{π} and Q^{π} directly
  - Estimate Q^{*} directly (policy extraction is trivial)

Policy evaluation

- For the policy π being used in the passive RL setting, how to estimate V^{π}?

Rest of lecture: Three methods (progressively improving):

- Direct policy evaluation
- "Better" policy evaluation
- Temporal difference learning

Direct policy evaluation

- Estimate V^{π} as follows

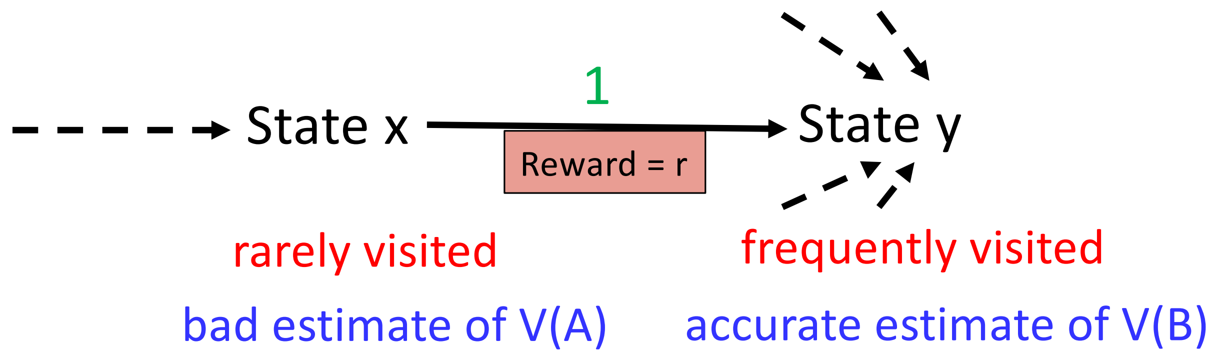
- Every time you visit a state, write down what the total reward across time turned out to be

- Average those samples

- *Monte-carlo estimates using samples of utility*

Direct policy evaluation

- Problem: This does **not** utilize the MDP structure, and can be wasteful

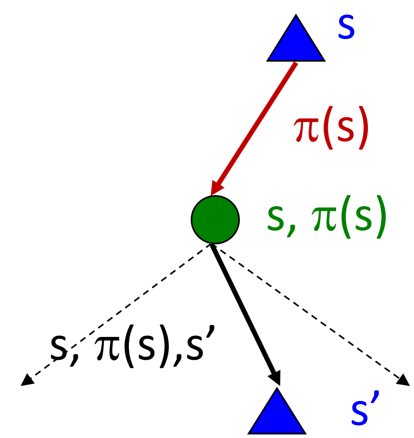


But, if we were to exploit the MDP structure: $V^{\pi}(x) = r + \gamma\, V^{\pi}(y)$

"Better" policy evaluation

- Recall value iteration (for given policy π)

$$V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^{\pi}_{k}(s')]$$



- Can we do this **without T, R** but just **via samples**?
  - Consider each time you are in state s and take action π(s)

$$\begin{aligned} sample_1 &= R(s,\pi(s),s'_1) + \gamma V^{\pi}_{k}(s'_1) \\ sample_2 &= R(s,\pi(s),s'_2) + \gamma V^{\pi}_{k}(s'_2) \\ &\ldots \\ sample_n &= R(s,\pi(s),s'_n) + \gamma V^{\pi}_{k}(s'_n) \end{aligned}$$

$$V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n}\sum_{i} sample_i$$

Challenges with "better" policy estimation

$$
\begin{aligned}
sample_1 &= R(s, \pi(s), s_1') + \gamma V_k^{\pi}(s_1') \\
&\dots \\
sample_n &= R(s, \pi(s), s_n') + \gamma V_k^{\pi}(s_n')
\end{aligned}
$$

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n} \sum_i sample_i$$

- Need to wait for n samples of (s,π(s)) for the update
- Any update to a different state V^{π}(s') in the meantime will use a stale estimate of V^{π}(s)
- Update for V^{π}(s) may be using stale estimates of V^{π}(s')

Reinterpreting sample average

$$V_{k+1}^{\pi}(s) \leftarrow \frac{1}{n}\sum_i sample_i$$

- Suppose we have average over n samples, and see a new sample, how do we update?

$$V_{k+1}^{\pi}(s) = \frac{n}{n+1} V_{k+1}^{\pi}(s) + \frac{1}{n+1} \text{sample}$$

- Interpolating between current estimate and new sample

Temporal Difference learning

- **Main idea:** Update V(s) each time we experience a transition (s, a, s′)

  Sample of V^{π}(s): $sample = R + \gamma\, V^{\pi}(s')$

  Update V^{π}(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

- Decreasing learning rate (α) towards zero leads to convergence