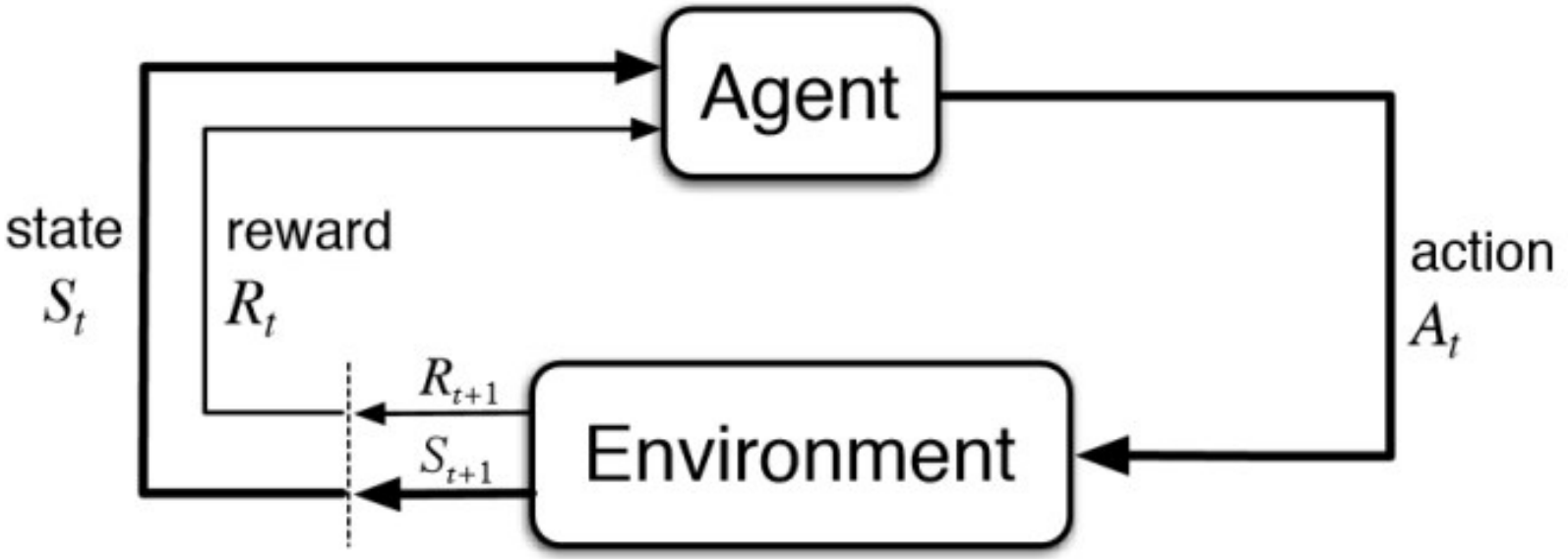
CS292F Statistical Foundations of Reinforcement Learning

Instructor: Yu-Xiang Wang

Spring 2021

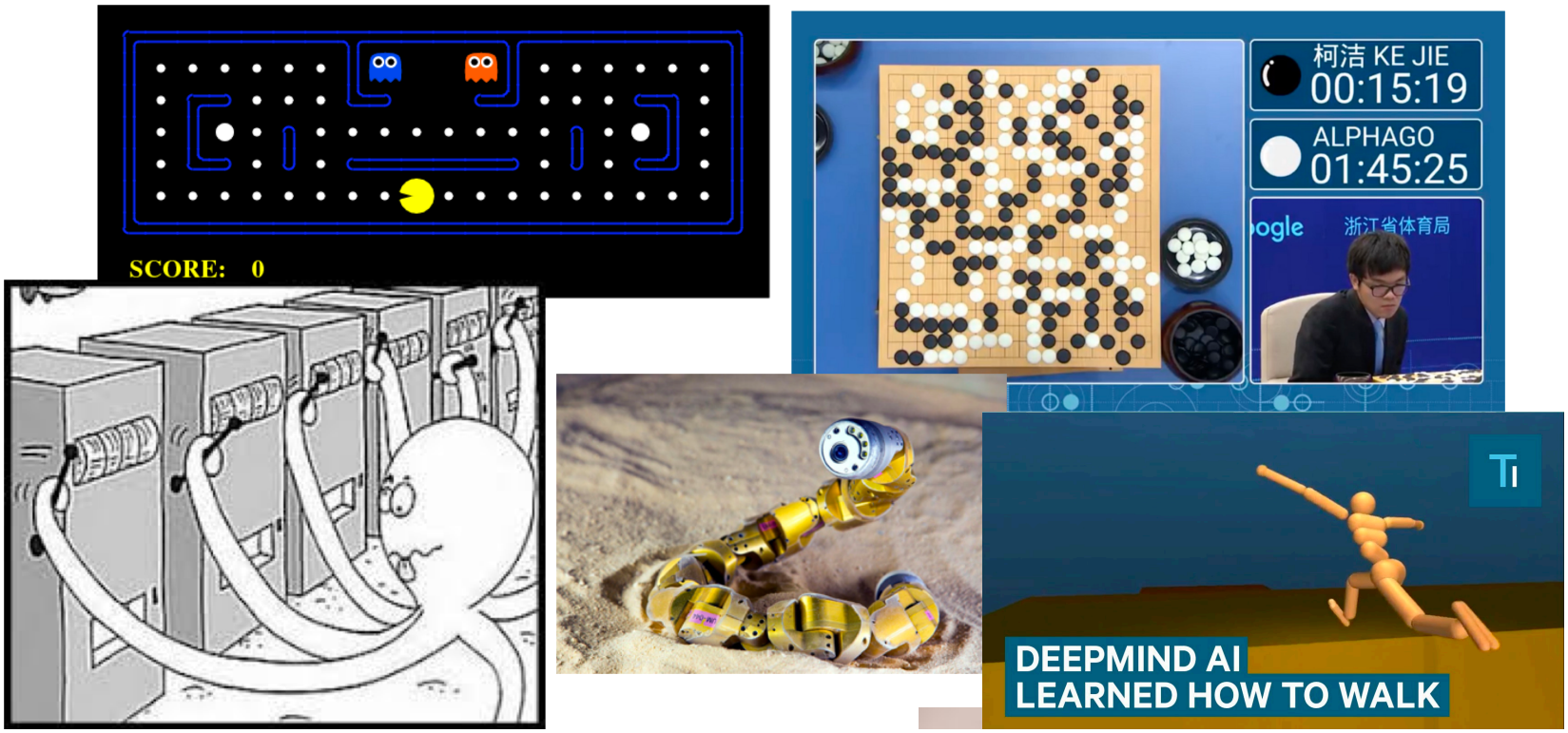
UC Santa Barbara

An RL agent learns interactively through the feedbacks of an environment.



- Learning how the world works (dynamics) and how to maximize the long-term reward (control) at the same time.

Reinforcement learning is among the hottest area of research in ML!



200+ papers on RL at NeurIPS'2019!

Applications of RL in the real life

- RL for robotics.
- RL for dialogue systems.
- RL for personalized medicine.
- RL for self-driving cars.
- RL for new material discovery.
- RL for sustainable energy.
- RL for feature-based dynamic pricing.
- RL for maximizing user satisfaction.
- RL for QoE optimization in networking
- …

This is a graduate level course on the theory of reinforcement learning.

- The statistical theory, in particular; though computational theory is just as relevant.

- You will learn:
  - How RL algorithms work and the intuition behind
  - How to prove sample complexity bounds for RL algorithms
  - What we know and what are the open in this research field.

- The course is not about:
  - Model a real-life problem as an RL problem (though we will have examples)
  - State-of-the-Art Deep RL algorithms (though it sets up the foundation for you to more easily self-study.)

Topics that I am planning to cover

1. Markov Decision Processes
2. RL algorithms
3. Online RL and Exploration
4. Offline RL theory
5. Function approximation

Reference materials:

1. Agarwal, Jiang, Kakade, Sun: https://rltheorybook.github.io/rltheorybook_AJKS.pdf
2. Relevant research papers
3. Sutton and Barto is a more entry-level introduction to RL.

Prerequisites

- The course would be tough for you if you do not have working knowledge in:
  - Linear algebra, basic calculus
  - Probability theory and statistics
  - Basic data structures and algorithms
  - writing mathematical proofs
  - writing simple code (in Python / Numpy)

- Advanced knowledge that will help:
  - Statistical machine learning
  - Dynamic programming
  - Linear / Convex optimization
  - Concentration inequalities

Homework 0 (uploaded to the course website) helps you with these pre-requisites.

Evaluation

- Lecture attendance: 10%
- Scribing: 10%
- Homework: 40%

- Project: 40% (Up to 3 students per team)
  - Basic project: Read an RL theory paper (or any related subject) and reproduce its proofs.
  - Advanced project: Develop an extension of existing results in a different setting / new application.
  - Special project: Apply RL to your own research.

Project milestones

1. Proposal: 5%
2. Midterm report: 10%
3. Final report: 10%
4. Project presentation: 15%

I will share a list of project ideas on Piazza soon.

Tentative schedule of this quarter

	Date	Lectures	Readings	Assignments
1	29-Mar	Introduction and MDP basics	AJKS Ch 1.1-1.2	HW0 out
2	31-Mar	Markov Decision Processes I	AJKS Ch 1.3-1.5	
3	5-Apr	Markov Decision Processes II	AJKS Ch 2	HW1 out
4	7-Apr	RL Algorithms I	SB Ch 5-6	
5	12-Apr	RL Algorithms II	AJKS Ch 9	HW0 due
6	14-Apr	Exploration: Multi-Armed Bandits	AJKS Ch 5.1	
7	19-Apr	Exploration: Tabular MDPs	AJKS Ch 6	HW2 out / Project proposal due
8	21-Apr	Exploration: Linear Bandits	AJKS Ch 5.2 / 5.3	
9	26-Apr	Exploration: Linear MDPs	AJKS Ch 7	HW1 due
10	28-Apr	Offline RL: OPE in bandits	TBA	
11	3-May	Offline RL: OPE in RL	TBA	HW3 out / Midterm report due
12	5-May	Offline RL: Offline Learning in Tabular MDPs	TBA	
13	10-May	Offline RL: Offline Learning via Uniform OPE	TBA	HW2 due
14	12-May	Offline RL: Policy improvement	TBA	
15	17-May	RL with general function approximation	TBA	
16	19-May	Advanced topic / Student presentation 1		
17	24-May	Advanced topic / Student presentation 2		HW3 due
18	26-May	Advanced topic / Student presentation 3		
19	31-May	No lecture, Memorial Day		
20	2-Jun	Advanced topic / Student presentation 4		Final project report due

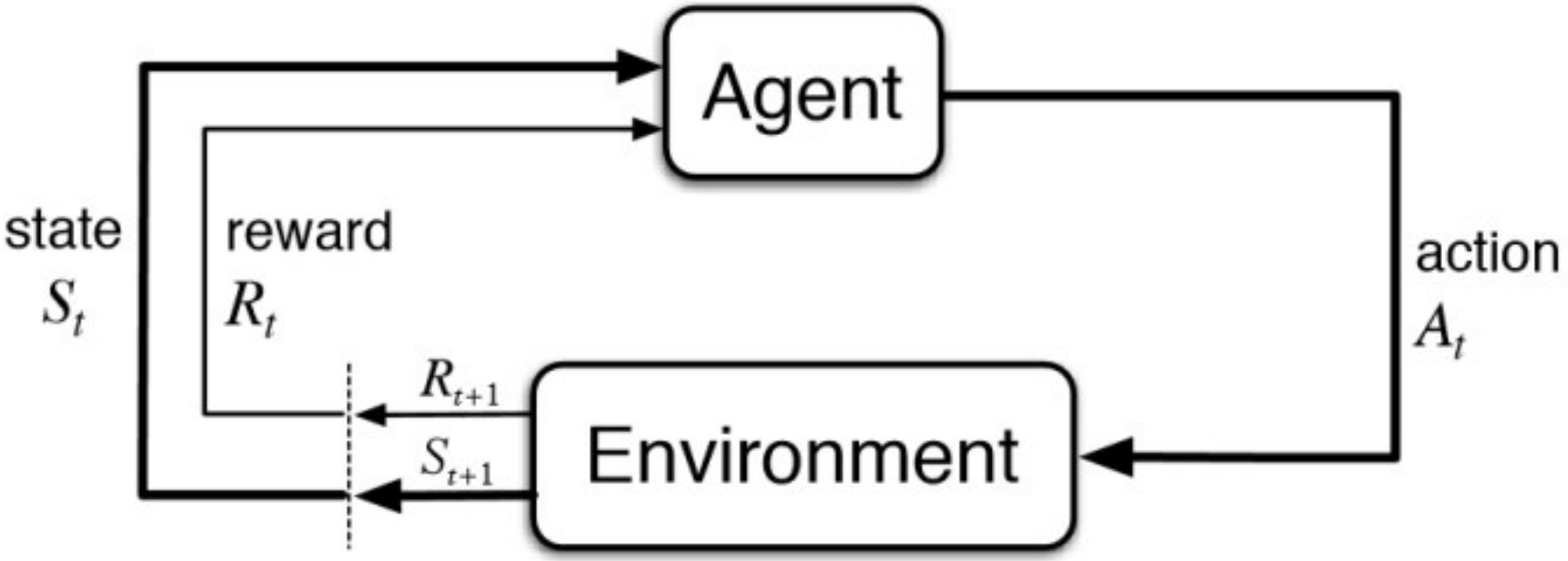
Remaining time today

- RL overview
  - Problem setup, goals, example
  - RL vs other machine learning settings
  - Online vs Offline RL

- Markov Decision Process
  - Notations
  - Bellman equations

Reinforcement learning problem setup

- State, Action, Reward
- Unknown reward function. unknown state-



Reinforcement learning problem setup

- State, Action, Reward and Observation

$$S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O}$$

- Policy: $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - When the state is observable:
  - Or when the state is not observable

$$\pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A}$$

- Learn the best policy that maximizes the expected reward
  - Finite horizon (episodic) RL: $\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{T} R_t]$

T: horizon

  - Infinite horizon RL:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} R_t]$$

γ: discount factor

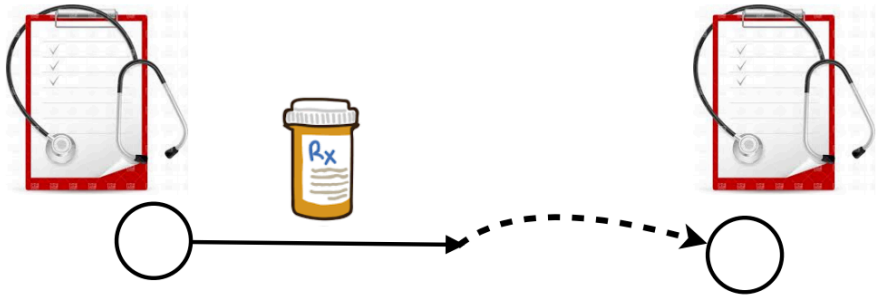
RL for robot control

- States: The physical world, e.g., location/speed/acceleration and so on.
- Observations: camera images, joint angles
- Actions: joint torques
- Rewards: stay balanced, navigate to target locations, serve and protect humans, etc.

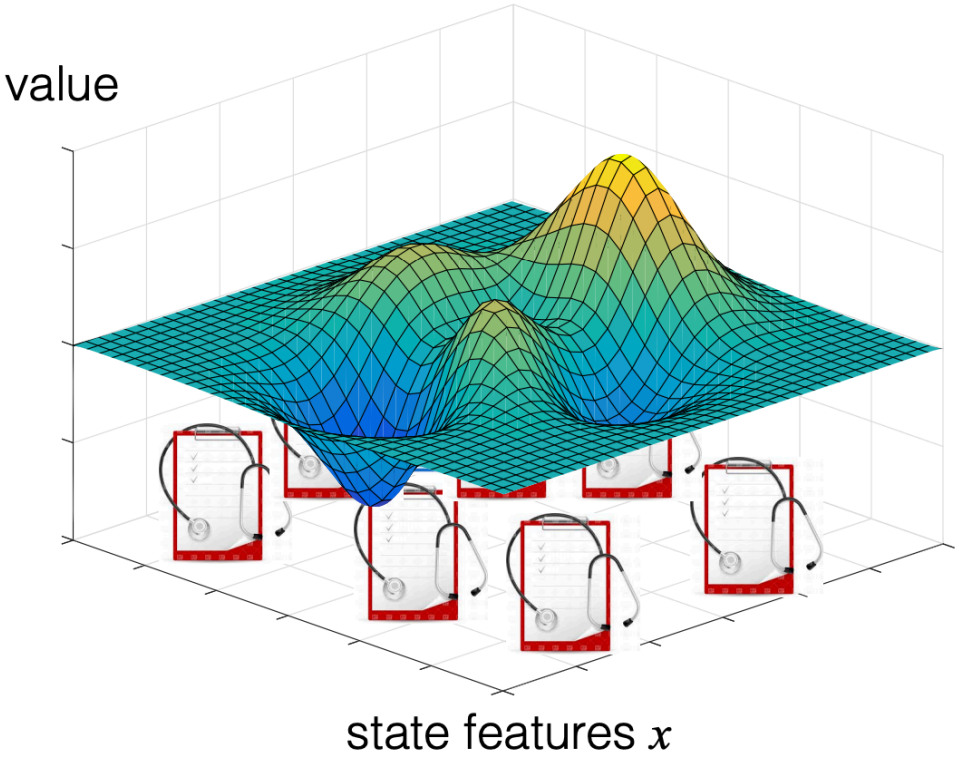
RL for Inventory Management

- State: Inventory level, customer demand, competitor's inventory
- Observations: current inventory levels and sales history
- Actions: amount of each item to purchase
- Rewards: profit

RL for Adaptive medical treatment



- State: diagnosis
- Action: treatment
- Reward: progress in recovery



(example / illustration due to Nan Jiang)

Reinforcement learning is, arguably, the most general AI framework.

- RL: State, Action, Reward, Nothing is known.

- Simplified RL models:
  - iid state → Contextual bandits
  - No state, finite action → Multi-arm bandits
  - iid state/action → Supervised Learning
  - Known dynamics / reward → Markov Decision Processes (Control/Cybernetics)
  - No reward / Unknown dynamics → System Identification

Example: Supervised learning vs RL in movie recommendation

- Bob is described by a feature vector
  - s =[Previous movies watched / Rating / Written reviews]

- Supervised learning predicts how likely Bob will click on “aliens vs predators”

- Reinforcement learning aims at controlling Bob
  - So in the future, Bob will develop a taste for “aliens vs predators” (e.g., from having watched “aliens” and “predators” both).

Reinforcement learning is very challenging

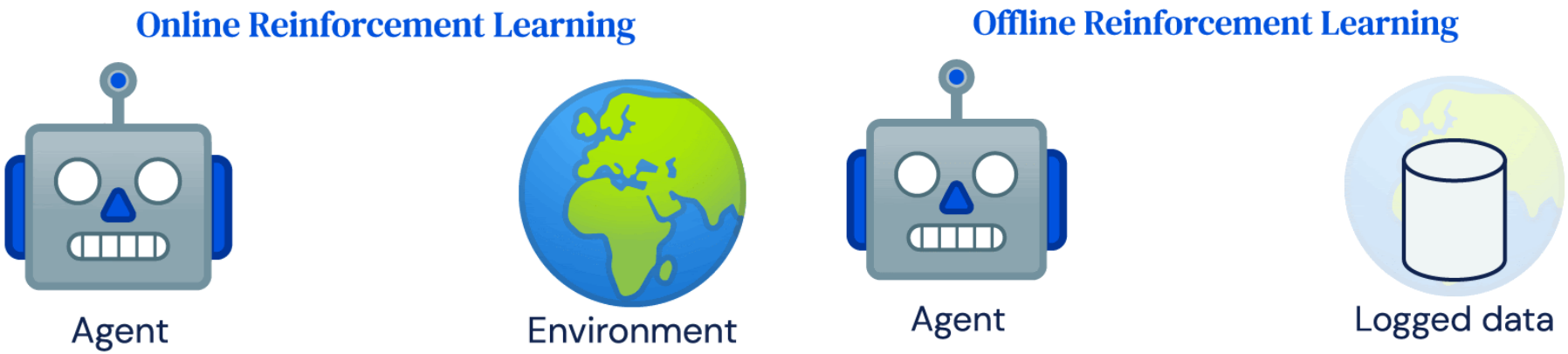
- The agent needs to:
  - Learn the state-transitions ----- How the world works
  - Learning the costs / rewards ----- Cost of actions
  - Learning how to search ----- Come up with a good strategy

- All at the same time

Let us tackle different aspects of the RL problem one at a time

- **Markov Decision Processes: (Lecture 1-3)**
  - Dynamics are given no need to learn. planning only.
- RL algorithms: (Lecture 4-5)
  - Ideas on how to use "sampled" transitions / trials-and-errors to learn and plan at the same time (without touching upon exploration).
- Strategic exploration: (Lecture 6- 9)
  - Bandits: Explore-Exploit in simple settings
  - RL: Explore-Exploit in Learning MDPs

Online RL vs Offline RL



Exploration is often **expensive**, **unsafe**, **unethical** or **illegal** in practice, e.g., in self-driving cars, or in medical applications.

Can we learn a policy from already **logged interaction data**?

***Lecture 10-14 will be about offline RL.**

Let’s start by formulating Markov Decision processes (MDP).

- Infinite horizon / discounted setting

Reward function and Value functions

- Immediate reward function r(s,a,s')
  - expected **immediate** reward

$$r(s,a,s') = \mathbb{E}[R_1 | S_1 = s, A_1 = a, S_2 = s']$$

$$r^{\pi}(s) = \mathbb{E}_{a \sim \pi(a|s)}[R_1 | S_1 = s]$$

- state value function: $V^{\pi}(s)$
  - expected **long-term** return when starting in *s* and following π

$$V^{\pi}(s) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s]$$

- state-action value function: $Q^{\pi}(s,a)$
  - expected **long-term** return when starting in *s*, performing *a*, and following π

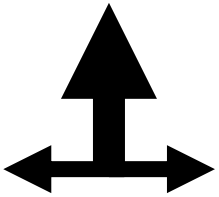
$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s, A_1 = a]$$

Example: Frozen lake.

			+1
			-1
START			

actions: UP, DOWN, LEFT, RIGHT

UP e.g.,



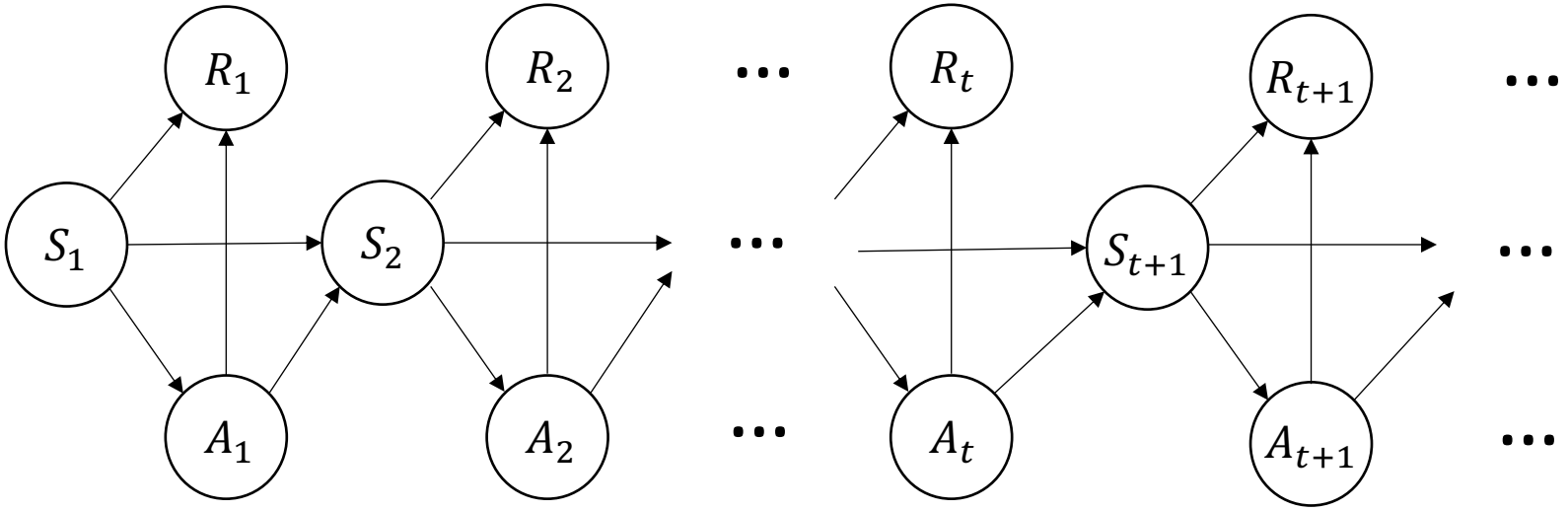
State-transitions with action **UP**:

80% move up
10% move left
10% move right

*If you bump into a wall, you stay where you are.

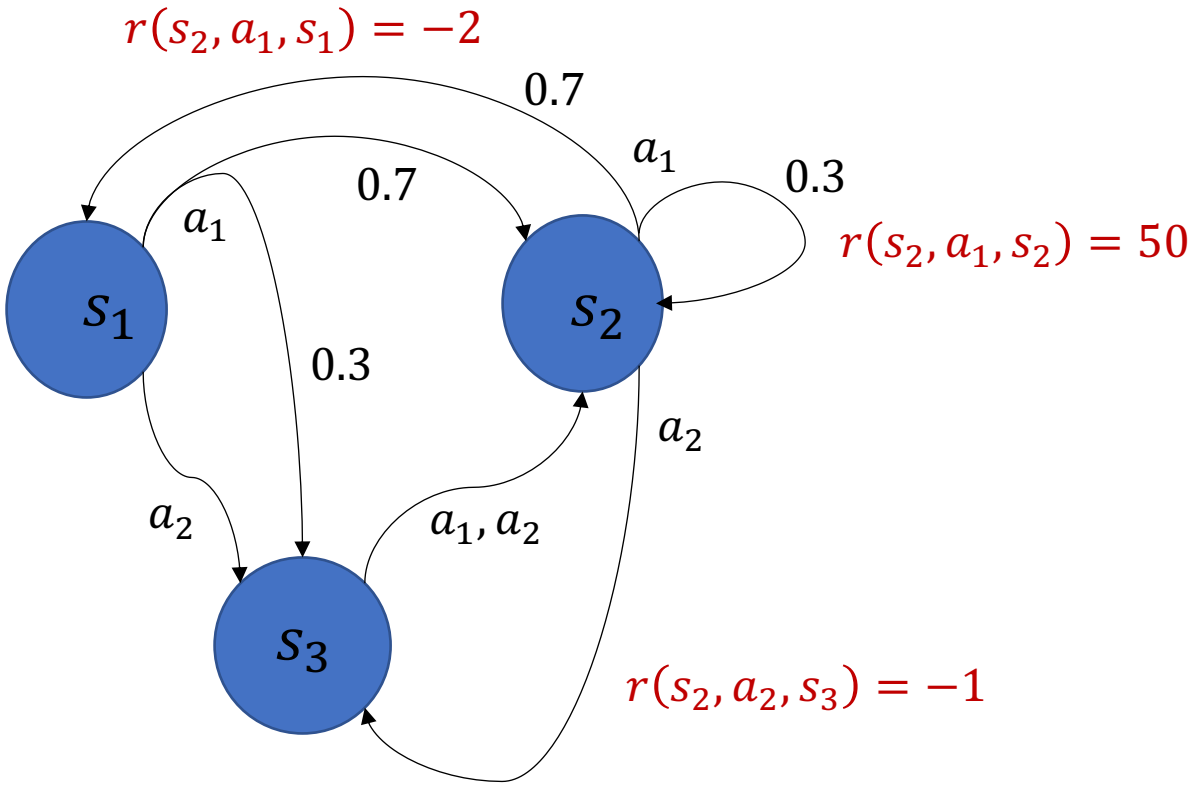
- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step
- Finite horizon or infinite horizon?
- What is a good policy?

Parameters of an MDP are factorizations of the joint distribution



- Initial state distribution
- Transition dynamics
- Reward distribution

State-space diagram representation of an MDP: An example with 3 states and 2 actions.



* The reward can be associated with only the state s' you transition into.
* Or the state that you transition from s and the action a you take.
* Or all three at the same time.

Optimal value function and the MDP planning problem

Policy, Stationary policy, Deterministic policy

- General policy could depend on the entire history

- Stationary policy

- Stationary, Deterministic policy

Two surprising facts about MDPs

1. It suffices to consider stationary / deterministic policies.

2. There exists a stationary / deterministic policy that is optimal simultaneously for all initial state distributions.

Proposition: It suffices to consider stationary policies.

- Proof:

Proposition: It suffices to consider stationary policies.

- Proof (continue):

Proposition: It suffices to consider stationary policies.

- Proof (continue):

Corollary: There exists an optimal policy that is stationary.

Bellman equations – the fundamental equations of MDP and RL

- For stationary policies there is an alternative, recursive and more useful way of defining the V-function and Q function

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi}(s')] = \sum_{a} \pi(a|s) Q^{\pi}(s,a)$$

- Exercise:
  - Prove Bellman equation from the (first principle) definition.
  - Write down the Bellman equation using Q function alone.

$$Q^{\pi}(s,a) = ?$$

Deriving Bellman Equation for stationary policies

Bellman optimality equations characterizes the optimal policy

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^*(s')]$$

- system of n non-linear equations
- solve for V*(s)
- easy to extract the optimal policy

- having Q*(s,a) makes it even simpler

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

More on this in the next lecture!