Introduction to MDPs

CSE599G: Deep Reinforcement Learning

University of Washington Seattle

March 28, 2018

Introduction to MDPs

- Formally describes a framework for Reinforcement Learning
- A very large fraction of problems can be modelled as MDPs
    - Most of robotics deals with MDPs (or POMDPs)
    - Chemical processes, power grids, manufacturing systems etc. in engineering
    - Inventory management, queues etc. in operations research
    - R2 (White '93) surveys a number of (old but relevant) applications of MDPs
- MDPs assume full observability and world without "intent". Some extensions to model these are POMDPs and Markov Games (more on these later)

Parts of an MDP

- Formally, MDP is a tuple: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \rho_0, \gamma, T \rangle$
  - $\mathcal{S}$ = states (joint positions in robot, concentrations in chemical reaction)
  - $\mathcal{A}$ = actions (motor torques, how much chemical to add)
  - $\mathcal{R}(s, a) \rightarrow \mathbb{R}$ is the “reward” function
  - $\mathcal{P} \equiv \mathbb{P}(s' | s, a)$ is the transition dynamics
  - ρ_0 = initial state distribution (i.e. state at time = 0)
  - T = horizon (how long does the MDP last)
  - γ = discount factor (immediate rewards are worth a bit more than future ones)

Markov Property

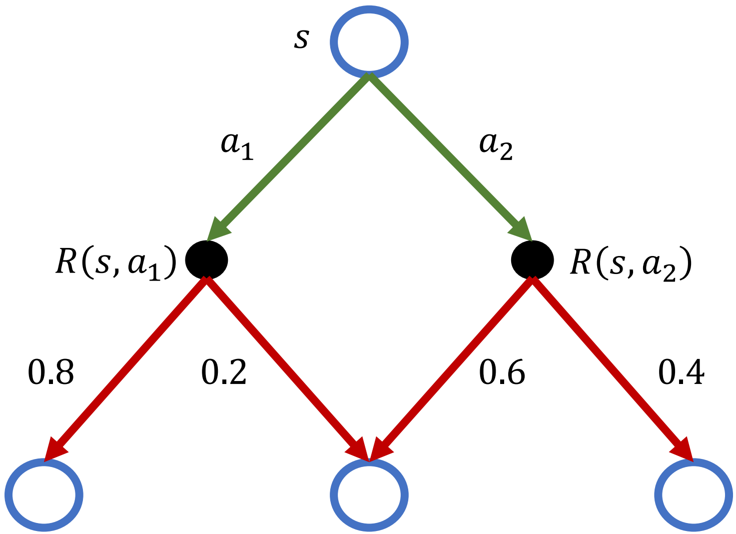
“ Future is independent of the past given the present”

- State is a sufficient statistic to summarize the system and to make decisions to control it.
- Let $H_t = (s_0, a_0, s_1, a_1, \dots s_t)$ denote the history till time t.
- Markov property implies that:

$$P(s_{t+1}|H_t, a_t) = P(s_{t+1}|s_t, a_t)$$

- For example, in Newtonian physics, state = positions + velocity

Closer look at the structure



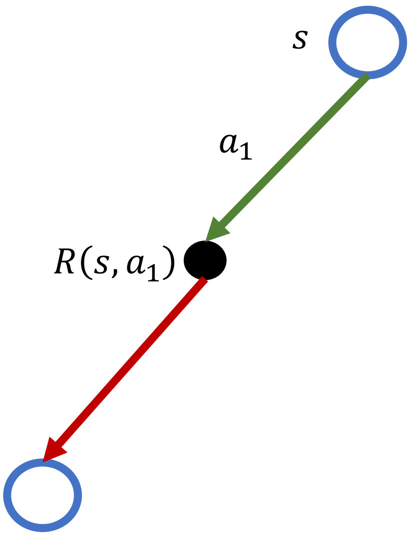
= state

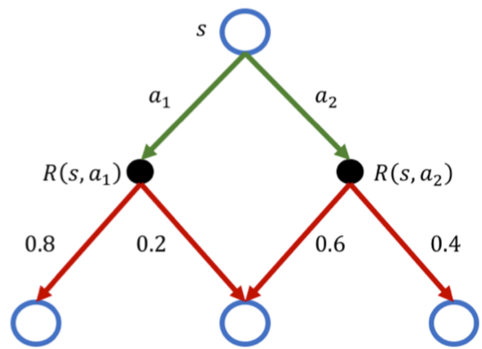
= (state, action)

→ = you choose

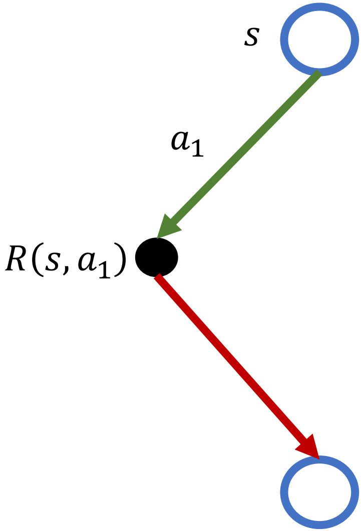
→ = environment dictates

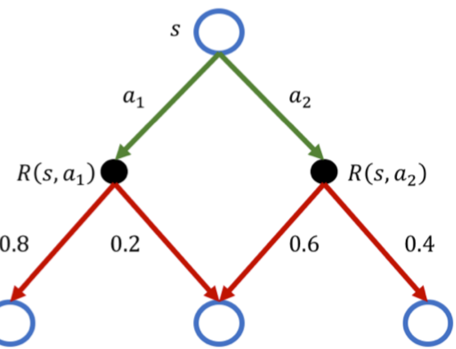
Closer look at the structure



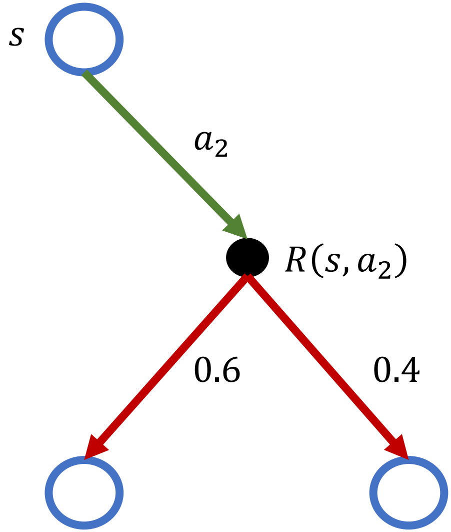


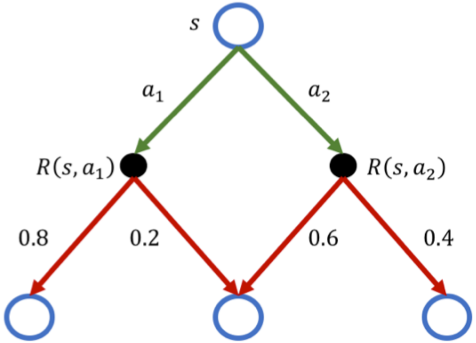
Closer look at the structure





Closer look at the structure





What is the goal for the agent?

- The agent's decision making rule is called "policy" (π)

$$\pi(a|s) = \mathbb{P}(A = a|S = s)$$

- We will mostly use a "randomized" decision making rule
- The policy fully defines the behavior of the agent.
- Fix the policy => MDP becomes a stochastic dynamical system that evolves in some way and generates rewards.
- Goal is to find policy such that the resulting dynamical system produces maximum reward (i.e. it behaves in a desirable way).

What is the goal for the agent?

Objective function for this problem:

$$\eta(\pi) = \mathbb{E}_{a_t \sim \pi(.|s_t), s_{t+1} \sim \mathbb{P}(.|s_t, a_t), s_0 \sim \rho_0}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)]$$

So that the optimal policy is defined as

$$\pi^* = argmax_\pi \, \eta(\pi)$$

Two important sub problems:

- Given a policy, determine how good it is (policy evaluation)
- Given a policy, make it better (policy improvement)

Examples

- **Robotic manipulation/locomotion:**
  States: joint positions, joint velocities, object poses, other sensing
  Actions: motor commands (say torques)
  Reward: task performance (e.g. distance to goal location)
- **Traffic management:**
  States: traffic light configuration, how many cars on each street etc.
  Actions: traffic light switching
  Reward: minimize waiting time, velocity changes, traffic jam etc.
- **Autonomous driving:**
  States: GPS location, on-board sensors etc.
  Actions: steering direction, accelerator
  Reward: distance to destination, various human preference settings
- **Cooking:**
  State: What’s on the pan and progress on recipe
  Action: Which ingredients to add, wait etc.
  Reward: happiness of significant other ☺

Types of MDPs

- **Time:** discrete time MDP vs continuous time MDP (requires PDEs)
- **States and Actions:** finite state/action MDPs vs real-valued states and actions (continuous MDPs).

  There is a distinction between "small + finite" vs "finite" (could be huge)
- **Dynamics:** Deterministic vs stochastic
- **Horizon:** finite horizon vs infinite horizon

Extensions to MDPs:

- POMDP: state not fully known(noisy sensors, unobservable quantities)
- Markov Games: $\mathbb{P}(s_{t+1}|s_t, a_t, c_t)$ where c_t is action by some other agent that has some "intent".

Value Functions

- How to measure long term performance of a policy?

  Run it on the environment. Too expensive, we would like data reuse.

- We will define a quantity that summarizes the long term performance, and attempt to learn this quantity.

- Define the value function of policy as:

$$\mathrm{V}^{\pi}(s, t) = \mathbb{E}_{a_{t'} \sim \pi(.|s_{t'}), s_{t'+1} \sim \mathbb{P}(.|s_{t'}, a_{t'})}[\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'}) \mid s_t = s]$$

- Depends on the policy, state, and time (in finite horizon case).

Value Functions

- Similarly, we can also define an action-value function

$$\mathrm{Q}^{\pi}(s,a,t) = \mathbb{E}_{a_{t'} \sim \pi(.|s_{t'}), s_{t'+1} \sim \mathbb{P}(.|s_{t'}, a_{t'})}[\sum_{t'=t}^{T} \gamma^{t'-t} R(s_{t'}, a_{t'}) \mid s_t = s, a_t = a]$$

- Note that every policy including π^* has an associated V^{π} and Q^{π}
- If we find the corresponding "optimal" value or action-value functions, we can obtain the optimal policy using one step look ahead.

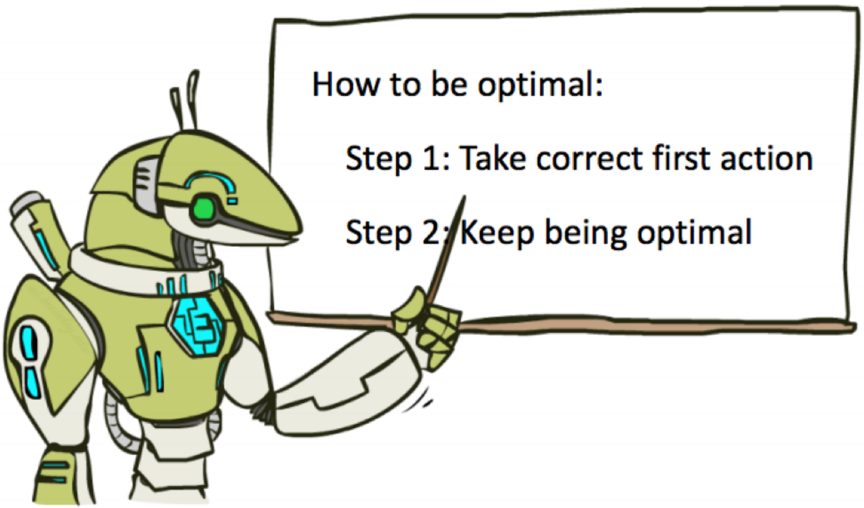
$$\pi^*(s) = argmax_a \, \mathbb{E}[\, R(s,a) + \gamma V^*(s')]$$

$$\pi^*(s) = argmax_a \, Q^*(s,a)$$

Value Functions

$$\pi^*(s) = argmax_a \, \mathbb{E}[\, R(s,a) + \gamma V^*(s')]$$

$$\pi^*(s) = argmax_a \, Q^*(s,a)$$



Value Functions

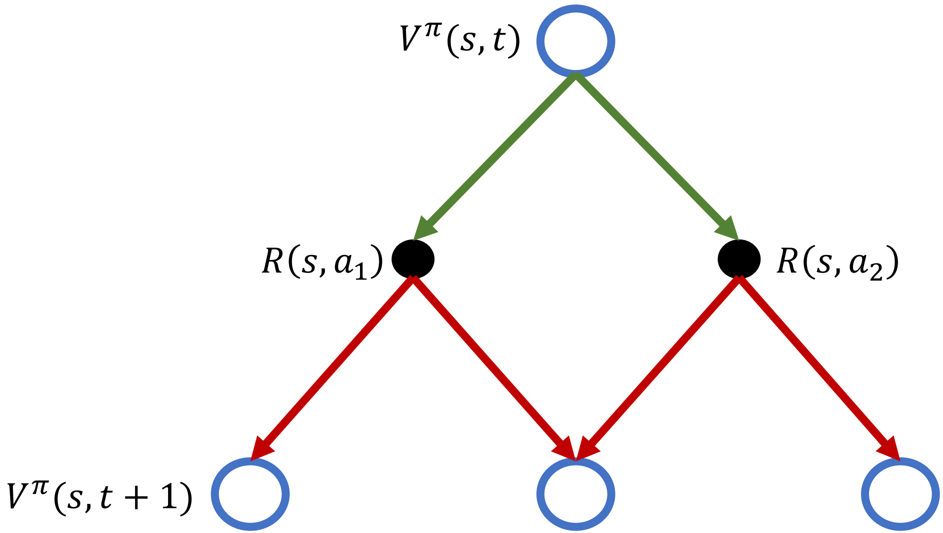
- Value functions help to abstract away the temporal nature of the problem, by summarizing the long term performance.

	State value	State-Action value
Given policy	$V^{\pi}(s)$	$Q^{\pi}(s,a)$
Optimal policy	$V^{*}(s)$	$Q^{*}(s,a)$

- But so far, we have only replaced one unknown with another unknown. Are there better ways to learn the value functions than Monte Carlo samples? Yes!

Bellman Recursion

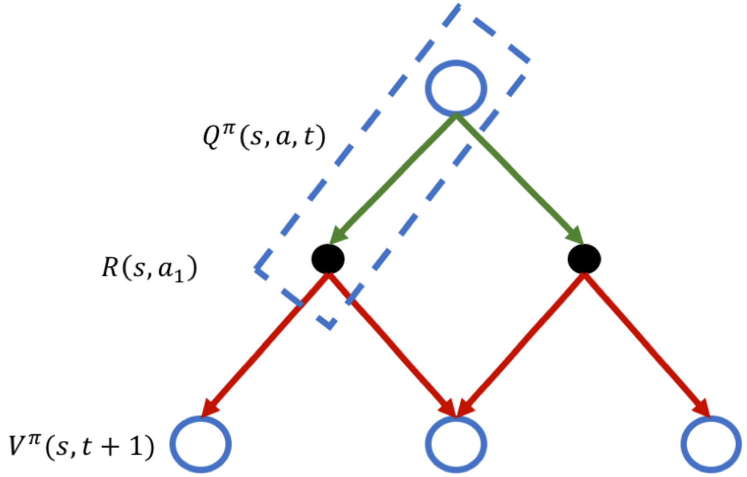
$$V^{\pi}(s,t) = \sum_{a} \pi(a|s) \left(R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^{\pi}(s',t+1) \right)$$



Bellman Recursion

$$Q^{\pi}(s,a,t) = R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) V^{\pi}(s',t+1)$$

$$Q^{\pi}(s,a,t) = R(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) \sum_{a'} \pi(a'|s') Q^{\pi}(s',a',t+1)$$



Bellman Recursion

- Recursive relationship due to the sequential nature of the problem!
- For infinite horizon problems, we can drop explicit dependence on time since the system will go to a stationary distribution.
- If we know values at some state, we can ”backup” this information to other states since values need to obey the recursion.
- This bootstrapping is the key to the efficiency of many RL methods.
- Next lecture: Dynamic programming

  Use the recursive relationship along with knowledge of dynamics to find the optimal policy and/or value function.