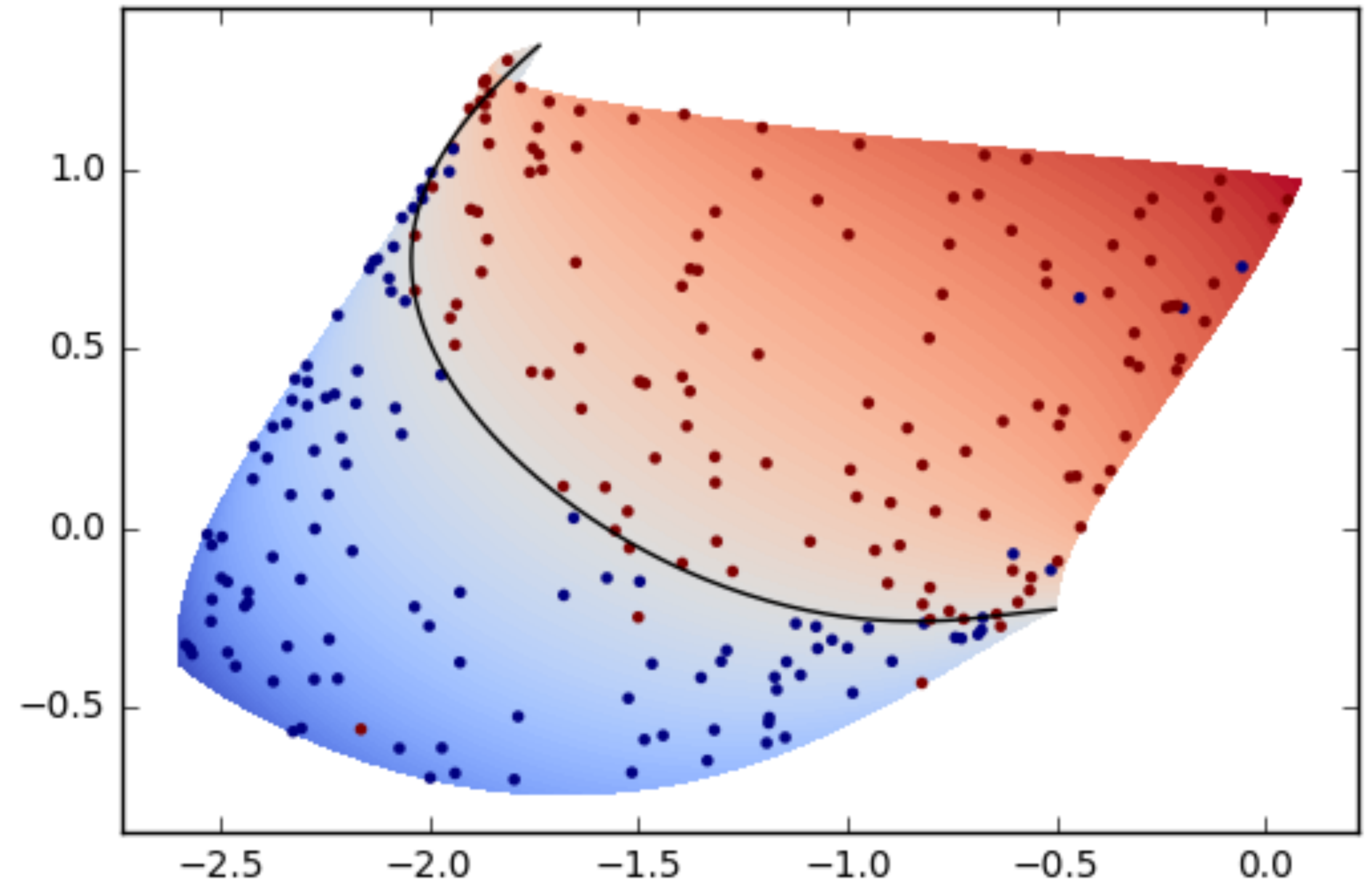
CS 273A: Machine Learning

Fall 2021

Lecture 17: Reinforcement Learning

Roy Fox
Department of Computer Science
Bren School of Information and Computer Sciences
University of California, Irvine

All slides in this course adapted from Alex Ihler & Sameer Singh



Logistics

assignments

- Assignment 5 due today

project

- Final report due Thursday

final exam

- Review: Thursday
- Final: next Tuesday, Dec 7, 10:30am–12:30

Today's lecture

Online learning

Sequential decision making

Imitation learning

Reinforcement learning

Online learning

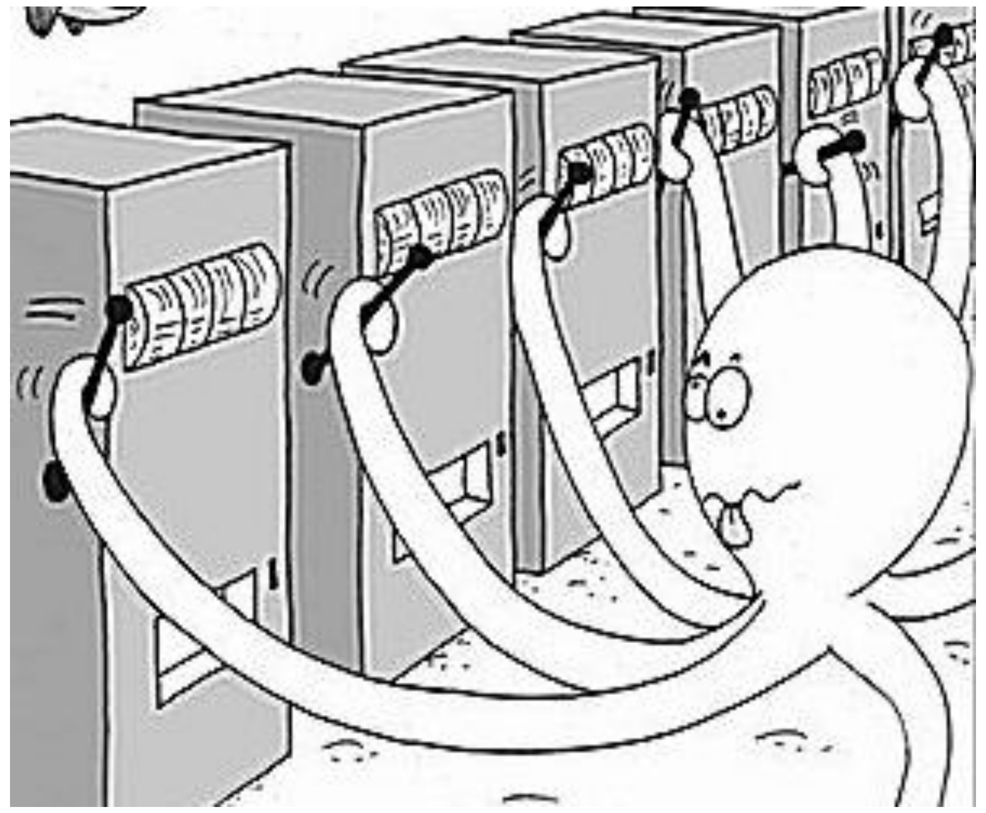
- In multi-class classification, we often assume 0–1 loss $\mathscr{L}(y, \hat{y}) = \delta[y \neq \hat{y}]$
- More generally, we can have different costs $\mathscr{L}(y, \hat{y}) = d(y, \hat{y})$
- Online learning:
  - Stream of instances, need to make predictions / decisions / actions online
  - We don't know the reward = -cost until we actually select $\hat{y}$
  - We'll never know the reward of other actions
- Objective:
  - Make better and better decisions (compared to what? later...)

Multi-Armed Bandits (MABs)

- Basic setting: single instance x, multiple actions $a_1, \ldots, a_k$
  - Each time we take action a_i we see a noisy reward $r_t \sim p_i$
- Can we maximize the expected reward $\max_i \mathbb{E}_{r \sim p_i}[r]$?
  - We can use the mean as an estimate $\mu_i = \mathbb{E}_{r \sim p_i}[r] \approx \frac{1}{m_i} \sum_{t \in T_i} r_t$
- Challenge: is the best mean so far the best action?
  - Or is there another that's better than it appeared so far?

One-armed bandit

Multi-armed bandit



Exploration vs. exploitation

- Exploitation = choose actions that seems good (so far)
- Exploration = see if we're missing out on even better ones
- Naïve solution: learn r by trying every action enough times
    - Suppose we can't wait that long: we care about rewards while we learn
- Regret = how much worse our return is than an optimal action

$$\rho(T) = T\mu_{a^*} - \sum_{t=0}^{T-1} r_t$$

    - Can we get the regret to grow sub-linearly with T? $\implies$ average goes to 0: $\frac{\rho(T)}{T} \to 0$

Let's play!

- http://iosband.github.io/2015/07/28/Beat-the-bandit.html

Simple exploration: ϵ-greedy

- With probability ϵ:
    - Select action uniformly at random
- Otherwise (w.p. $1 - \epsilon$):
    - Select best (on average) action so far
- Problem 1: all non-greedy actions selected with same probability
- Problem 2: must have $\epsilon \to 0$, or we keep accumulating regret
    - But at what rate should ϵ vanish?

Optimism under uncertainty

- Tradeoff: explore less used actions, but don't be late to start exploiting what's known
  - Principle: optimism under uncertainty = explore to the extent you're uncertain, otherwise exploit
- By the central limit theorem, the mean reward of each arm $\hat{\mu}_i$ quickly $\to \mathcal{N}\left(\mu_i, O\left(\frac{1}{m_i}\right)\right)$
- Be optimistic by slowly-growing number of standard deviations: $a = \arg\max_i \hat{\mu}_i + \sqrt{\frac{2 \ln T}{m_i}}$
  - Confidence bound: likely $\mu_i \leq \hat{\mu}_i + c\sigma_i$; unknown constant in the variance $\implies$ let c grow
  - But not too fast, or we fail to exploit what we do know
- Regret: $\rho(T) = O(\log T)$, provably optimal

Thompson sampling

- Consider a model of the reward distribution $p_{\theta_i}(r \mid a_i)$
- Suppose we start with some prior $q(\theta)$
  - Taking action a_t, see reward $r_t \implies$ update posterior $q(\theta \mid \{(a_{\leq t}, r_{\leq t})\})$
- Thompson sampling:
  - Sample $\theta \sim q$ from the posterior
  - Take the optimal action $a^* = \max_i \mathbb{E}_{r \sim p_{\theta i}}[r]$
  - Update the belief (different methods for doing this)
  - Repeat

Other online learning settings

- What is the reward for action a_i?
  - MAB: random variable with distribution $p_i(r)$
  - Adversarial bandits: adversary selects r_i for every action
    - The adversary knows our algorithm! And past action selection! But not future actions
      - Learner must be stochastic (= unpredictable) in choosing actions
    - Amazingly, there are learners with regret guarantees
- Contextual bandits: we also get instance x, make decision $\pi(a|x)$
  - Can we generalize to unseen instances?

Today's lecture

Online learning

Sequential decision making

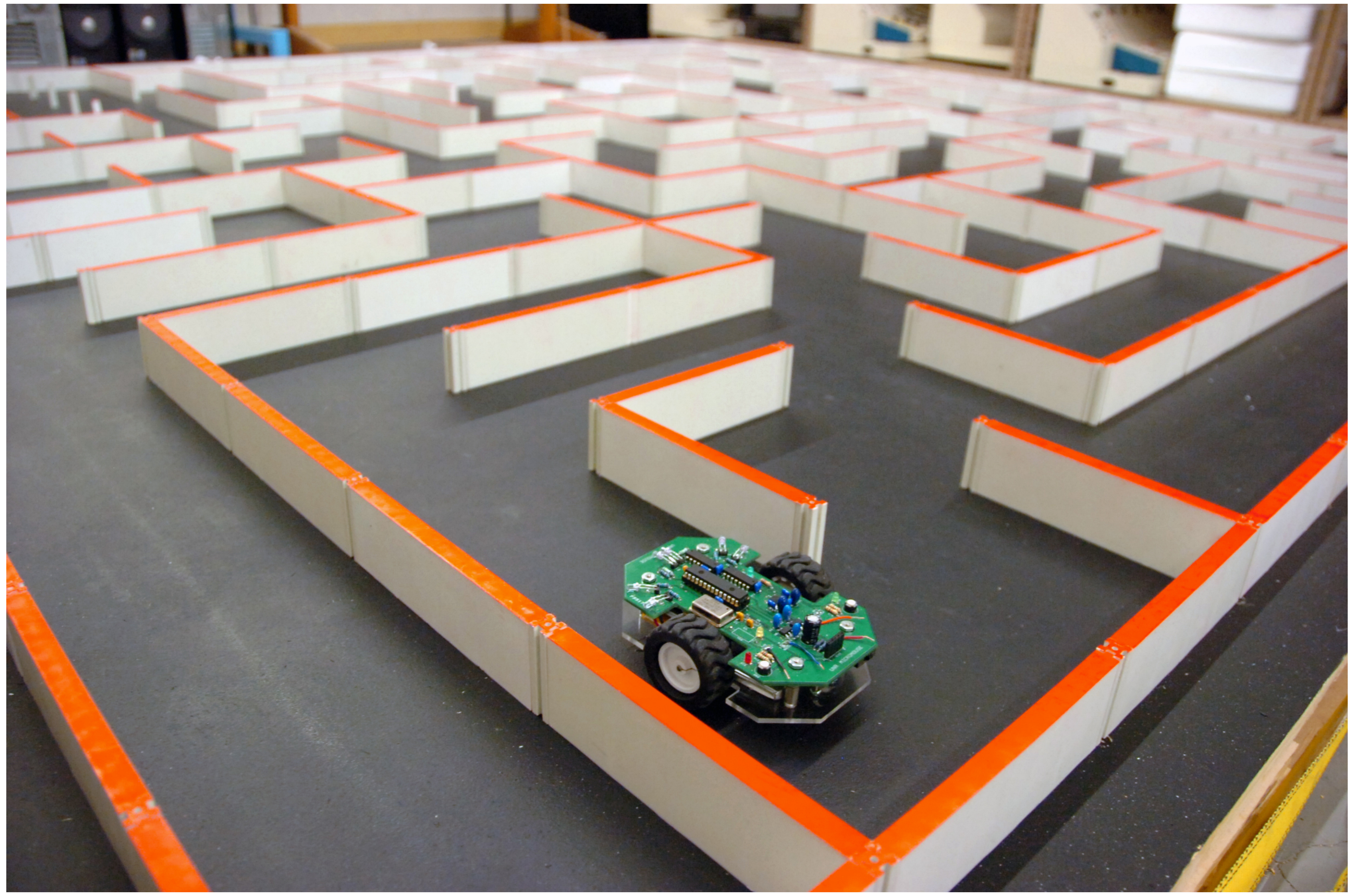
Imitation learning

Reinforcement learning

Sequential decision making

- Make a decision $f : x_1 \mapsto y_1$
- Based on y_1, face a new decision $f : x_2 \mapsto y_2$
  - Sequential: x_2 may depend on (x_1, y_1)
- After t steps (t for time), decide $f : x_t \mapsto y_t$
  - Sequential: x_t may depend on the history $(x_1, y_1, x_2, \ldots, y_{t-1})$
- Notation change: $\pi : s \mapsto a$
  - π for policy; s for state; a for action

System state



$$\dashrightarrow s_{t-1} \rightarrow s_t \rightarrow s_{t+1} \dashrightarrow$$

Markov Chain

- Markov property: the future is independent of the past, given the present

$$p(s_{t+1}, s_{t+2}, \ldots \mid s_1, s_2, \ldots, s_t) = p(s_{t+1}, s_{t+2}, \ldots \mid s_t)$$

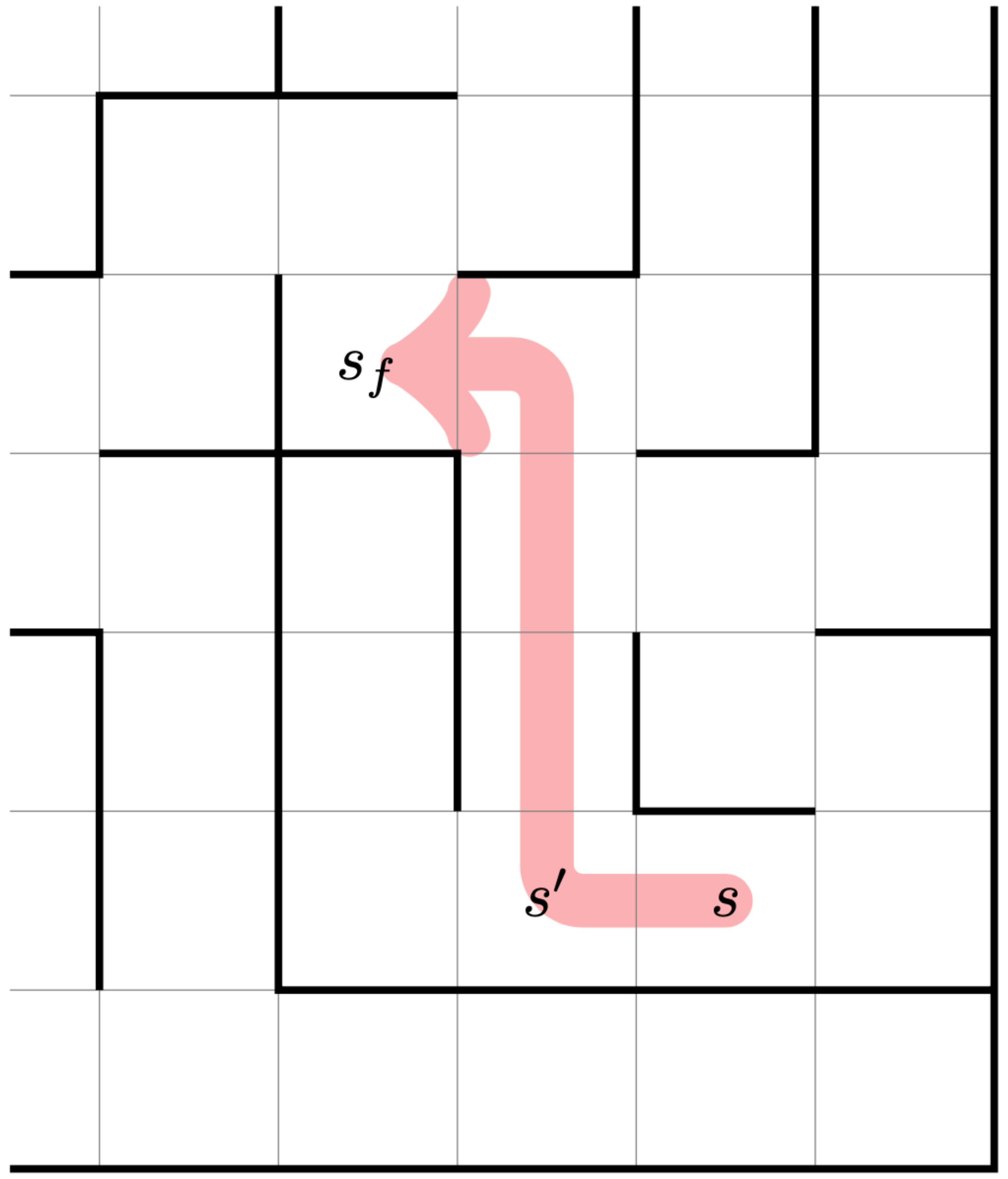
- State = all relevant information from history

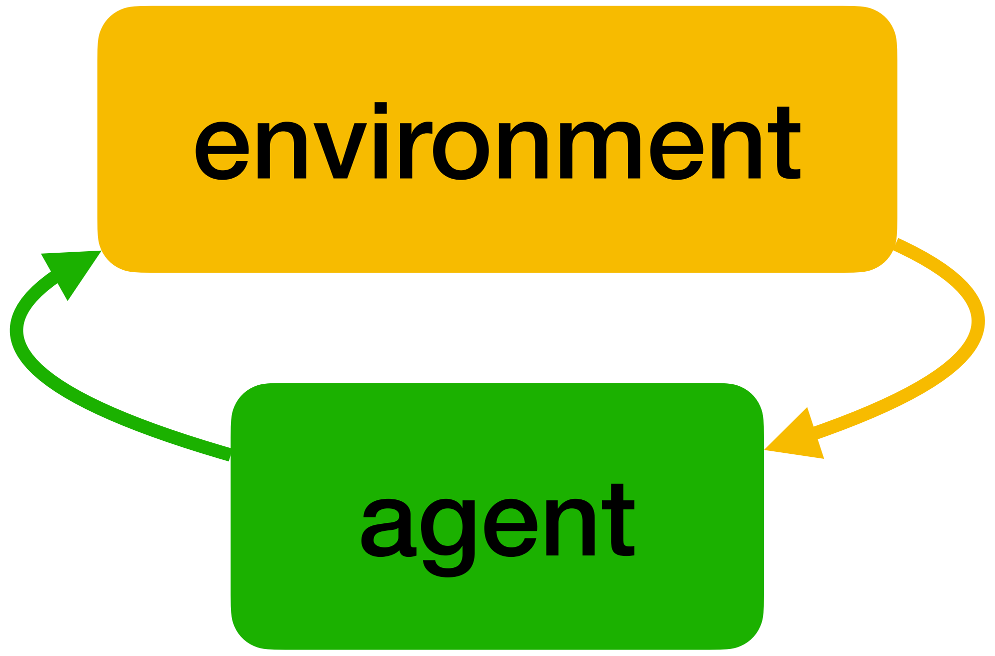
  for future!

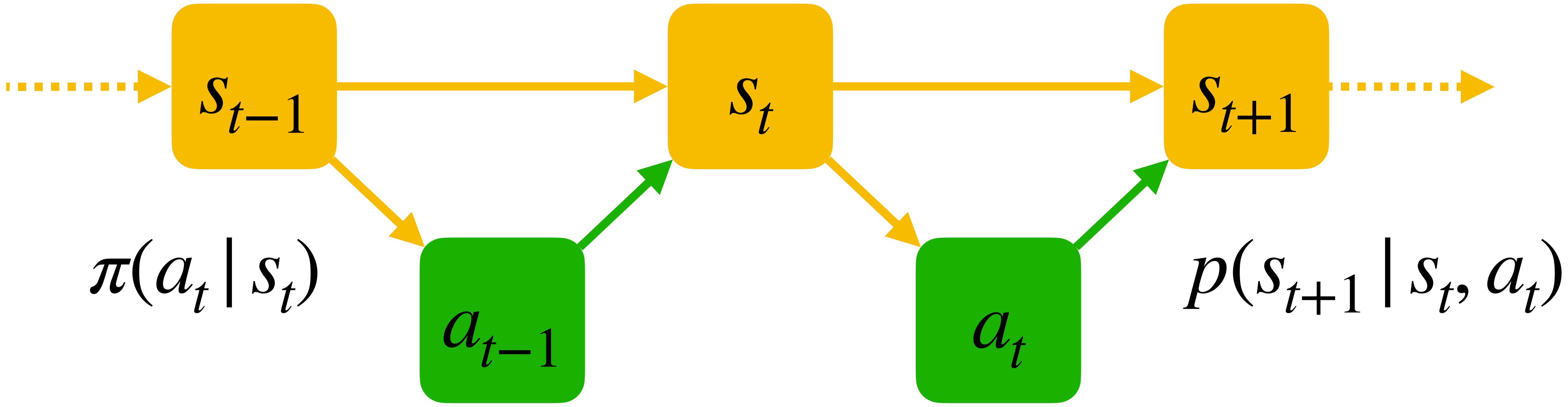
  - s_t is a sufficient statistic of $h = (s_1, \ldots, s_t)$ for $s_{t+1}, s_{t+2}, \ldots$

$$\cdots \to s_{t-1} \to s_t \to s_{t+1} \to \cdots$$

System = agent + environment

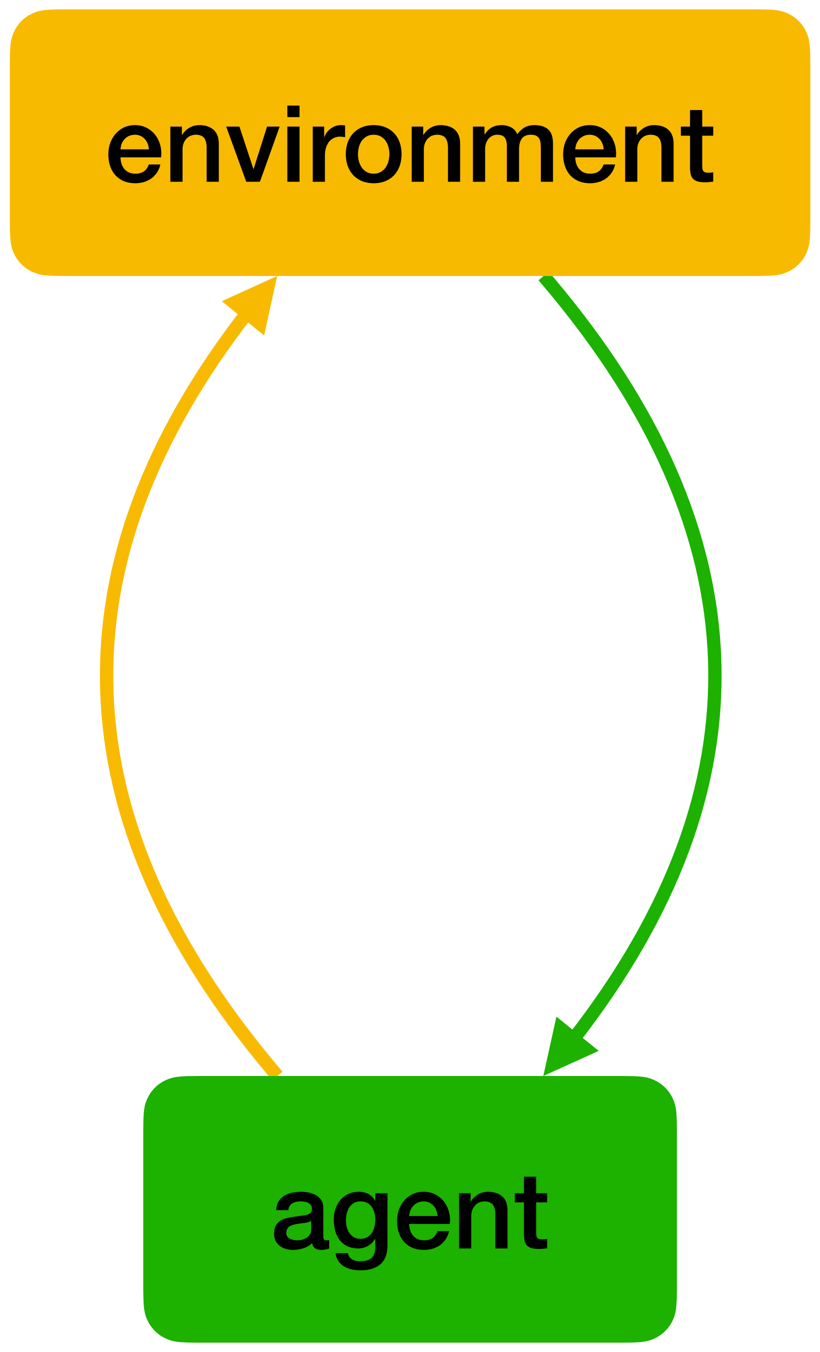




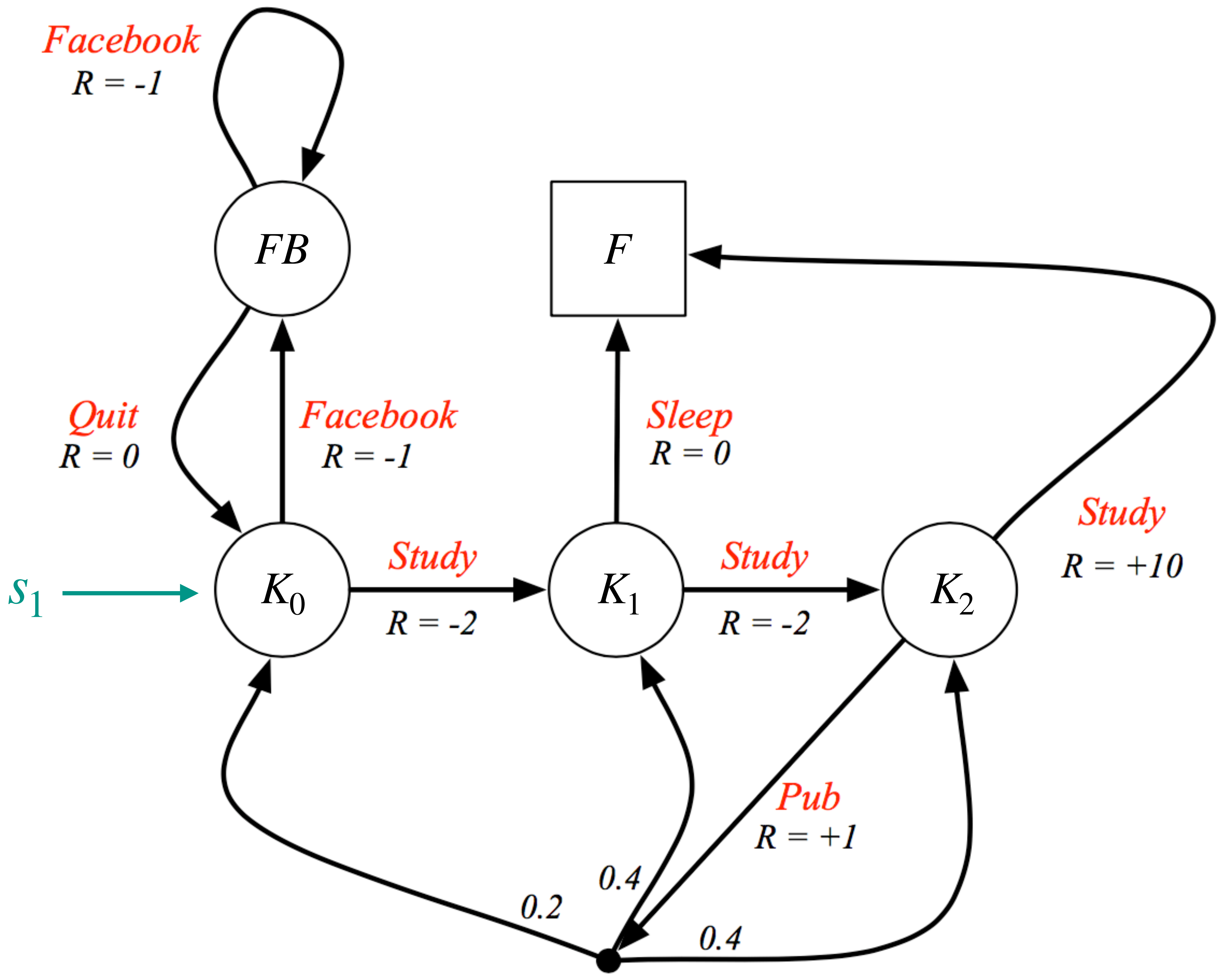


Markov Decision Process (MDP)

- Model of environment
  - S = set of states
  - A = set of actions
  - $p(s' \mid s, a)$ = probability that $s_{t+1} = s'$, if $s_t = s$ and $a_t = a$

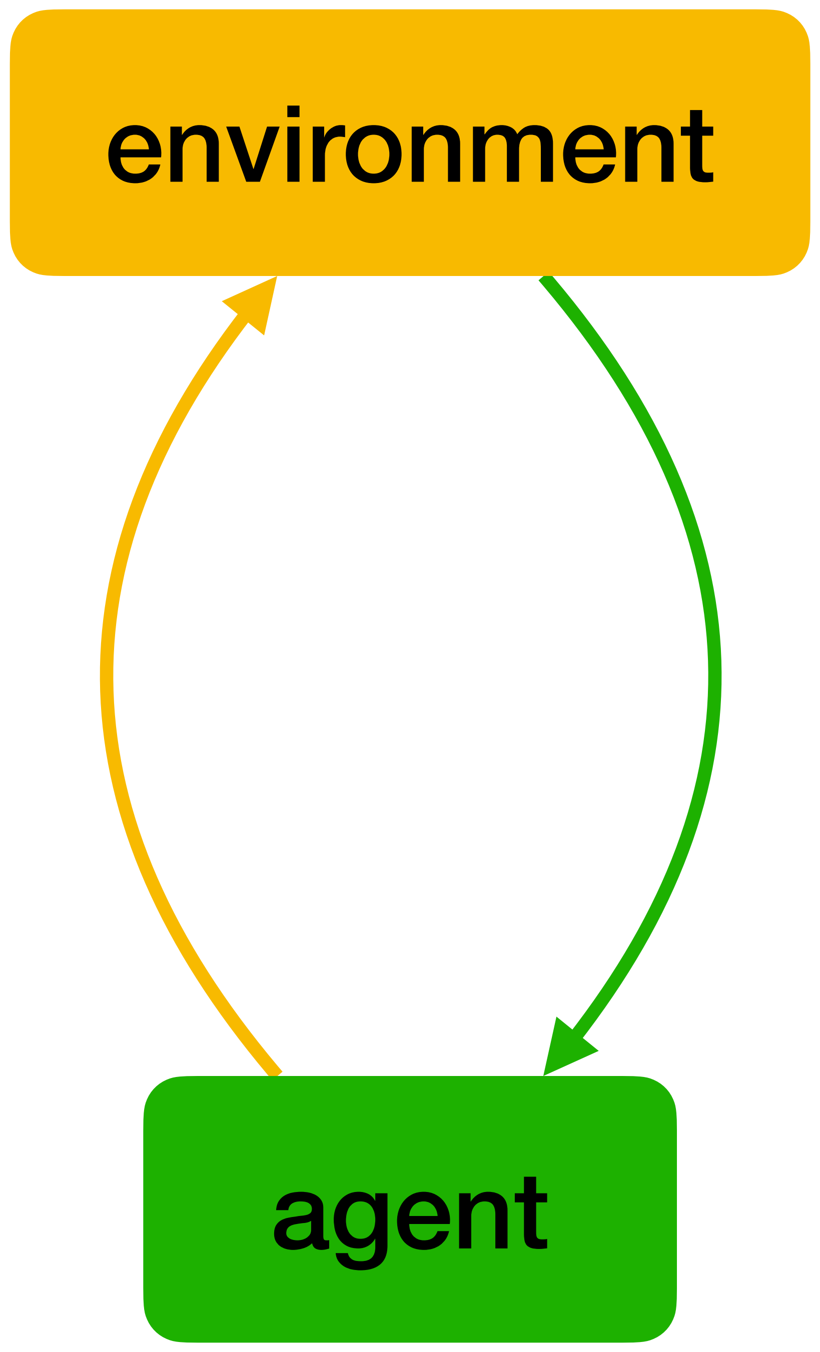


The Student MDP



Agent policy

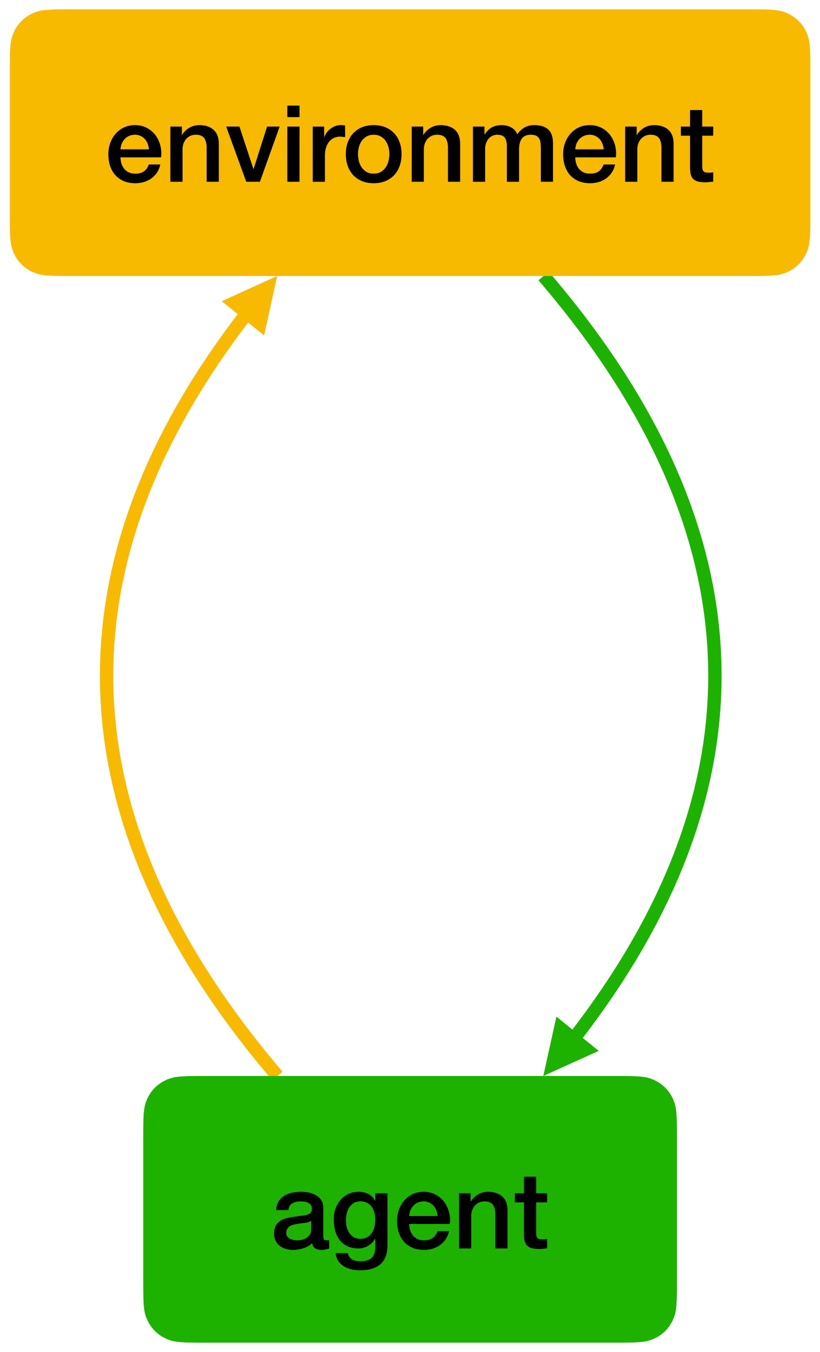
- "Model" of agent decision-making
  - policy $\pi(a \mid s)$ = probability of taking action $a_t = a$ in state $s_t = s$
  - In MDP, action a_t only depends on current state s_t:
    - Markov property = s_t is all that matters in history
    - Causality = cannot depend on the future
  - Should the policy depend on time? $\pi_t : s_t \mapsto a_t$
    - Sometimes; can add t as feature: $s_t \to (t, s_t)$



Trajectories

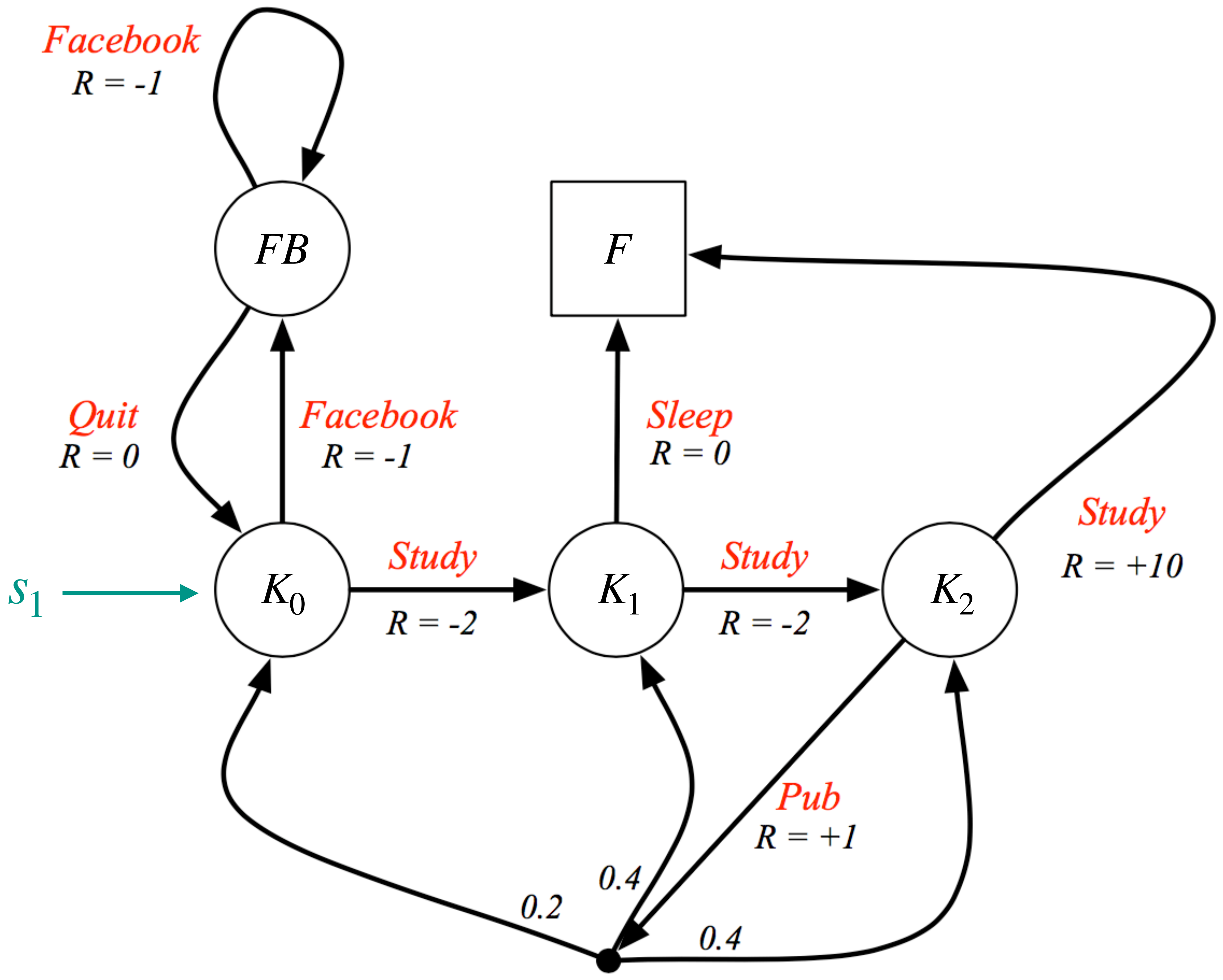
- The agent's behavior iteratively uses (rolls out) the policy
- Trajectory: $\xi = (s_1, a_1, s_2, a_2, \ldots, s_{T+1})$
- MDP + policy induce distribution over trajectories

$$p_\pi(\xi) = p(s_1)\pi(a_1 | s_1)p(s_2 | s_1, a_1)\cdots\pi(a_T | s_T)p(s_{T+1} | s_T, a_T)$$
$$= p(s_1)\prod_{t=1}^{T}\pi(a_t | s_t)p(s_{t+1} | s_t, a_t)$$



The Student MDP

$$p(K_0, \textit{Facebook}, FB, \textit{Quit}, K_0, \textit{Study}, K_1, \textit{Study}, K_2, \textit{Pub}, K_1, \textit{Sleep}, F)$$
$$= 1 \cdot \pi(\textit{Facebook} \mid K_0) \cdot 1 \cdot \pi(\textit{Quit} \mid FB) \cdot 1 \cdot \pi(\textit{Study} \mid K_0) \cdot 1 \cdot \pi(\textit{Study} \mid K_1) \cdot 1 \cdot \pi(\textit{Pub} \mid K_2) \cdot p(K_1 \mid K_2, \textit{Pub}) \cdot \pi(\textit{Sleep} \mid K_1) \cdot 1$$



Today's lecture

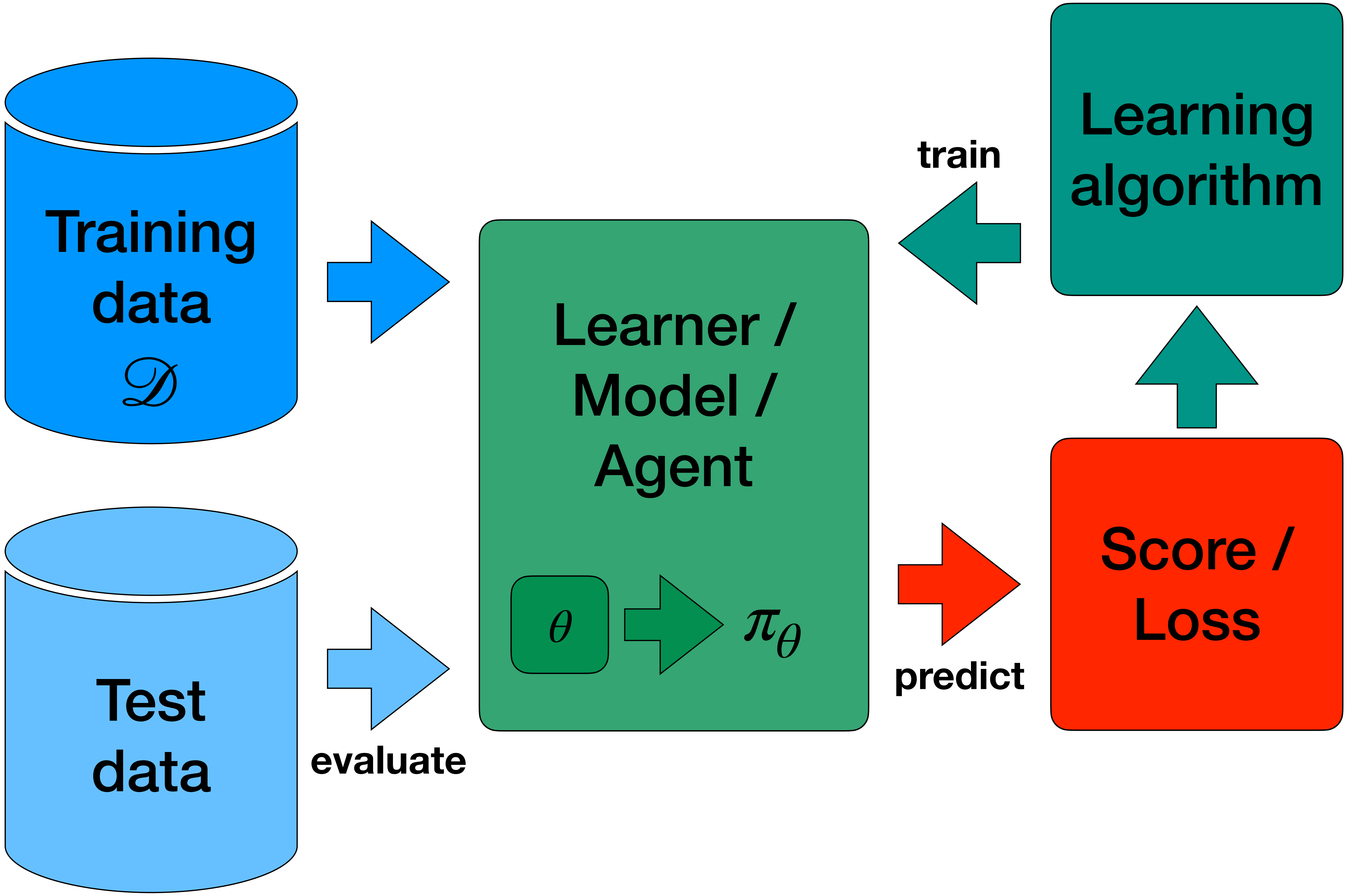
Online learning

Sequential decision making

Imitation learning

Reinforcement learning

Learning from Demonstrations (LfD)



Learning from Demonstrations (LfD)

- Teacher provides demonstration trajectories $\mathscr{D} = \{\xi^{(1)}, \ldots, \xi^{(m)}\}$

- Learner trains a policy π_θ to minimize a loss $\mathscr{L}(\theta)$

- For example, negative log-likelihood (NLL):

$$\arg\min_\theta \mathscr{L}_\theta(\xi) = \arg\min_\theta(-\log p_\theta(\xi))$$

$$= \arg\max_\theta \left(\log p(s_1) + \sum_{t=1}^{T} \log \pi_\theta(a_t \,|\, s_t) + \log p(s_{t+1} \,|\, s_t, a_t) \right)$$

$$= \arg\max_\theta \sum_{t=1}^{T} \log \pi_\theta(a_t \,|\, s_t)$$

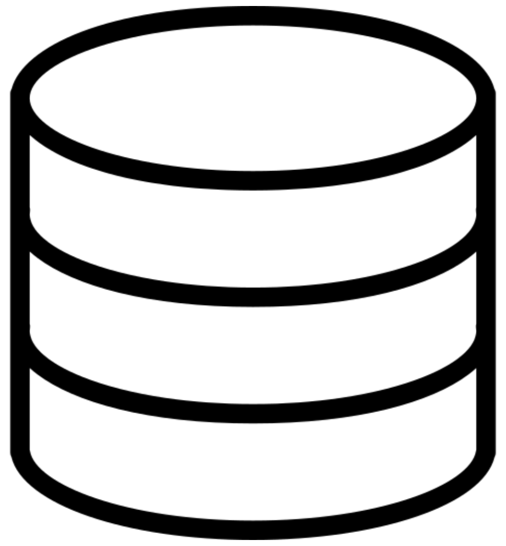
model-free
= no need to know the environment dynamics p

Behavior Cloning (BC)

- Behavior Cloning:
  - Break down trajectories $\{\xi^{(1)}, \ldots, \xi^{(m)}\}$ into steps $\{(s_1^{(1)}, a_1^{(1)}), \ldots, (s_{T_m}^{(m)}, a_{T_m}^{(m)})\}$
  - Train $\pi_\theta : s \mapsto a$ using supervised learning
- Benefits:
  - Simple, flexible — can use any learning algorithm
  - Model-free — never need to know the environment
- Drawbacks:
  - Only as good as the demonstrator
  - Only good in demonstrated states — how do we evaluate?

Inaccuracy in BC

observations + actions

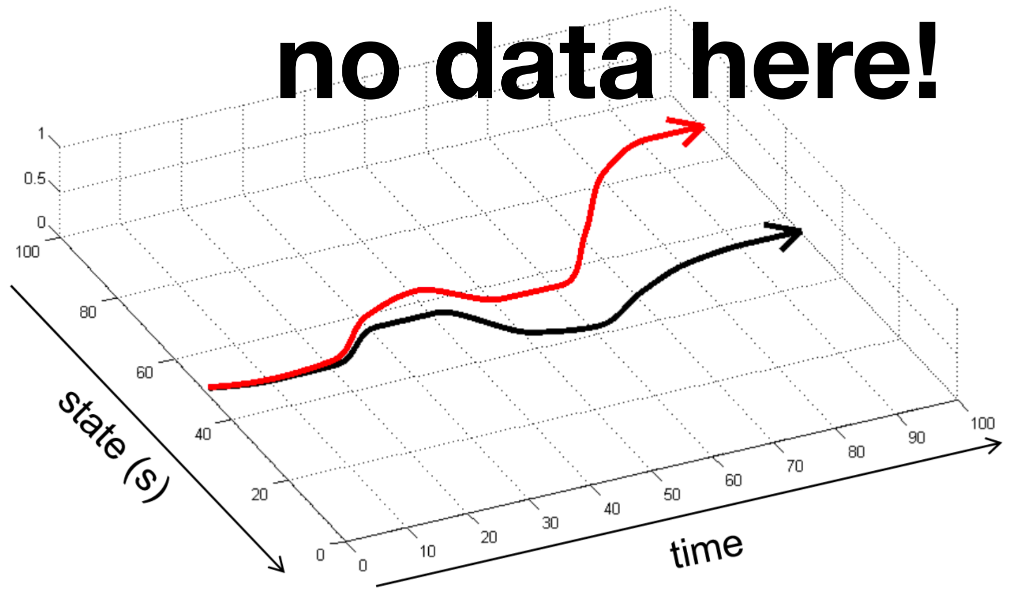


training data

supervised learning

$\pi_\theta(a_t | s_t)$

- We could evaluate on held out teacher data, but really interested in using π_θ
- If the policy approximates the teacher $\pi_\theta(a_t | s_t) \approx \pi^*(a_t | s_t)$
  - The trajectory distribution will also approximate teacher behavior $p_{\pi_\theta}(\xi) \approx p_{\pi^*}(\xi)$
- But errors accumulate over time
  - May reach states not seen in the training dataset



Gathering experience

- Machine learning works when training distribution = test distribution
    - We train on p_{π^*} but test on p_{π_θ}
    - Problem: we don't know π_θ until after training
- Dataset Aggregation (DAgger):
    - Roll out learner trajectories $\xi \sim p_{\pi_\theta}$
    - Ask teacher to label reached states s_t with correct actions a_t as in active learning
    - Add to dataset, train new π_θ, repeat

Today's lecture

Online learning

Sequential decision making

Imitation learning

Reinforcement learning

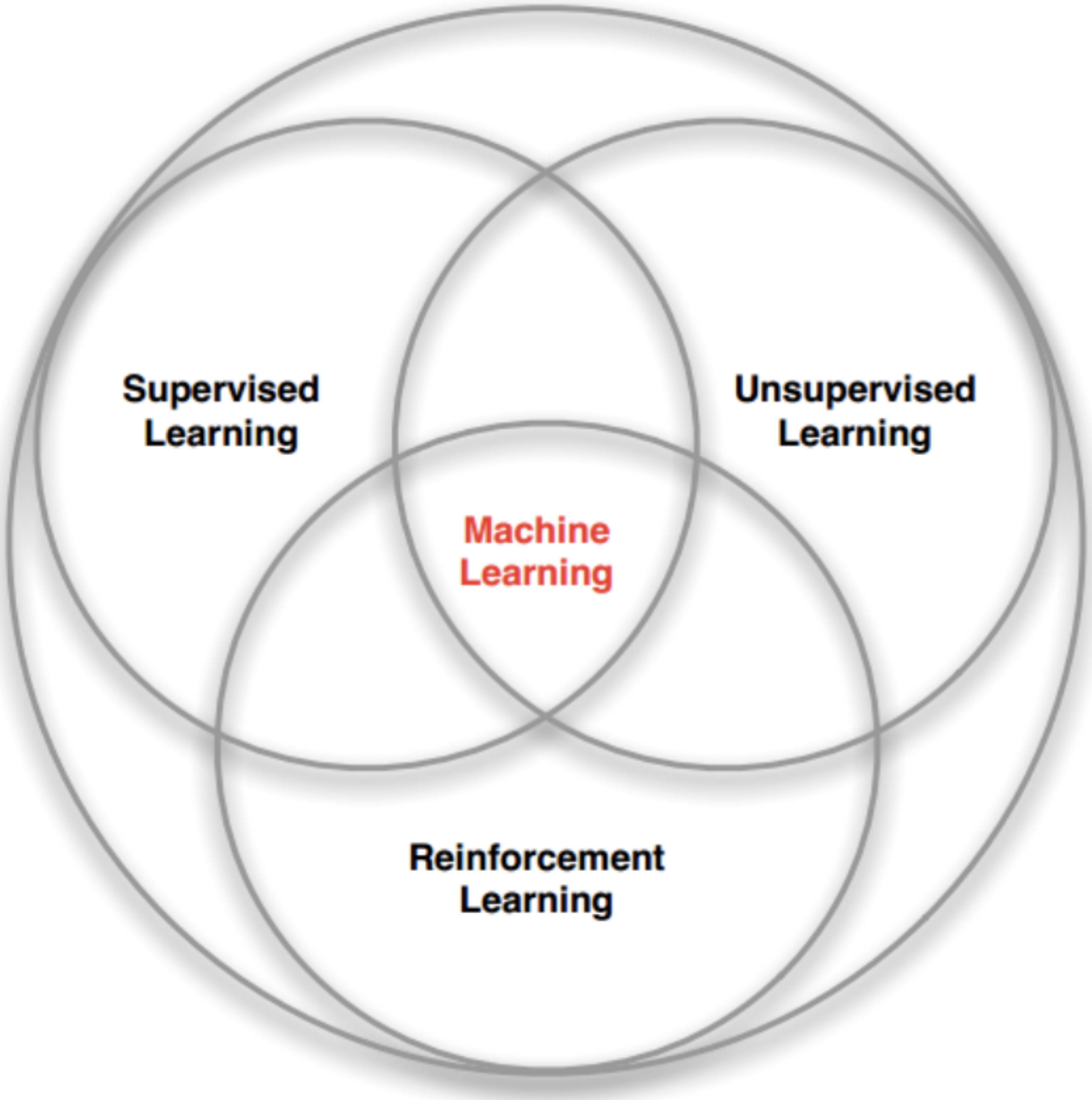
Learning from rewards

- Providing demonstrations is hard
  - Particularly for learner-generated trajectories

as in online learning

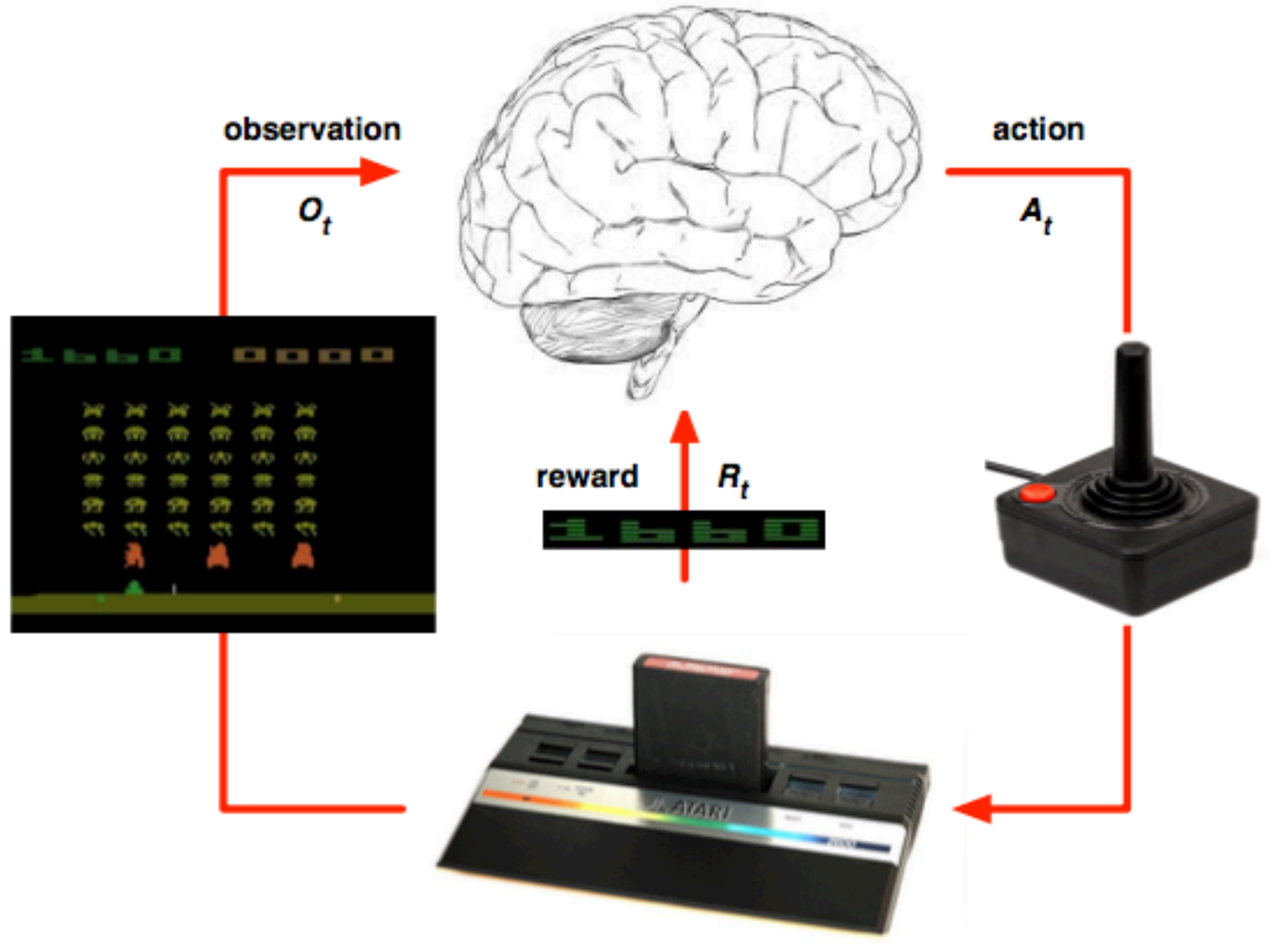
- Can the teacher just score learner actions?
  - Reward: $r(s, a) \in \mathbb{R}$
- High reward is positive reinforcement for this behavior (in this state)
  - Much closer to how natural agents learn
  - Designing and programming r often easier than programming / demonstrating π

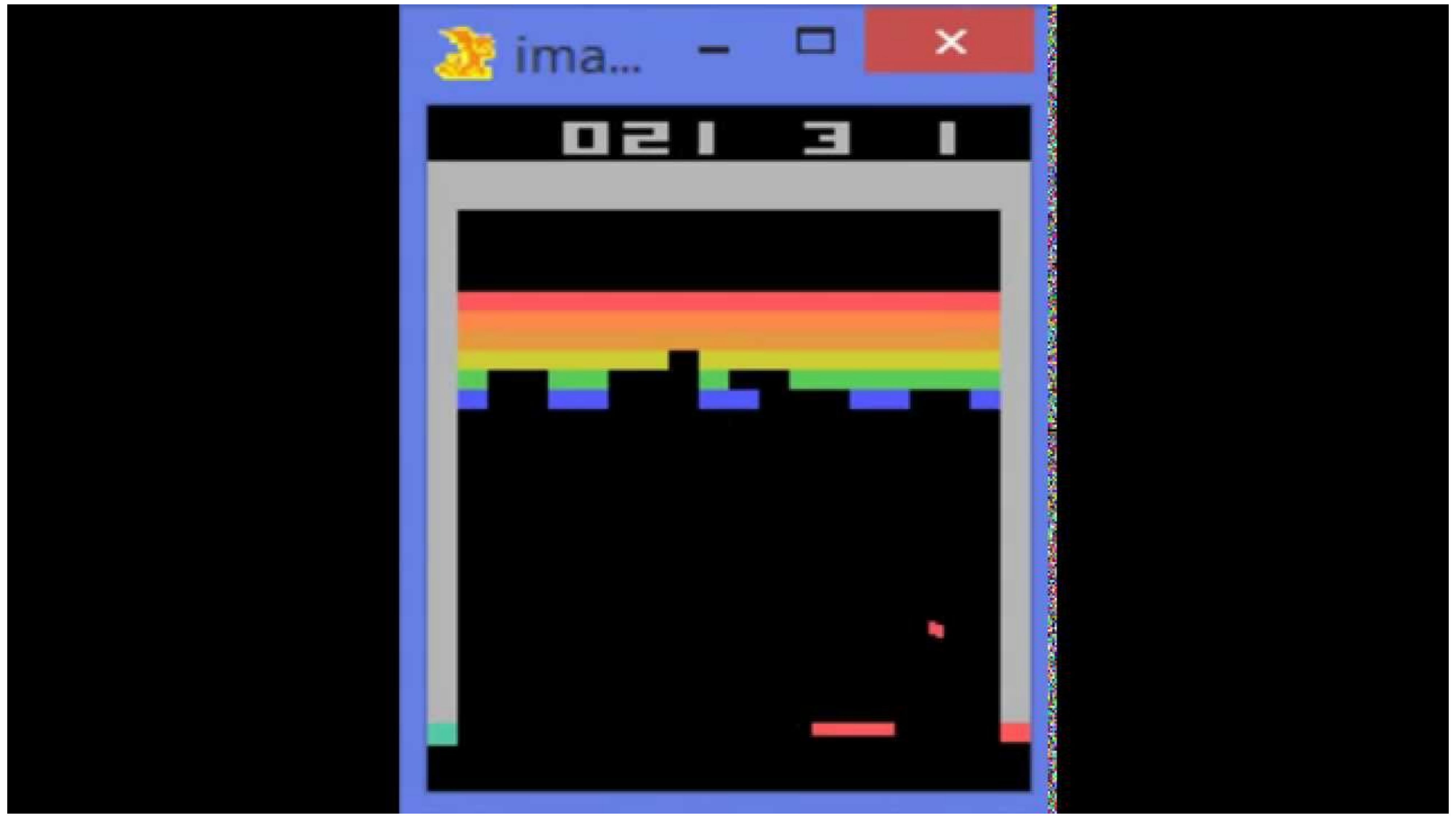
Reinforcement Learning (RL)



Example: Atari

- Rules are unknown
  - What makes the score increase?
- Dynamics are unknown
  - How do actions change pixels?
- Reward = score increase is known
  - Try to maximize total reward





https://www.youtube.com/watch?v=V1eYniJ0Rnk

Actions have long-term consequences

- Tradeoff: short-term rewards vs. long-term returns (accumulated rewards)
    - Fly drone: slow down to avoid crash?
    - Games: slowly build strength? block opponent? all out attack?
    - Stock trading: sell now or wait for growth?
    - Infrastructure control: reduce power output to prevent blackout?
    - Life: invest in college, obey laws, get started early on course project
- Forward thinking and planning are hallmarks of intelligence

Returns

- Return = total reward = $R = \sum_t \gamma^t r(s_t, a_t)$
  - Summarize reward sequence $r_t = r(s_t, a_t)$ as single number to be maximized
- Discount factor $\gamma \in [0,1]$
  - Higher weight to short-term rewards (and costs) than long-term
  - Good mathematical properties:
    - Assures convergence, simplifies algorithms, reduces variance
  - Vaguely economically motivated (inflation)

Policy evaluation

- What future return can the agent expect in state s_t by using policy π?

$$V_\pi(s_t) = \mathbb{E}\left[\sum_{t' \geq t} \gamma^{t'-t} r(s_{t'}, a_{t'}) \,\middle|\, s_t \right]$$

the rest of the process

first reward

$$= \mathbb{E}_{(a_t|s_t)\sim\pi}\left[r(s_t, a_t) + \gamma \mathbb{E}_{(s_{t+1}|s_t,a_t)\sim p}\left[\mathbb{E}\left[\sum_{t' \geq t+1} \gamma^{t'-(t+1)} r(s_{t'}, a_{t'}) \,\middle|\, s_{t+1} \right] \right] \right]$$

first action

first state transition

$$= \mathbb{E}_{(a_t|s_t)\sim\pi}[r(s_t, a_t) + \gamma \mathbb{E}_{(s_{t+1}|s_t,a_t)\sim p}[V_\pi(s_{t+1})]]$$

Action-value function

- What future return can the agent expect by taking a_t in s_t (and π in future)?

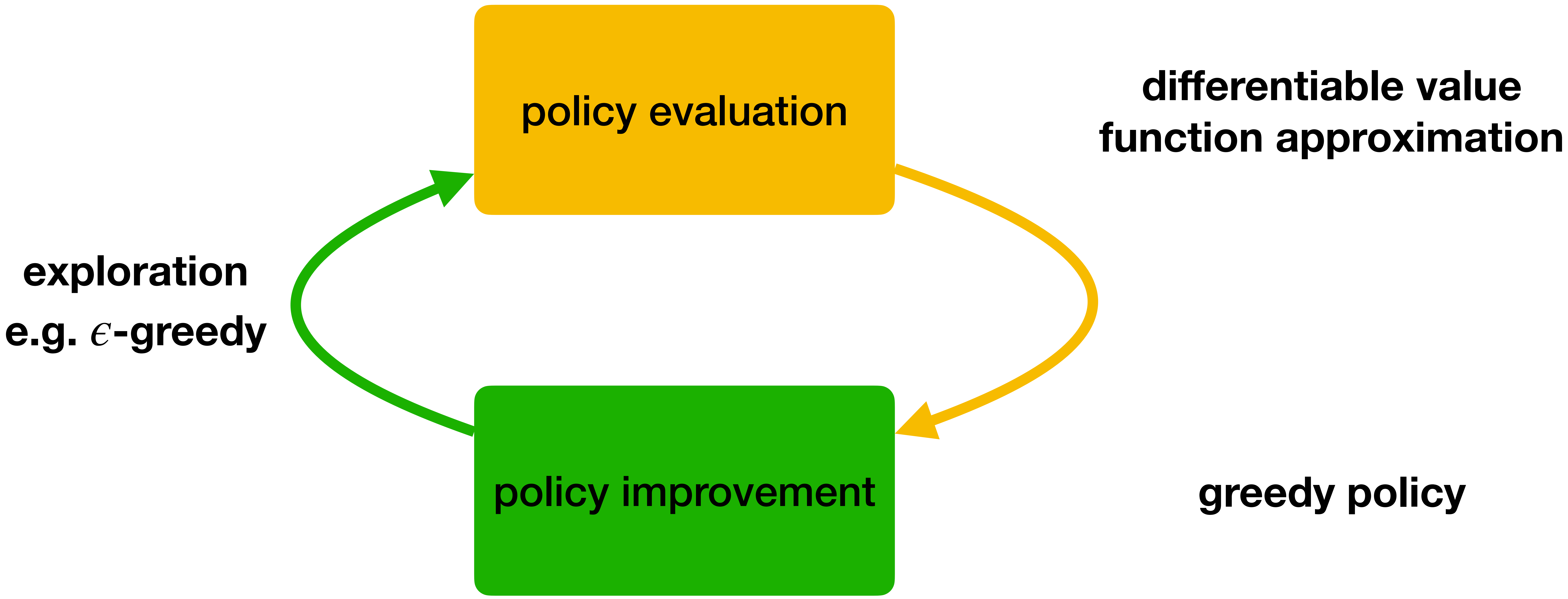
$$Q_\pi(s_t, a_t) = \mathbb{E}\left[\sum_{t' \geq t} \gamma^{t'-t} r(s_{t'}, a_{t'}) \middle| s_t, a_t\right] = r(s_t, a_t) + \gamma \mathbb{E}_{(s_{t+1}|s_t,a_t)\sim p}[V_\pi(s_{t+1})]$$

- If we have a guess for the value function V_π
    - And we interact with the environment to get experience (s_t, a_t, r_t, s_{t+1})
    - Then $r(s_t, a_t) + \gamma V_\pi(s_{t+1})$ is (in expectation) a good guess for $Q_\pi(s_t, a_t)$
    - Policy evaluation: on experience (s, a, r, s'), update $Q(s, a)$ toward $r + \gamma V(s')$

Optimal policy

- If we know we will use π in the future, what should we do now?
  - Greedy policy: $\pi^*(s_t) = \arg\max_{a_t} Q_\pi(s_t, a_t)$
  - In stochastic notation: $\pi^*(a_t \mid s_t) = 1$ for the greedy action
- If we have a guess for the action-value function $Q(s, a)$
  - Then $V(s) = \max_a Q(s, a)$ is a value function of a better policy
- This gives us a policy improvement step
  - Can be put together with policy evaluation $Q(s, a) \to r + \gamma V(s')$

Putting it all together: Deep Q Learning (DQN)



DQN pseudocode

Algorithm 1 DQN

initialize θ for Q_θ, set $\bar\theta \leftarrow \theta$
for each step **do**
  if new episode, reset to s_0
  observe current state s_t
  take ϵ-greedy action a_t based on $Q_\theta(s_t,\cdot)$

exploration

$$\pi(a_t|s_t) = \begin{cases} 1 - \frac{|\mathcal{A}|-1}{|\mathcal{A}|}\epsilon & a_t = \operatorname{argmax}_a Q_\theta(s_t,a) \\ \frac{1}{|\mathcal{A}|}\epsilon & \text{otherwise} \end{cases}$$

  get reward r_t and observe next state s_{t+1}
  add (s_t, a_t, r_t, s_{t+1}) to replay buffer $\mathcal{D}$ — agent creates its own "training set"
  for each (s,a,r,s') in minibatch sampled from $\mathcal{D}$ **do**

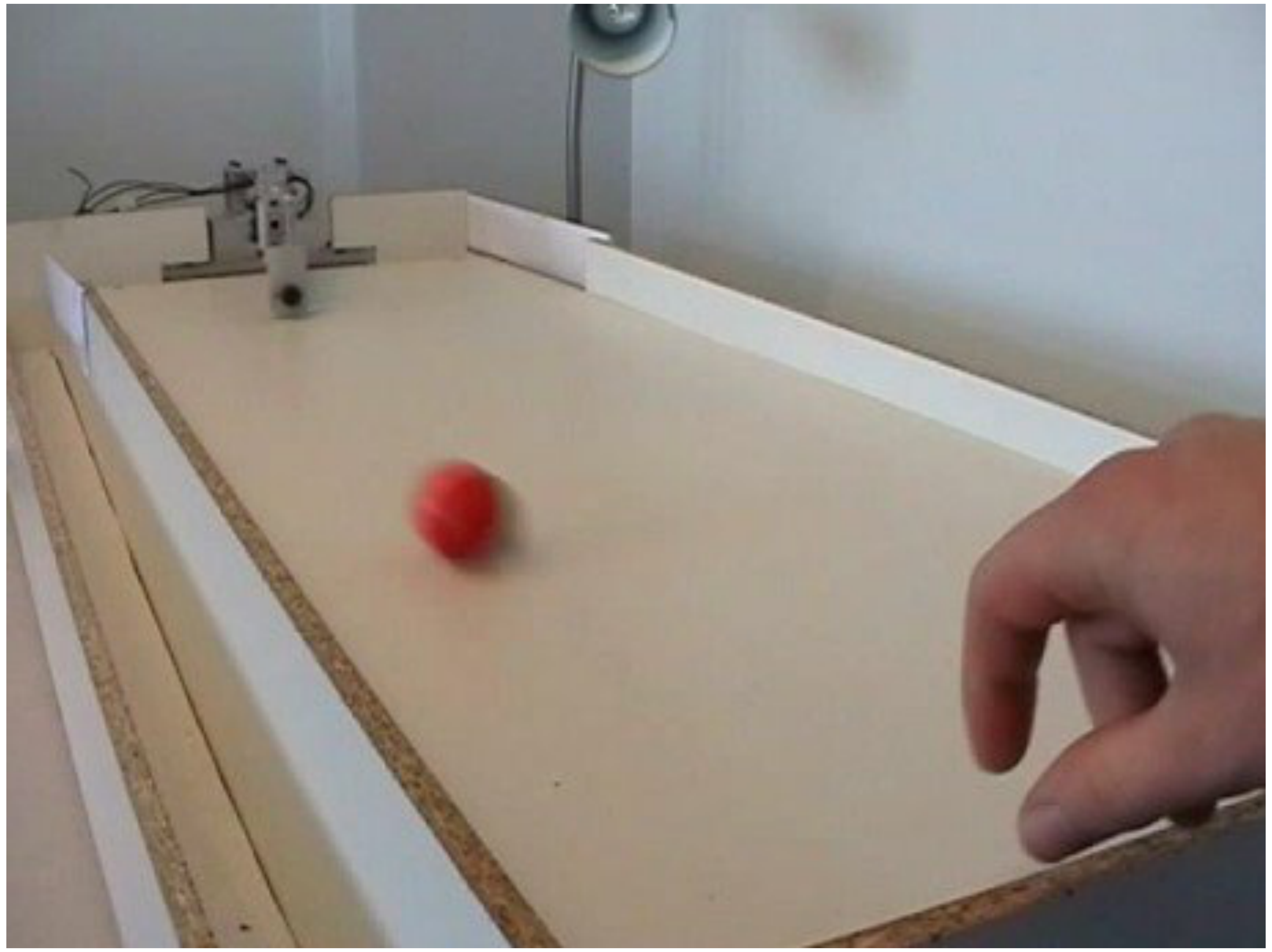
policy improvement

$$y \leftarrow \begin{cases} r & \text{if episode terminated at } s' \\ r + \gamma \max_{a'} Q_{\bar\theta}(s',a') & \text{otherwise} \end{cases}$$

   compute gradient $\nabla_\theta (y - Q_\theta(s,a))^2$ — policy evaluation
  take minibatch gradient step
  every K steps, set $\bar\theta \leftarrow \theta$

the target y isn't fixed as in supervised learning
it keeps changing as the agent improves
$\bar\theta$ is a slowly updated target network, to stabilize y

Example: Table Soccer



https://www.youtube.com/watch?v=CIF2SBVY-J0

Logistics

assignments	• Assignment 5 due today
project	• Final report due Thursday
final exam	• Review: Thursday • Final: next Tuesday, Dec 7, 10:30am–12:30