Reinforcement Learning & Multi-Armed Bandits

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning

Fall 2022

Today

- Overview of reinforcement learning and this course
- Multi-armed bandits
    - Problem statement
    - Baseline approach 1: pure exploration
    - Baseline approach 2: pure greedy

Course staff introductions

- **Instructors:** Lucas Janson and Sham Kakade
- **TFs:** Daniel Garces, Kuanhao Jiang, Yanke Song
- **CAs:** Alex Cai, Howie Guo, Angela Li, Richard Qiu, Eric Shen, Lara Zeng, Saba Zerefa
- Homework 0 goes out today

Course objectives:

- We seek the students to obtain fundamental and working knowledge of RL: the algorithms, aspects of their analysis, and the practice
- Lectures will be math heavy; all HWs have programming components.
- HW0 +HW1-HW4 + Final
- HW0 goes out today
    - HW0 is review of helpful background!
- We will have an “embedded ethics lecture” + assignment
- The class will be challenging, and we hope you will enjoy it!

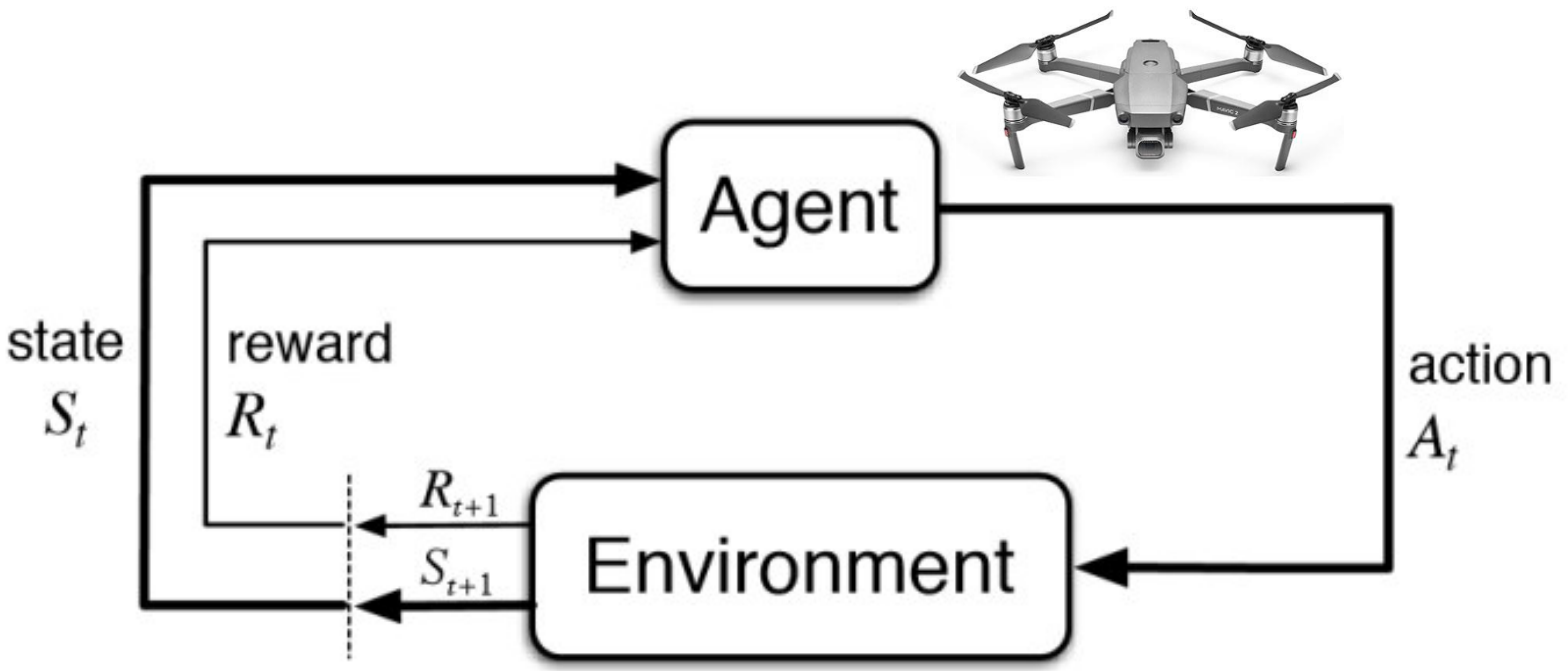
Course logistics

All policies are stated on the course website:

https://shamulent.github.io/CS_Stat184_Fall22.html

- Our policies seek consistency among all the students.
- Communication: please only use Ed to contact us
- Late policy (basically): you have 96 cumulative hours of late time.
  - *Please use this to plan for unforeseen circumstances.*
- Regrading: ask us in writing on Ed in a week

What is Reinforcement Learning?



Goal: optimize some long-term objective function

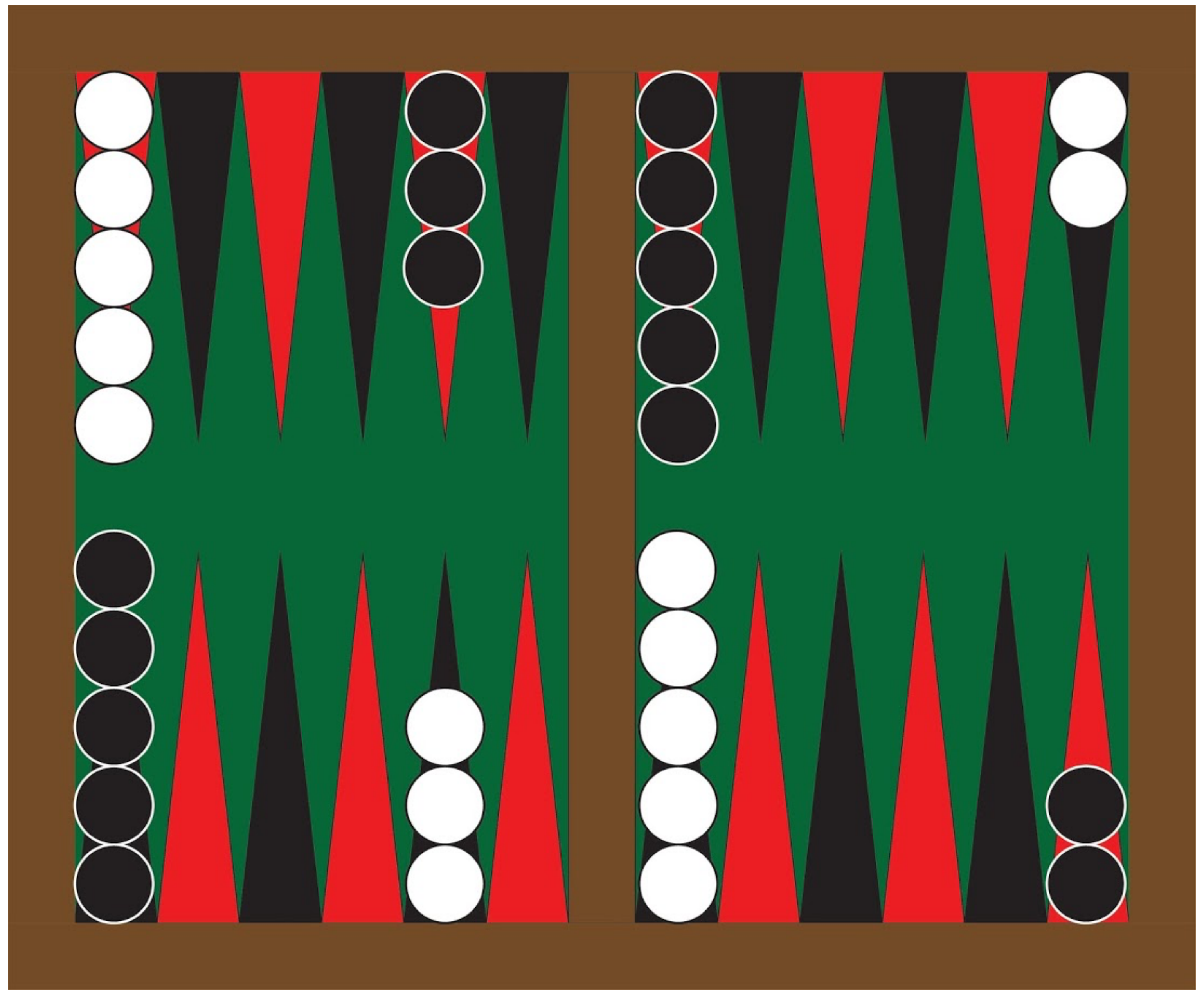
Humans do this all the time:

past experience (input) informs every decision (output) to achieve some end (goal)

Different from other ML (supervised/unsupervised learning) b/c **interactive**

Flashy Successes of RL so far: games

Backgammon



TD GAMMON [Tesauro 95]

Go

[AlphaZero, Silver et.al, 17]

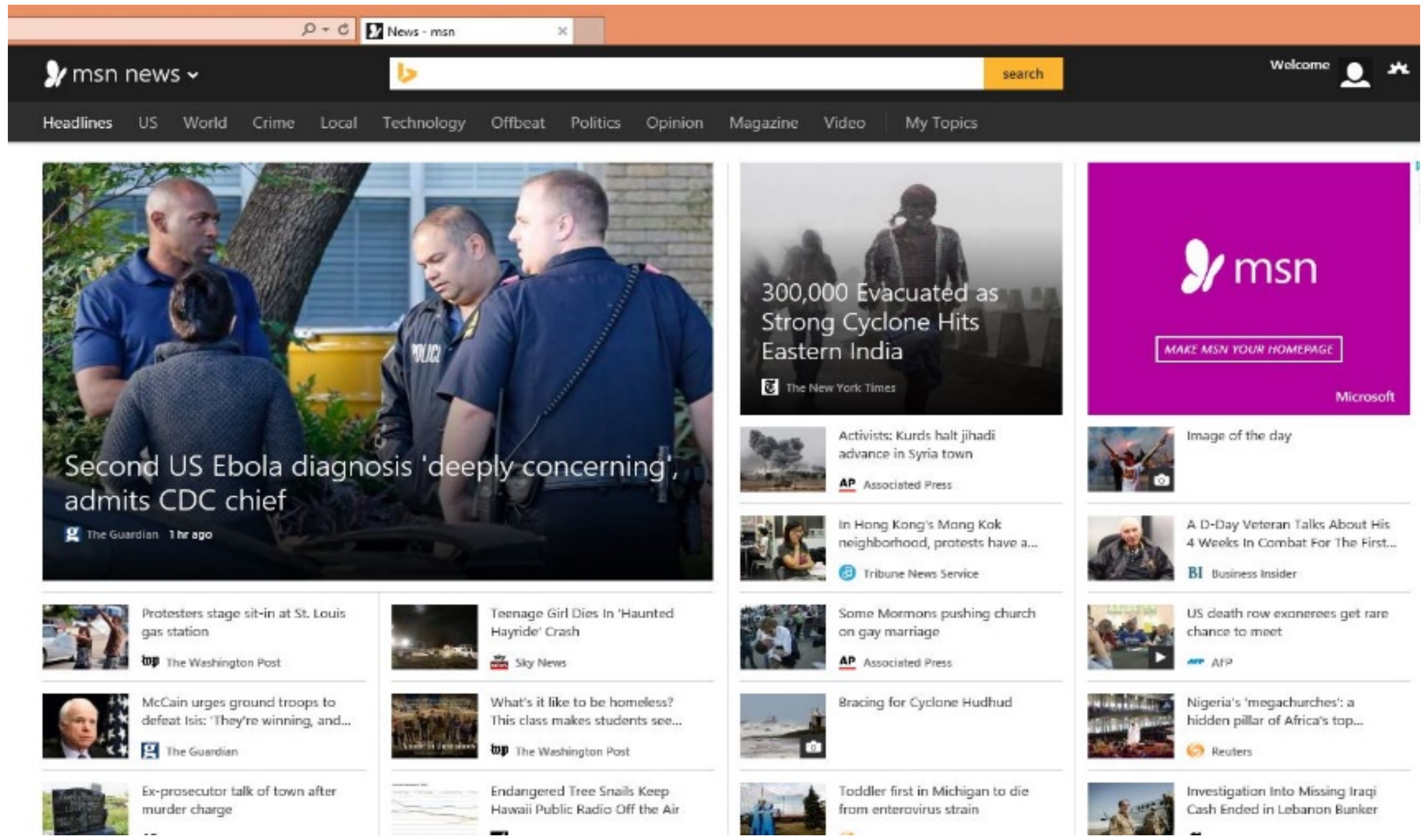
Dota

[OpenAI Five, 18]

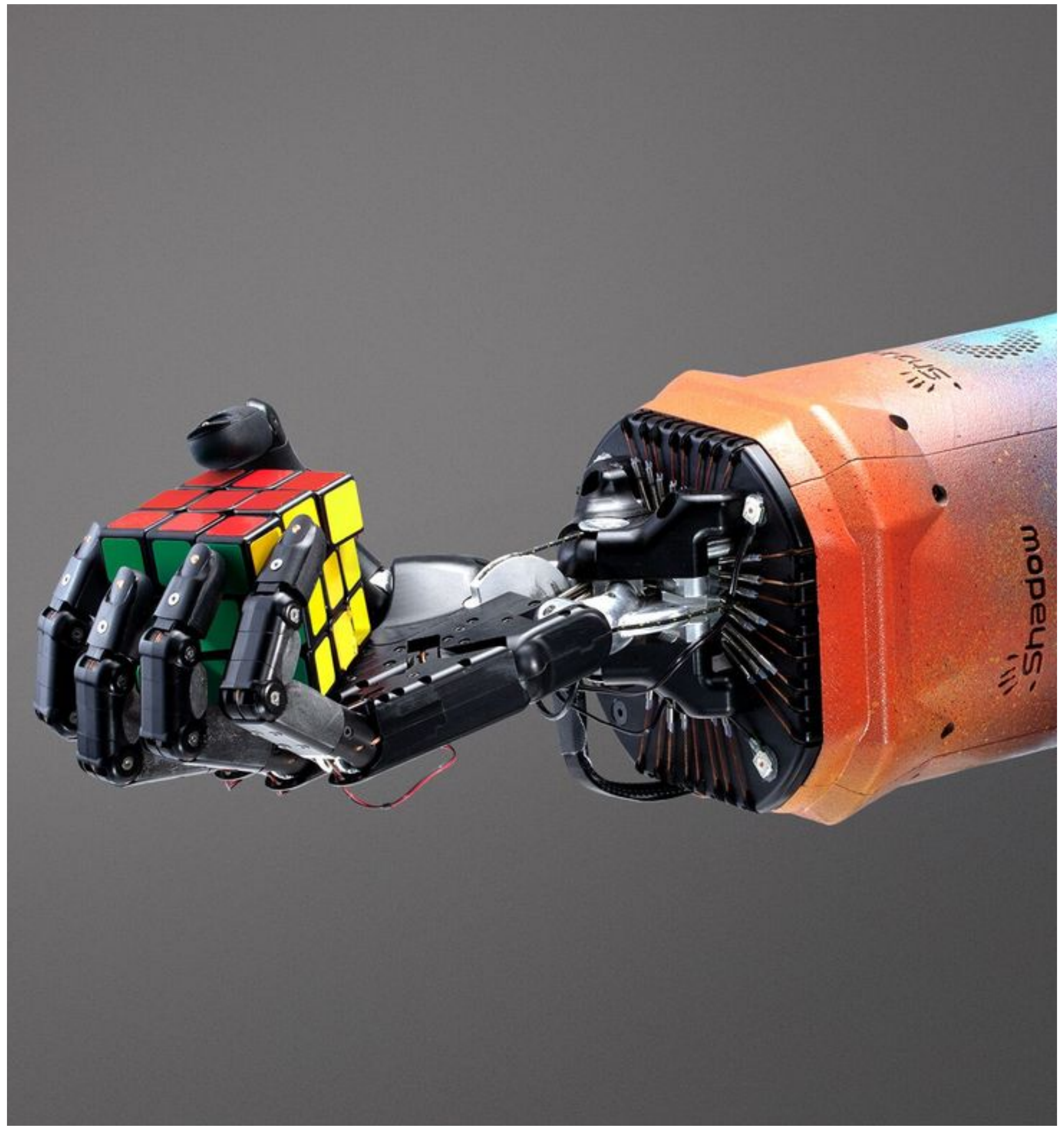
Reinforcement Learning in Real World:

Mobile health

Online advertising



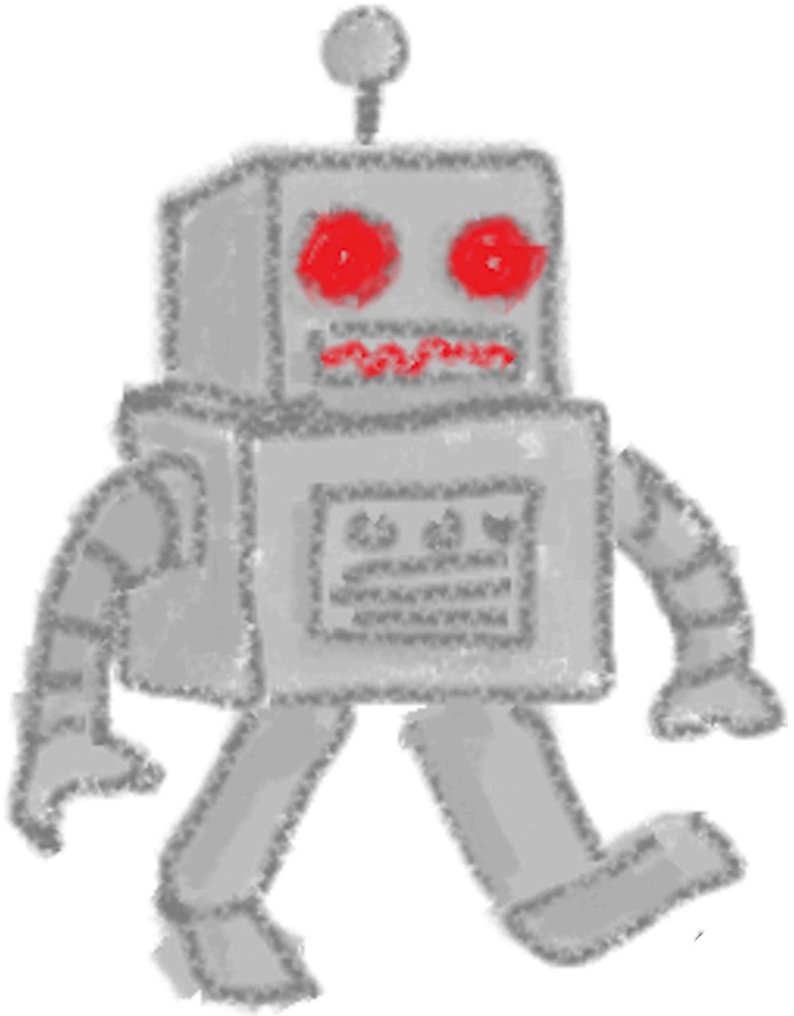
Robotic manipulation



Mathematical framework in which RL happens:

Markov Decision Process

Agent



Environment

$\pi(s) \rightarrow a$

Policy: determine **action** based on **state**

Multiple Steps

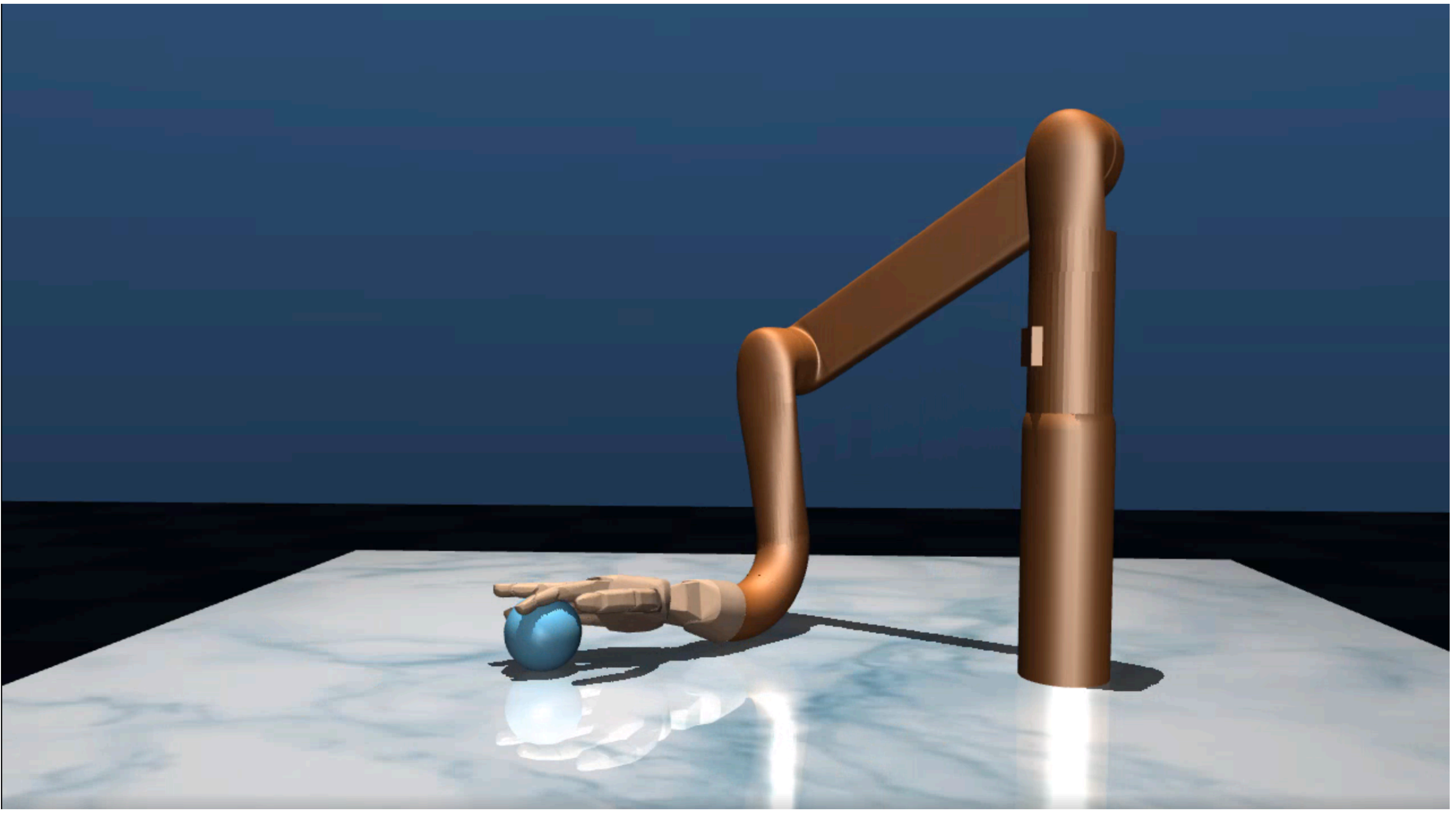
Send **reward** and **next state** from a Markovian transition dynamics

$$r(s,a), \quad s' \sim P(\cdot \,|\, s,a)$$

Policy π is what is under agent's control

Reinforcement Learning = updating π from initial π_0 towards (hopefully) optimal $\pi^\star$

Example:
robot hand needs to pick the ball and hold it in a goal (x,y,z) position



State s: robot configuration (e.g., joint angles) and the ball's position

Action a: Torque on joints in arm & fingers

Transition $s' \sim P(\cdot \mid s, a)$: physics + some noise

policy $\pi(s)$: a function mapping from robot state to action (i.e., torque)

Cost $c(s, a)$: torque magnitude + dist to goal

$$\pi^\star = \arg\min_{\pi} \mathbb{E}\left[c(s_0, a_0) + \gamma c(s_1, a_1) + \gamma^2 c(s_2, a_2) + \gamma^3 c(s_3, a_3) + \ldots\ldots \,\middle|\, a_h = \pi(s_h), s_{h+1} \sim P(\cdot \mid s_h, a_h) \right]$$

Fundamental challenges of RL

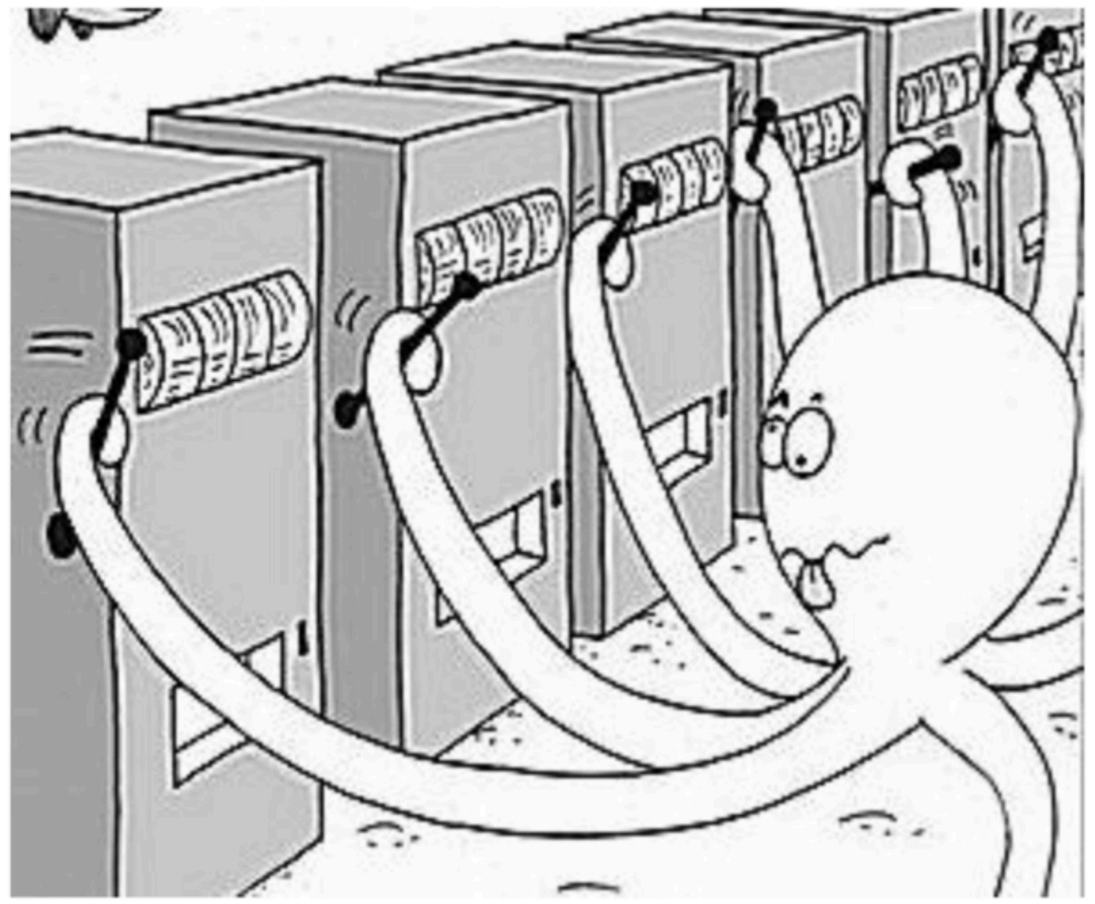
1. **Learning the environment:** learn online without sacrificing too much reward
   - Exploration-Exploitation tradeoff
   - Aspects of <u>experimental design</u>: what data to collect to *learn fastest* [no reward]
   - Aspects of <u>supervised learning</u>: *predict* outcome of an action [i.i.d. data]
2. **Optimizing policy**: in *known* complex environment, hard to find best policy
   - Aspects of <u>control theory</u>: how to act optimally in complex environment [known]

This course: addressing these challenges in increasingly complex environments

Today + next 6 lectures isolate **challenge 1 (learning the environment)**
(Multi-armed) Bandits: very simple but unknown environment

Today

- Overview of reinforcement learning and this course
- Multi-armed bandits
  - Problem statement
  - Baseline approach 1: pure exploration
  - Baseline approach 2: pure greedy



Intro to Multi-armed bandits (MAB)

Setting:

We have K many arms; label them $1, \ldots, K$

Each arm has a <u>unknown</u> reward distribution, i.e., $\nu_k \in \Delta([0,1])$,

w/ mean $\mu_k = \mathbb{E}_{r \sim \nu_k}[r]$

Example: ν_k is a Bernoulli distribution w/ mean $\mu_k = \mathbb{P}_{r \sim \nu_k}(r = 1)$

Every time we pull arm k, we observe an i.i.d reward $r = \begin{cases} 1 & \text{w/ prob } \mu_k \\ 0 & \text{w/ prob } 1 - \mu_k \end{cases}$

Application: online advertising

Arms correspond to Ads

Reward is 1 if user clicks on ad

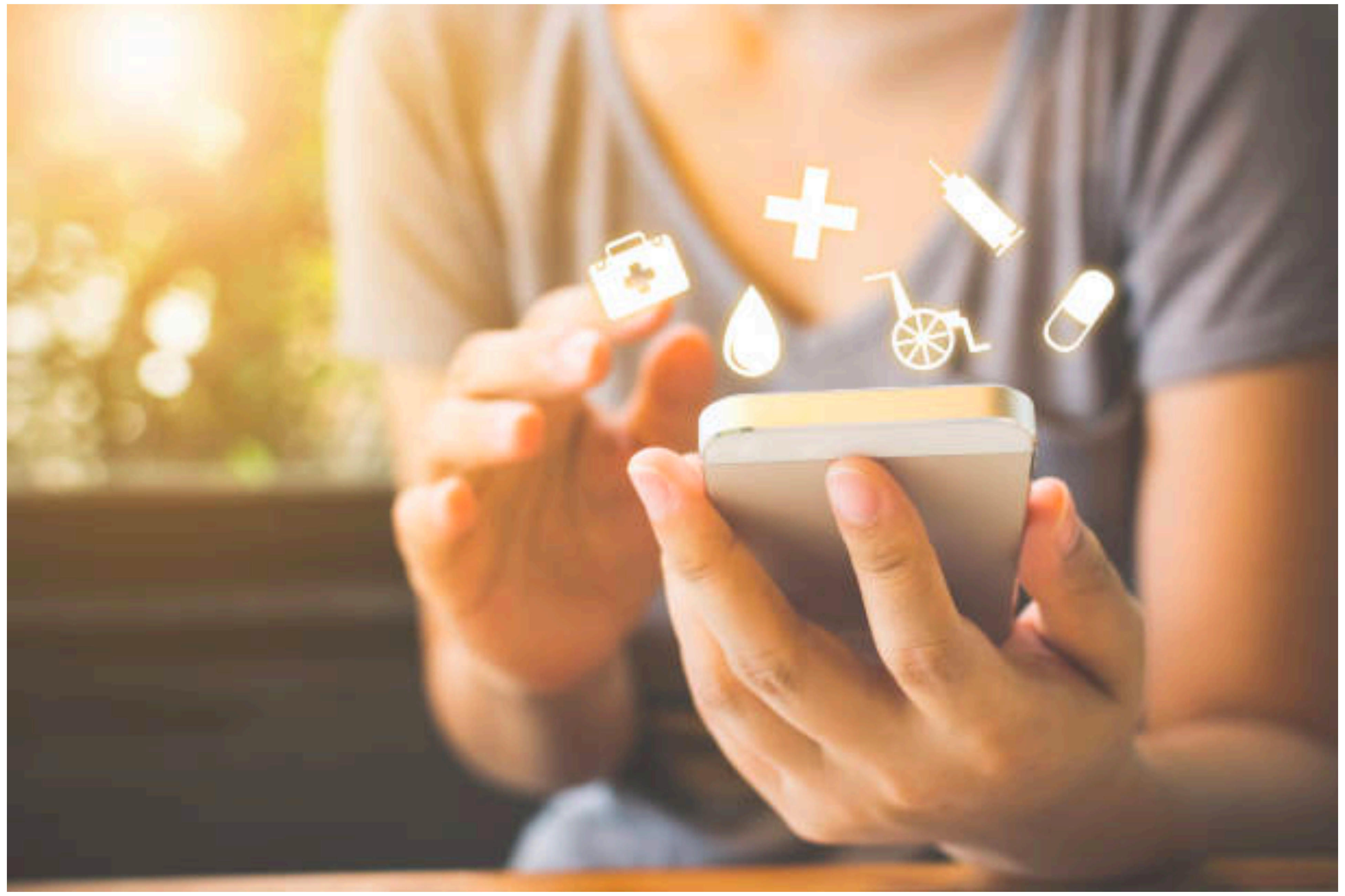
A learning system aims to maximize clicks in the long run:

1. **Try** an Ad (pull an arm)

2. **Observe** if it is clicked (see a zero-one **reward**)

3. **Update**: Decide what ad to recommend for next round

Application: mobile health



Arms correspond to messages sent to users

Reward is, e.g., 1 if user exercised
after seeing message

A learning system aims to maximize fitness in the long run:

1. **Send** an message (pull an arm)

2. **Observe** if user exercises (see a zero-one **reward**)

3. **Update**: Decide what message to send next round

MAB sequential process

More formally, we have the following interactive learning process:

For $t = 0 \rightarrow T - 1$

(# based on historical information)

1. Learner pulls arm $a_t \in \{1, \ldots, K\}$

2. Learner observes an i.i.d reward $r_t \sim \nu_{a_t}$ of arm a_t

Note: each iteration, we do not observe rewards of arms that we did not try

Note: there is no state s; rewards from a given arm are i.i.d. (data NOT i.i.d.!)

MAB learning objective

Optimal policy when reward distributions known is trivial: $\mu^\star := \max_{k \in [K]} \mu_k$

$$\text{Regret}_T = T\mu^\star - \sum_{t=0}^{T-1} \mu_{a_t}$$

Total expected reward if we pulled best arm over T rounds

Total expected reward of the arms we pulled over T rounds

Goal: want Regret_T as small as possible

Why is MAB hard?

Exploration-Exploitation Tradeoff:

Every round, we need to ask ourselves:

Should we pull the arm that currently appears best now (**exploit**; immediate payoff)?
Or pull another arm, in order to potentially learn it is better (**explore**; payoff later)?

Today

- Overview of reinforcement learning and this course
- Multi-armed bandits
    - Problem statement
    - Baseline approach 1: pure exploration
    - Baseline approach 2: pure greedy

Naive baseline: pure exploration

Algorithm: at each round choose an arm uniformly at random from among $\{1, \ldots, K\}$

Clearly no learning taking place!

$$\mathbb{E}[\text{Regret}_T] = \mathbb{E}\left[T\mu^\star - \sum_{t=0}^{T-1} \mu_{a_t}\right] = T\left(\mu^\star - \bar{\mu}\right) > 0$$

$$\bar{\mu} = \frac{1}{K}\sum_{k=1}^{K} \mu_k$$

Today

- Overview of reinforcement learning and this course
- Multi-armed bandits
    - Problem statement
    - Baseline approach 1: pure exploration
    - Baseline approach 2: pure greedy

Baseline: pure greedy

Algorithm: try each arm once, and then commit to the one that has the **highest observed** reward

Q: what could go wrong?

A bad arm (i.e., low μ_k) may generate a high reward by chance (or vice versa)!

Example: pure greedy

More concretely, let's say we have two arms:

Reward distribution for arm 1: ν_1 = Bernoulli($\mu_1 = 0.6$)

Reward distribution for arm 2: ν_2 = Bernoulli($\mu_2 = 0.4$)

Clearly the first arm is better!

$(1 - \mu_1)\mu_2 = (1 - 0.6) \times 0.4$

First $a_0 = 1$, $a_1 = 2$:

with probability 16%, we observe reward pair $(r_0, r_1) = (0,1)$

$$\mathbb{E}[\text{Regret}_T] \geq (T-2) \times \mathbb{P}(\text{select arm 2 for all } t > 1) \times (\text{regret of arm 2})$$
$$= (T-2) \times .16 \times 0.2 = \Omega(T)$$

Same rate as pure exploration!

Today

- Overview of reinforcement learning and this course
- Multi-armed bandits
  - Problem statement
  - Baseline approach 1: pure exploration
  - Baseline approach 2: pure greedy

Today's summary:

- Reinforcement learning is an *interactive* form of machine learning
    - Applicable whenever you want to learn to do something better
    - One component is learning while acting: exploration vs exploitation
    - Other component is optimization
- Multi-armed bandits (or MAB or just bandits)
    - Exemplify first component (exploration vs exploitation)
    - Pure greedy not much better than pure exploration (linear regret)

- Next time: trade off greediness with exploration
    - Explore-then-commit
    - ε-greedy