CS 446/ECE 449: Machine Learning

A. G. Schwing

University of Illinois at Urbana-Champaign, 2020

L28: Markov Decision Processes

Goals of this lecture

- Getting to know reinforcement learning
- Getting to know Markov decision processes

Recap: What have we talked about so far?

Pattern recognition and machine learning frameworks

Machine learning paradigms

- Discriminative learning and its applications
- Generative learning and its applications
- Now: Reinforcement learning and its applications

Machine learning paradigms:

- Discriminative learning:

$$p(y|x)$$

- Generative learning:

$$p(x)$$

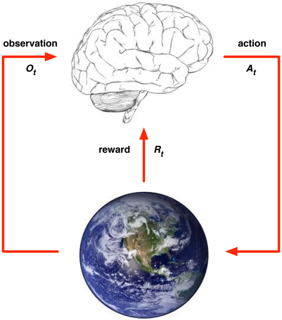
- Reinforcement learning (RL)

Examples

Reinforcement learning examples:

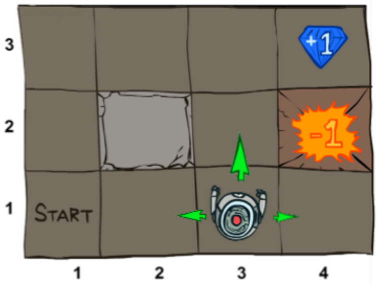
- Fly stunt manoeuvres in a helicopter
- Play Atari games
- Defeat the world champion at Go
- Manage investment portfolio
- Control a power station
- Make a humanoid robot walk

How are those tasks formulated?



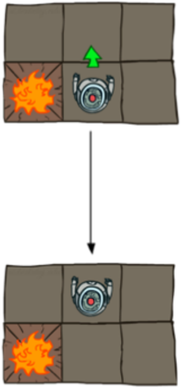
At each step t the agent

- Thinks/Knows about being in state s_t
- Performs action a_t
- Receives scalar reward $r_t \in \mathbb{R}$
- Finds itself in state s_{t+1}

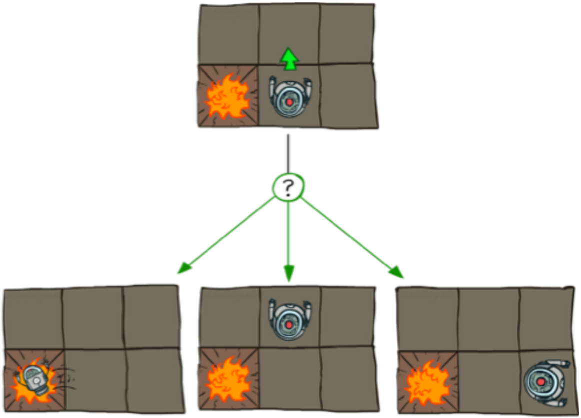

3
2
1
START
+1
-1
1
2
3
4

Settings:

Deterministic



Stochastic



Formally: Markov Decision Process (MDP)

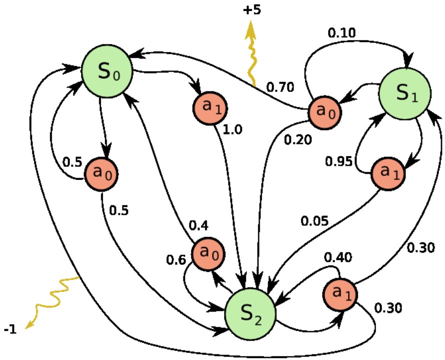
- A set of states $s \in \mathcal{S}$
- A set of actions $a \in \mathcal{A}_s$
- A transition probability $P(s' \mid s, a)$
- A reward function $R(s, a, s')$ (sometimes just $R(s)$ or $R(s')$)
- A start and maybe a terminal state

What is Markov about an MDP?

Given the present state, the future and the past are independent

$$
\begin{aligned}
&P(S_{t+1} = s' \mid S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = \ldots, S_0 = s_0) \\
&\quad = P(S_{t+1} = s' \mid S_t = s_t, A_t = a_t)
\end{aligned}
$$

Pictorial representation of MDP:



What makes RL different from other paradigms?

- No supervisor, only **reward** signal
- Delayed feedback
- Actions affect received data

Given a description of an MDP, what do we want?

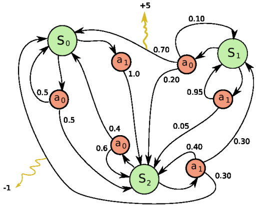
We want to perform actions according to a policy π^* so as to maximize the expected future reward.

How to encode the policy?

$$\pi(s) : \mathcal{S} \rightarrow \mathcal{A}_s$$

How to find the best policy π^*?

- Exhaustive search
- Policy iteration
- Value iteration



Exhaustive search for best policy π^*:

- How many policies?

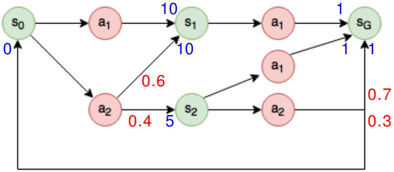
$$\prod_{s \in \mathcal{S}} |\mathcal{A}_s|$$

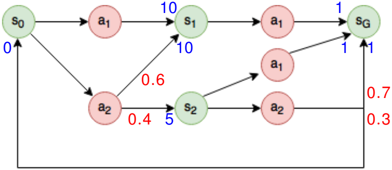
- How to evaluate quality of π? Compute expected future reward $V^{\pi}(s_0)$
- Choose policy π^* with largest expected future reward $V^{\pi^*}(s_0)$

Policy evaluation:

How to compute expected future reward $V^{\pi}(s)$ for a given policy?

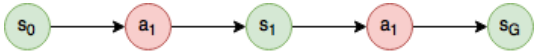
Example: rewards & transition probabilities





$\pi(s_0) = a_1,\ \pi(s_1) = a_1$

Policy graph:

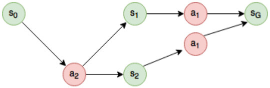


$V^\pi(s_1) = 1,\ V^\pi(s_0) = 11$

easy

$\pi(s_0) = a_2,\ \pi(s_2) = a_1$

Policy graph:



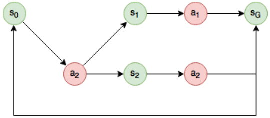
$V^\pi(s_1) = 1,\ V^\pi(s_2) = 1$

$$V^\pi(s_0) = .6 \cdot (10 + 1) + .4 \cdot (5 + 1) = 9$$

backpropagation

$\pi(s_0) = a_2,\ \pi(s_2) = a_2$

Policy graph:



$V^\pi(s_1) = 1$

$$V^\pi(s_2) = 0.7 \cdot 1 + 0.3 V^\pi(s_0)$$

$$V^\pi(s_0) = 0.4 \cdot (5 + V^\pi(s_2)) + 0.6 \cdot (10 + V^\pi(s_1))$$

linear system

Exhaustive search: for each policy π

Policy evaluation requires to solve linear system of equations:

$$\begin{aligned} V^\pi(s) &= 0 && \text{if } s \in \mathcal{G} \\ V^\pi(s) &= \textstyle\sum_{s' \in \mathcal{S}} P(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + V^\pi(s')\right] \end{aligned}$$

Expensive

Instead of solving system of linear equations use iterative refinement:

$$V_{i+1}^\pi(s) \leftarrow \sum_{s' \in \mathcal{S}} P(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + V_i^\pi(s')\right]$$

But searching over all policies is still expensive.

Policy iteration:

- Initialize policy π
- Repeat until policy π does not change
  - Solve system of equations (e.g., iteratively)

$$V^\pi(s) = \sum_{s' \in \mathcal{S}} P(s' \mid s, \pi(s)) \left[R(s, \pi(s), s') + V^\pi(s')\right]$$

  - Extract new policy π using

$$\pi(s) = \arg\max_{a \in \mathcal{A}_s} \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \left[R(s, a, s') + V^\pi(s')\right]$$

Can we directly find the optimal value function V^*?

Value Iteration:

- Changes search space (search over values, not over policies)
- Compute the resulting policy at the end

Bellman optimality principle:

$$V^*(s) = \max_{a \in \mathcal{A}_s} \underbrace{\sum_{s' \in \mathcal{S}} P(s' \mid s, a) \left[R(s, a, s') + V^*(s')\right]}_{Q^*(s,a)}$$

$$Q^*(s, a) = \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \left[R(s, a, s') + \max_{a' \in \mathcal{A}_{s'}} Q^*(s', a')\right]$$

Decoding policy:

$$\pi(s) = \arg\max_{a \in \mathcal{A}_s} \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \left[R(s, a, s') + V^*(s')\right]$$

$$\pi(s) = \arg\max_{a \in \mathcal{A}_s} Q^*(s, a)$$

Bellman optimality principle:

$$V^*(s) = \max_{a \in \mathcal{A}_s} \underbrace{\sum_{s' \in \mathcal{S}} P(s' \mid s, a) \left[R(s, a, s') + V^*(s')\right]}_{Q^*(s,a)}$$

How to solve for V^*?

- Solve via linear program (for very small MDPs)
- Iteratively refine

$$V_{i+1}(s) \leftarrow \max_{a \in \mathcal{A}_s} \sum_{s' \in \mathcal{S}} P(s' \mid s, a) \left[R(s, a, s') + V_i(s')\right]$$

Recap so far: Known MDP

- To compute V^*, Q^*, π^*: use **policy/value iteration or exhaustive**
- To evaluate fixed policy π: use policy evaluation

Quiz:

- What differentiates RL from supervised learning?
- What is a MDP?
- What differentiates policy iteration from policy evaluation?

Important topics of this lecture

- Getting a feeling for reinforcement learning
- Understanding how to use MDPs

What's next:

What to do if the MDP model is not known?