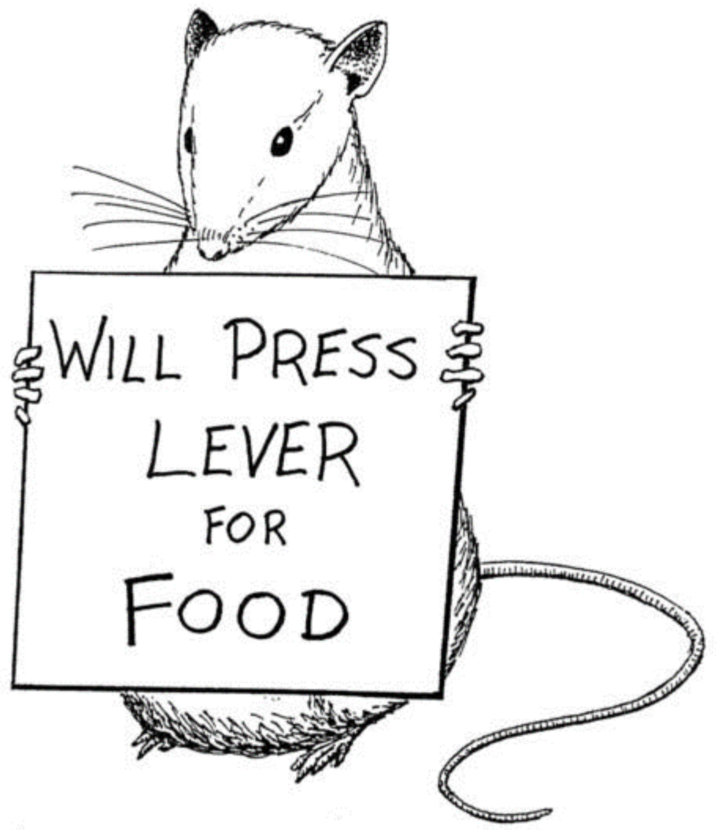
CS 277: Control and Reinforcement Learning

Winter 2024

Lecture 2: Imitation Learning

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



Logistics

logistics

- Follow announcements and discussions on ed
- See website for schedule, recordings, resources, etc.

exercises

- Quiz 1 due next Monday
- Exercise 1 to be published soon, due next Friday

Today's lecture

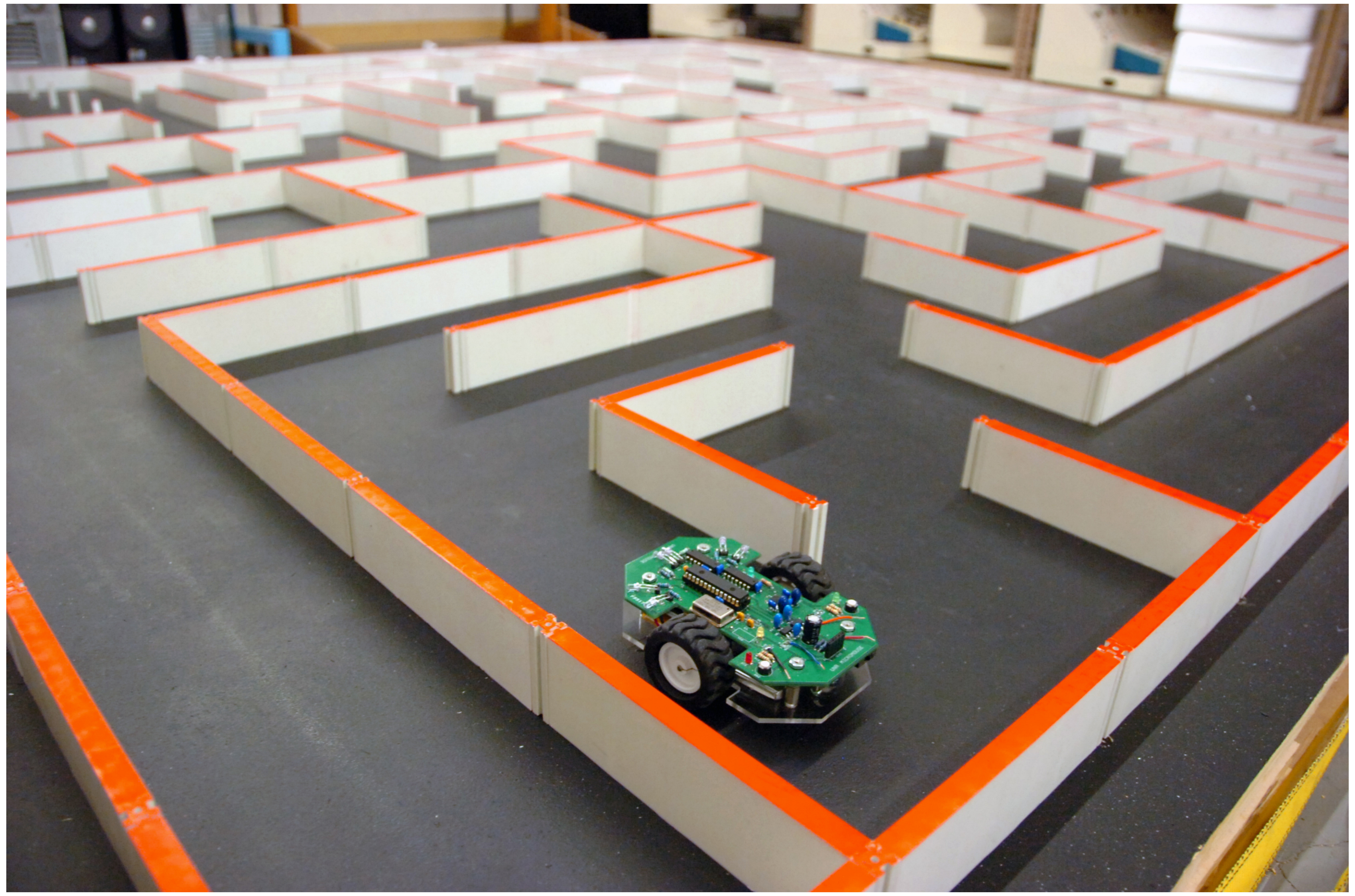
Basic RL concepts

Behavior Cloning

Better behavior modeling

Alleviating train–test mismatch

System state



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

System state

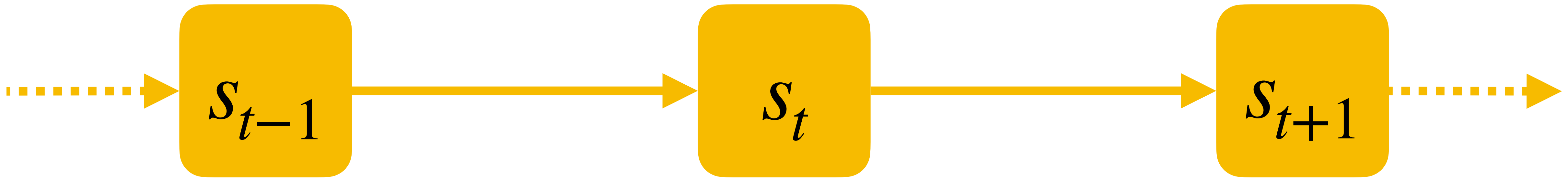
- Markov property: the future is independent of the past, given the present

$$p(s_{t+1}, s_{t+2}, \ldots \mid s_0, s_2, \ldots, s_t) = p(s_{t+1}, s_{t+2}, \ldots \mid s_t)$$

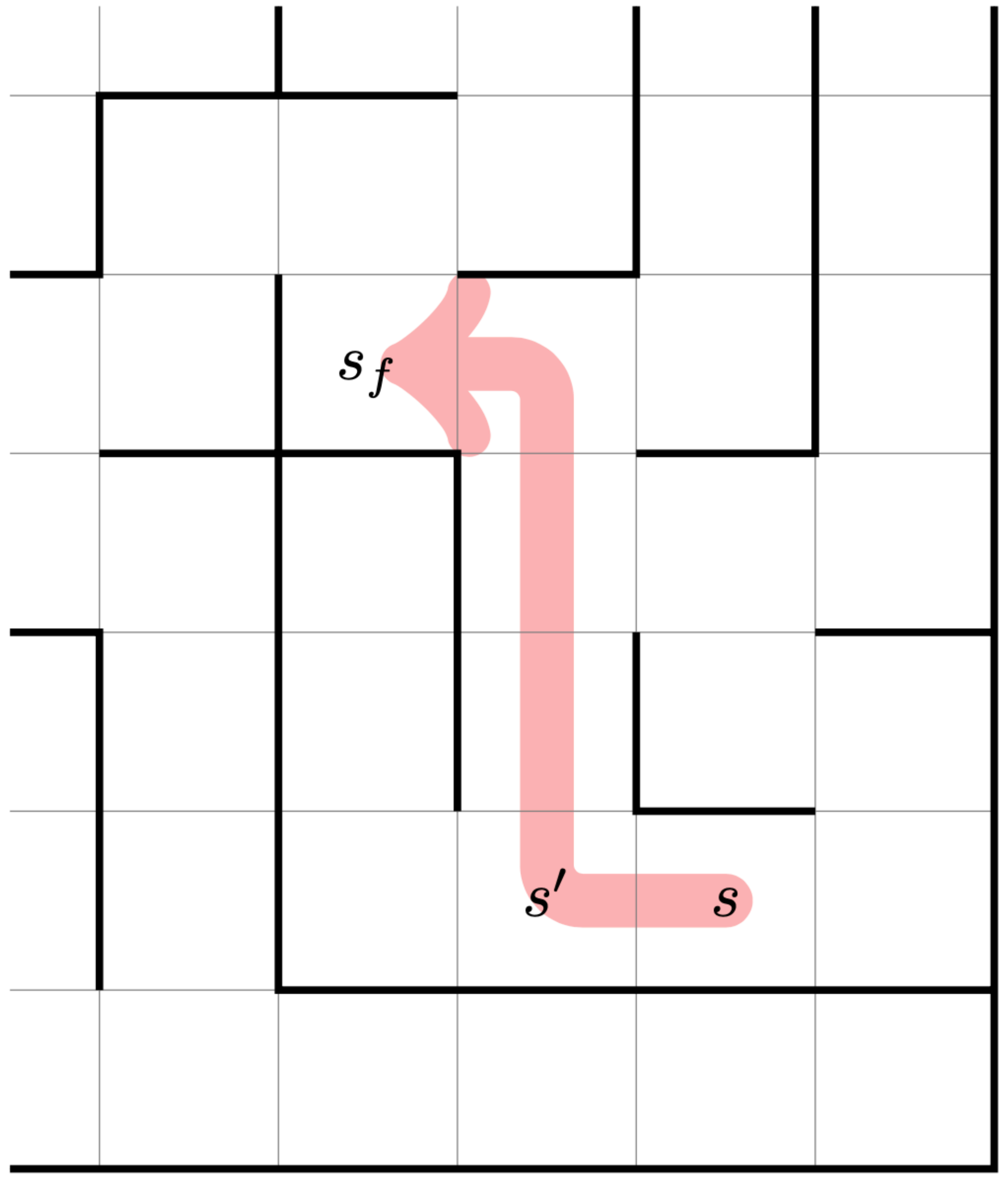
- State = all relevant information from history

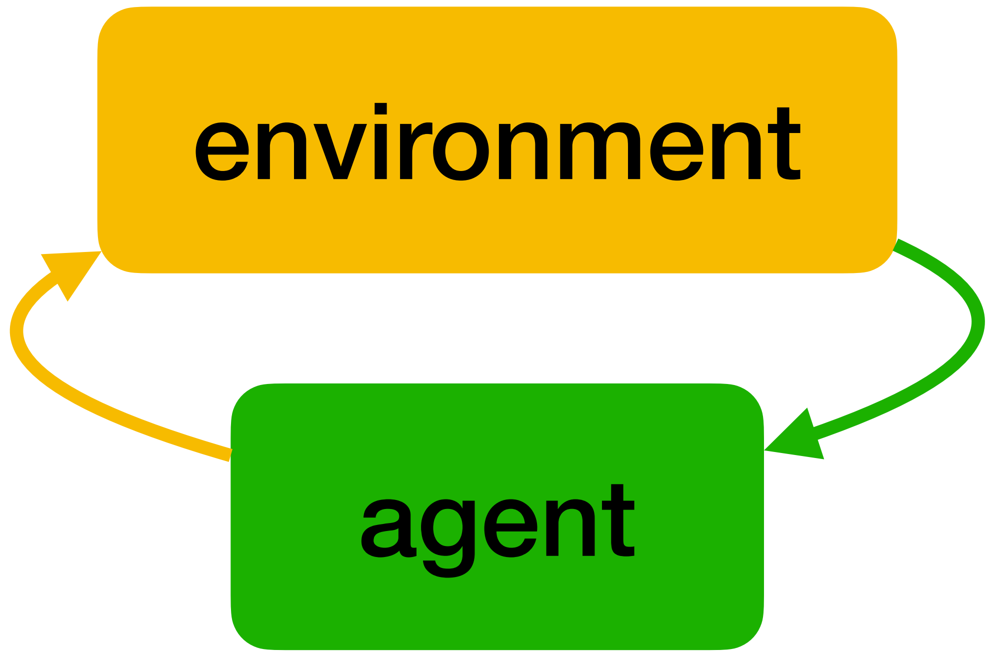
  for future!

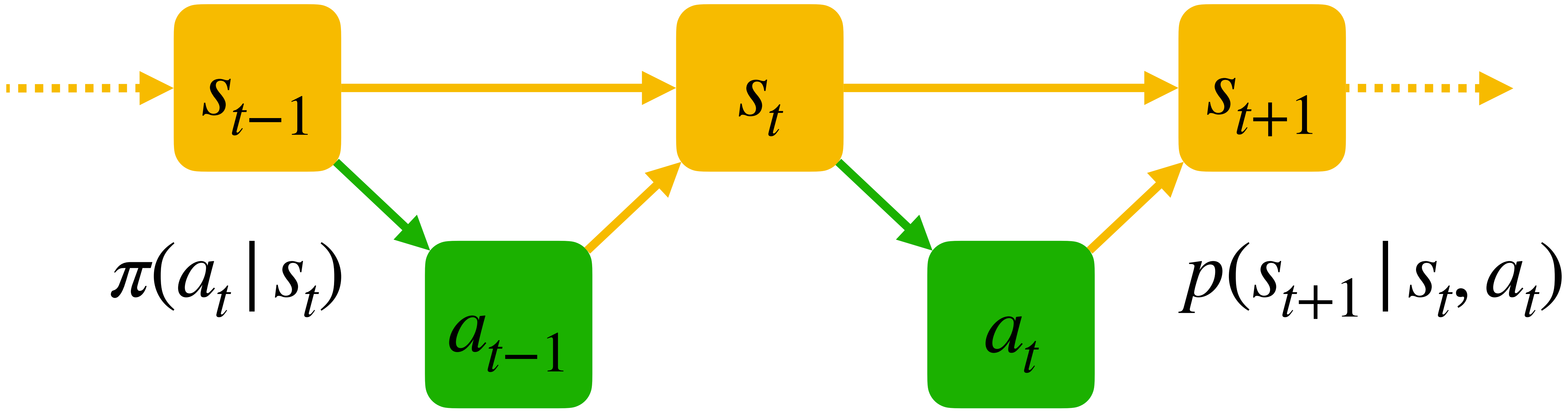
  - Given s_t, the history $h = (s_0, \ldots, s_t)$ and the future $(s_{t+1}, s_{t+2}, \ldots)$ are independent



System = agent + environment

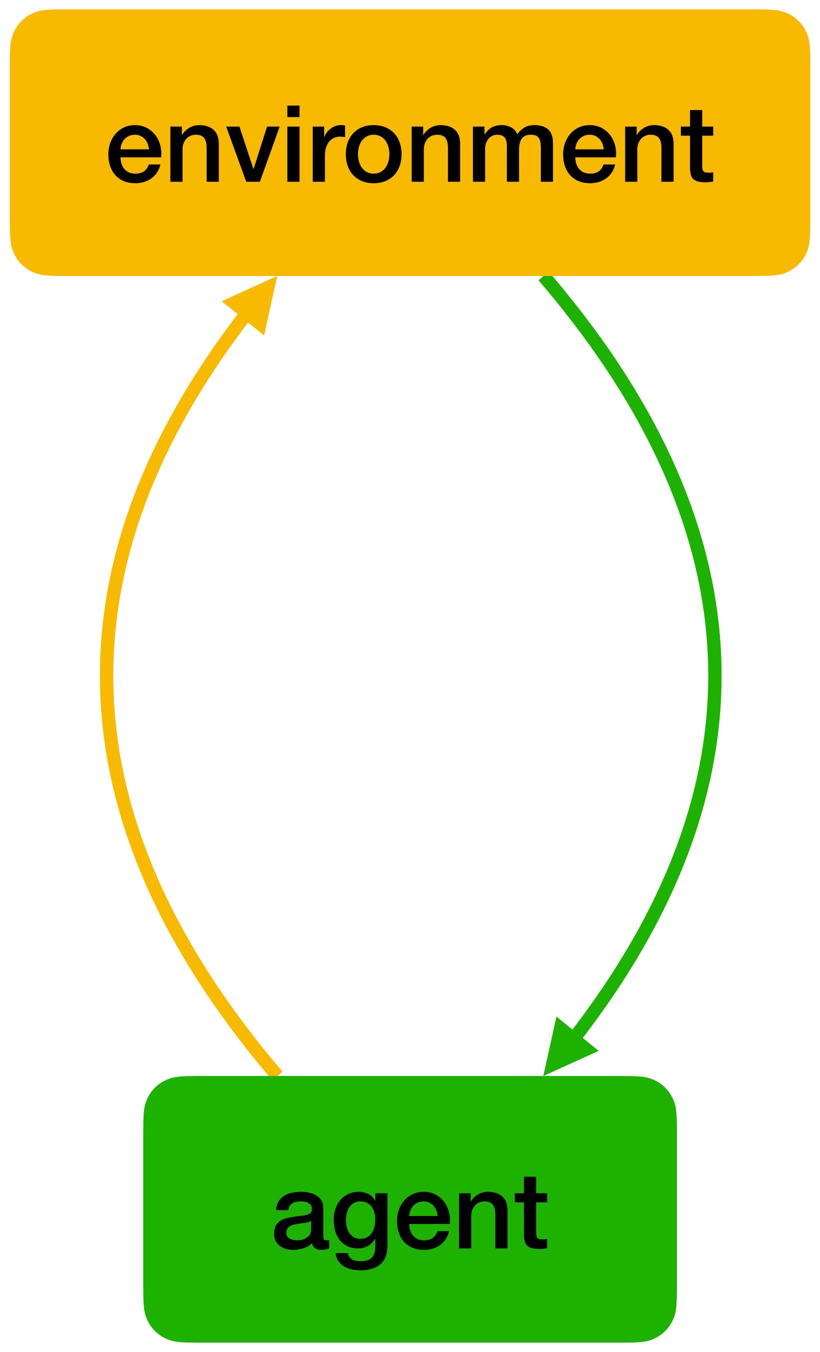






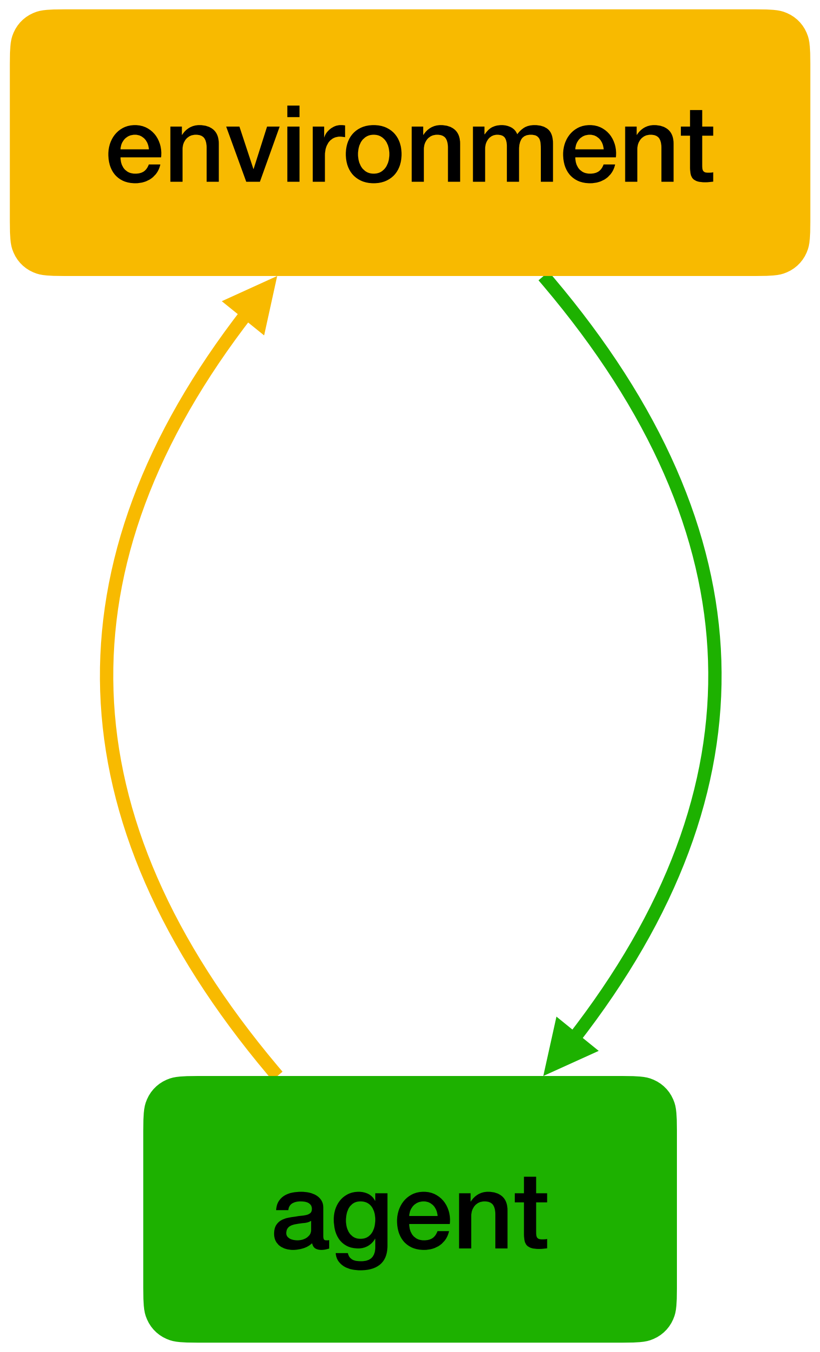
Markov Decision Process (MDP)

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



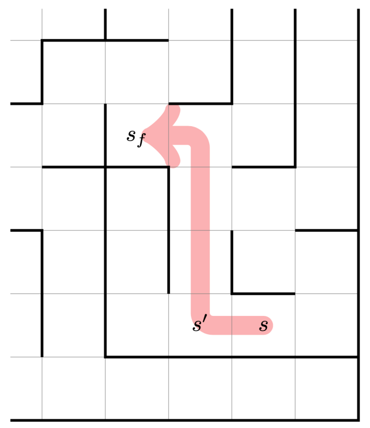
Agent policy

- "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



Trajectories

- The agent's behavior iteratively uses (rolls out) the policy

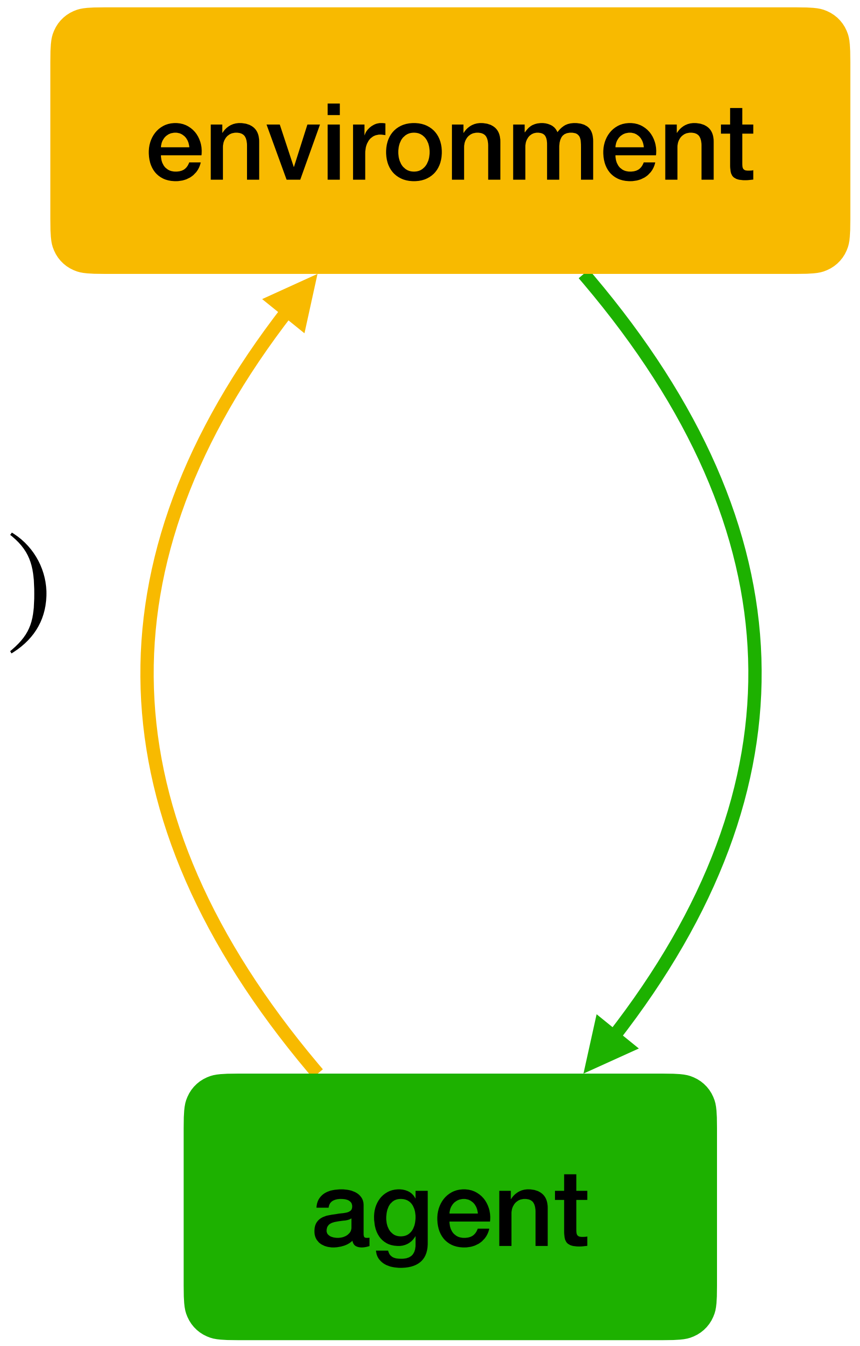


- Trajectory: $\xi = (s_0, a_0, s_2, a_2, \ldots, s_T)$

- MDP + policy induce distribution over trajectories

$$p_\pi(\xi) = p(s_0)\pi(a_0 \,|\, s_0)p(s_1 \,|\, s_0, a_0)\cdots\pi(a_{T-1} \,|\, s_{T-1})p(s_T \,|\, s_{T-1}, a_{T-1})$$

$$= p(s_0)\prod_{t=0}^{T-1}\pi(a_t \,|\, s_t)p(s_{t+1} \,|\, s_t, a_t)$$



- Imitation learning: learn from dataset of expert demonstrations
  - Supervised learning of $\pi(a \,|\, s)$ from "labeled" states (s_t, a_t)

Learning from rewards

- Providing demonstrations is hard
  - Particularly for learner-generated trajectories
- Can the teacher just score learner actions?
  as in online learning
  - 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

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

Discounted returns

- Return = total reward = $R = \sum_{t \geq 0} \gamma^t r(s_t, a_t)$
  - 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)

Other horizon classes

- Finite: $R^T(\xi) = \sum_{t=0}^{T-1} r(s_t, a_t)$
- Infinite: $R^{\text{avg}}(\xi) = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} r(s_t, a_t)$
- Discounted: $R^\gamma(\xi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \qquad 0 \leq \gamma < 1$
- Episodic: $R^{s_f}(\xi) = \sum_{t=0}^{T-1} r(s_t, a_t)$ s.t. $s_T = s_f$

Recap: basic RL concepts

- State: $s \in \mathcal{S}$; action: $a \in \mathcal{A}$; reward: $r(s, a) \in \mathbb{R}$
- Dynamics: $p(s_{t+1} | s_t, a_t)$ for stochastic; $s_{t+1} = f(s_t, a_t)$ for deterministic
- Policy: $\pi(a_t | s_t)$ for stochastic; $a_t = \pi(s_t)$ for deterministic
- Trajectory: $p_\pi(\xi = s_0, a_0, s_1, a_1, \ldots) = p(s_0) \prod_t \pi(a_t | s_t) p(s_{t+1} | s_t, a_t)$
- Return: $R(\xi) = \sum_t \gamma^t r(s_t, a_t) \qquad 0 \leq \gamma < 1$
- Value: $V(s) = \mathbb{E}_{\xi \sim p_\pi}[R | s_0 = s]$
  $Q(s, a) = \mathbb{E}_{\xi \sim p_\pi}[R | s_0 = s, a_0 = a]$

Today's lecture

Basic RL concepts

Behavior Cloning

Better behavior modeling

Alleviating train–test mismatch

Imitation Learning (IL)

- How can we teach an agent to perform a task?
- Often there is an expert that already knows how to perform the task
  - A human operator who controls a robot
  - A black-box artificial agent that we can observe but not copy
  - An agent with different representation or embodiment

- The expert can demonstrate the task to create a training dataset $\mathscr{D} = \{\xi^{(i)}\}_i$
  - Each demonstration is a trajectory $\xi = s_0, a_0, s_1, a_1, \ldots$
  - Then the learner imitates these demonstrations

IL = Learning from Demonstrations (LfD)

- Teacher provides demonstration trajectories $\mathcal{D} = \{\xi^{(1)}, \ldots, \xi^{(m)}\}$
- Learner trains a policy π_θ to minimize a loss $\mathcal{L}(\theta)$
- For example, negative log-likelihood (NLL):

$$
\begin{aligned}
\arg\min_\theta \mathcal{L}_\theta(\xi) &= \arg\min_\theta(-\log p_\theta(\xi)) \\
&= \arg\max_\theta \left(\log p(s_0) + \sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t) + \log p(s_{t+1} \mid s_t, a_t) \right) \\
&= \arg\max_\theta \sum_{t=0}^{T-1} \log \pi_\theta(a_t \mid s_t)
\end{aligned}
$$

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 $\{(s_0^{(1)}, a_0^{(1)}), \ldots, (s_{T_m-1}^{(m)}, a_{T_m-1}^{(m)})\}$
  - Train $\pi_\theta : s \mapsto a$ using supervised learning

observations
+
actions

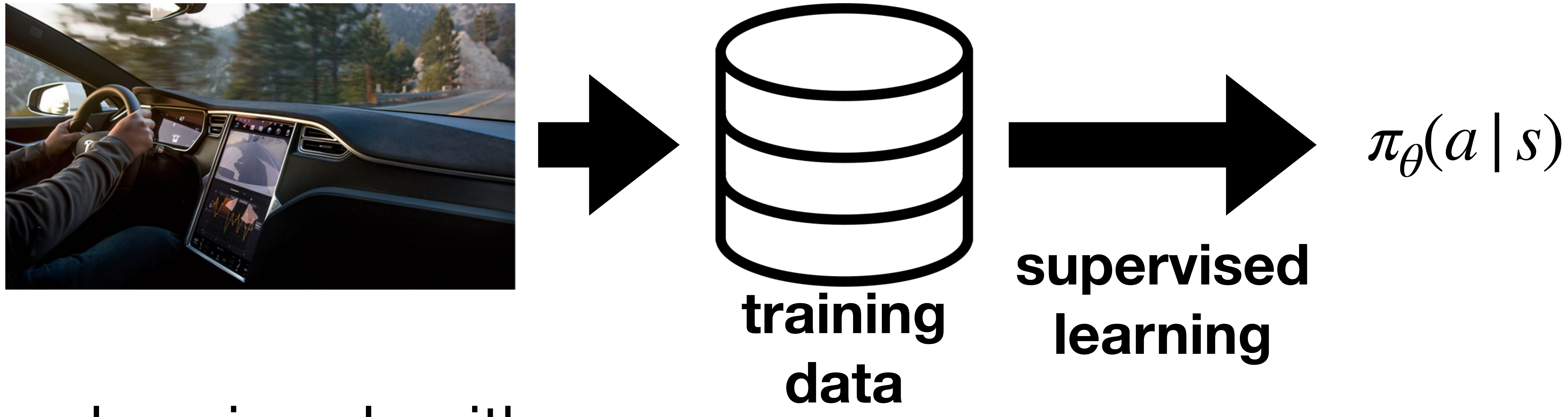
training data

$\mathscr{D} = \{(s_t^{(i)}, a_t^{(i)})\}_{i,t}$

$$\max_\theta \frac{1}{|\mathscr{D}|} \sum_{(s,a)\in\mathscr{D}} \log \pi_\theta(a \mid s)$$

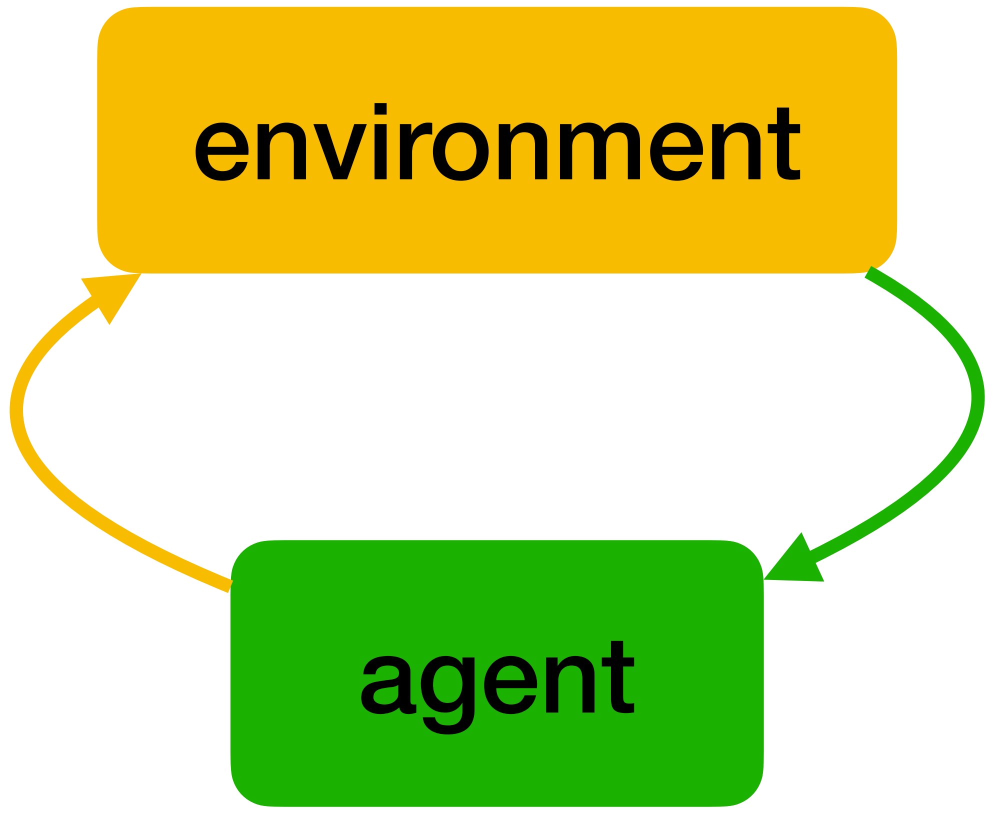
$\pi_\theta(a \mid s)$

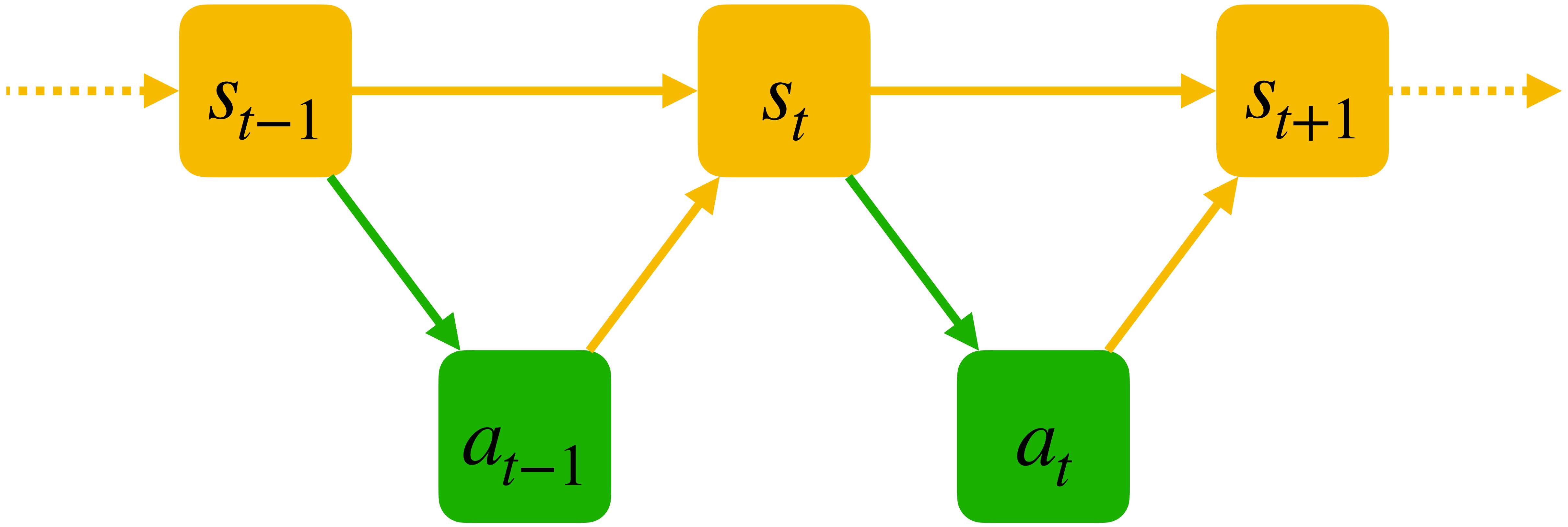
Behavior Cloning (BC)

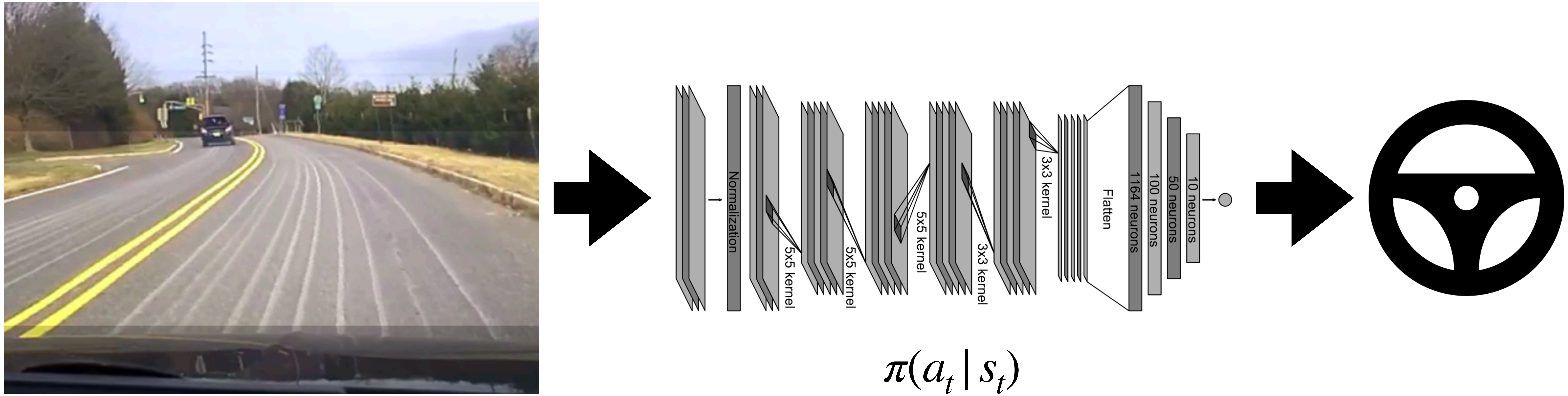


- Benefits:
  - Simple, flexible — can use any learning algorithm
  - Model-free — never need to know the environment
- Drawbacks:
  - Only as good as the demonstrator
  - Only good in demonstrated states — how do we evaluate?
    - Validation loss (on held out data)? Task success rate in rollouts?

A policy is a (stochastic) function





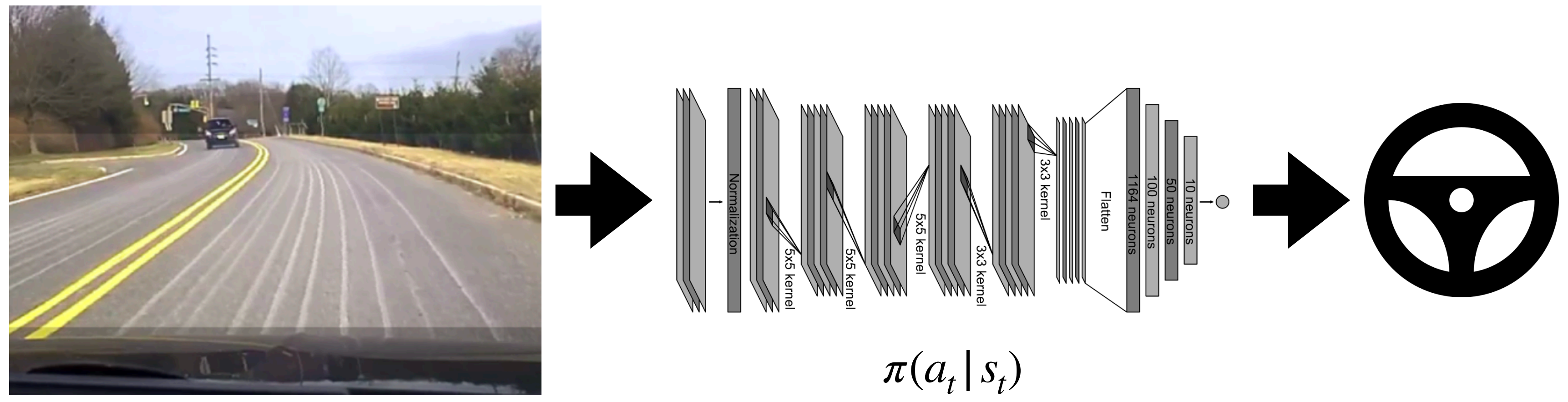


$\pi(a_t \mid s_t)$

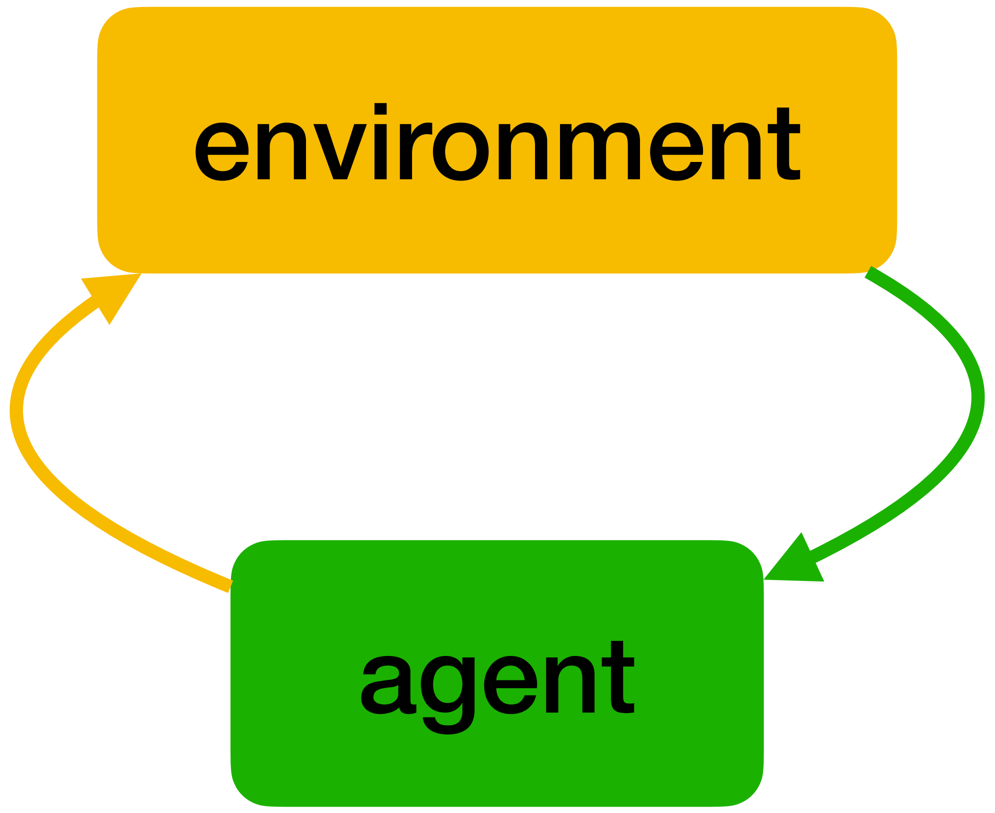
[Bojarski et al., 2016]

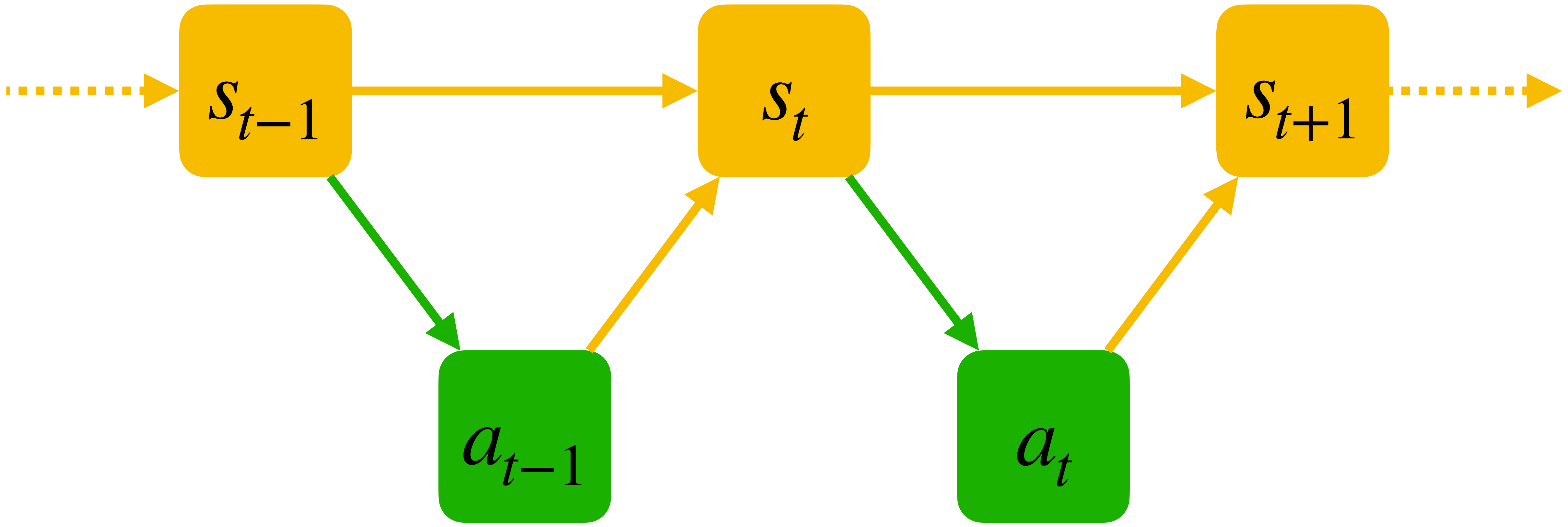
Stochastic policies

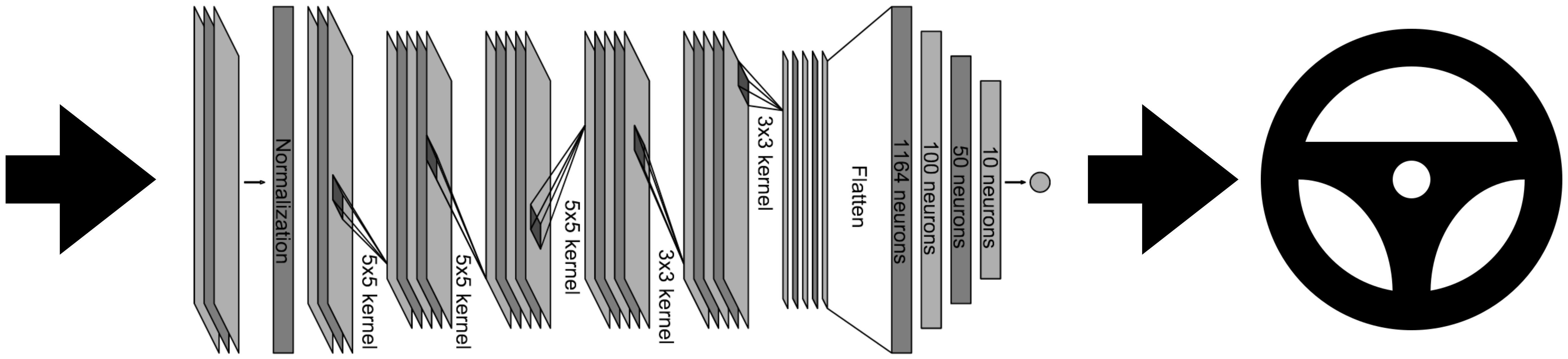
- Learned models are often deterministic functions $f_\theta : x \mapsto y$
- To implement a stochastic policy: output distribution parameters
- Examples:
  - Discrete action space: categorical distribution
    - $\pi_\theta : s \mapsto \{\lambda_a\}_a ; \pi_\theta(a \mid s) = \text{softmax}_a \lambda_a \propto \exp \lambda_a$
  - Continuous action space: Gaussian distribution
    - $\pi_\theta : s \mapsto (\mu, \Sigma) ; \pi_\theta(a \mid s) = \mathcal{N}(\mu, \Sigma)$



A policy is a (stochastic) function

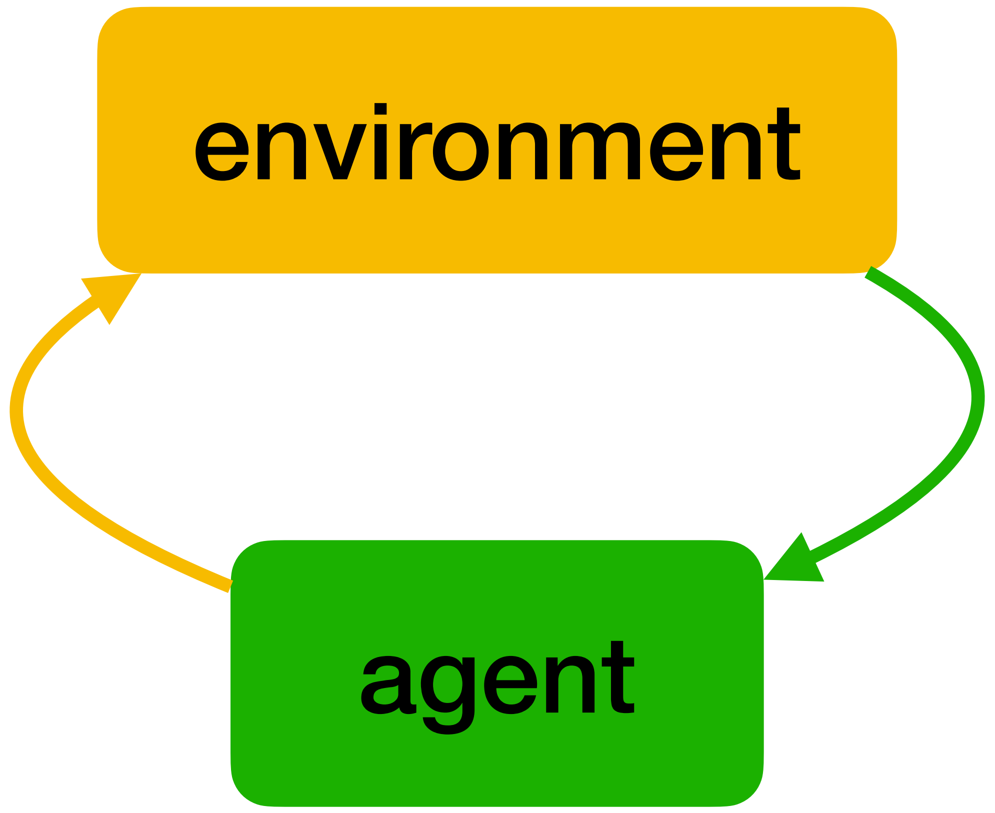


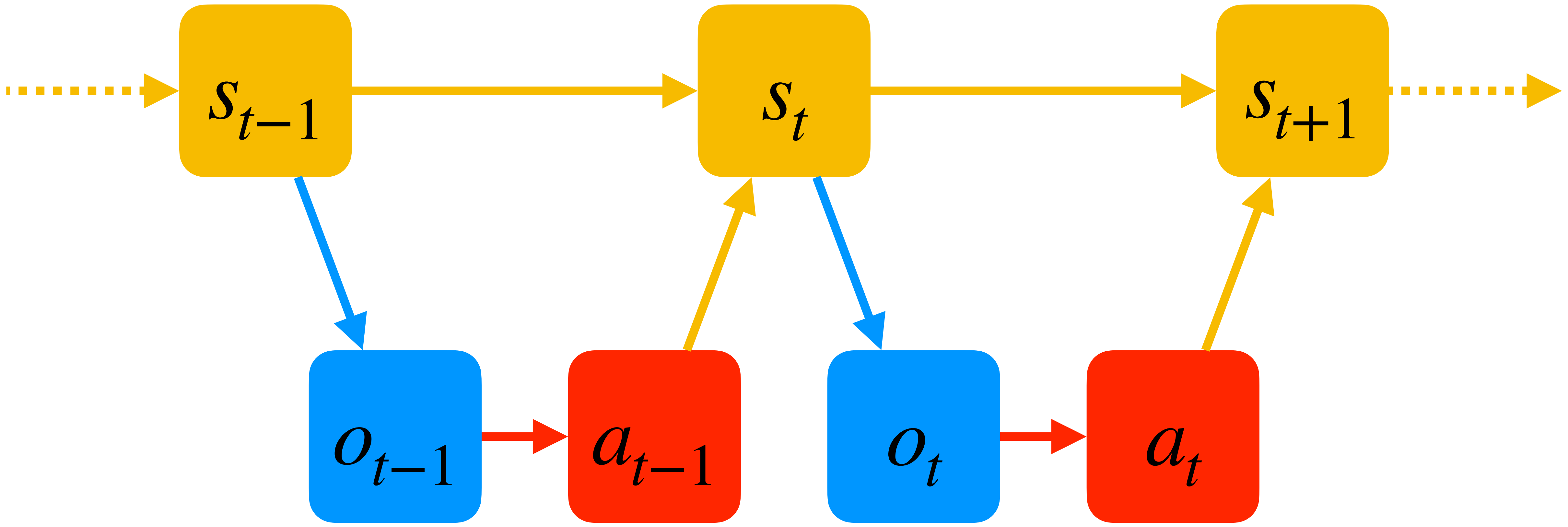


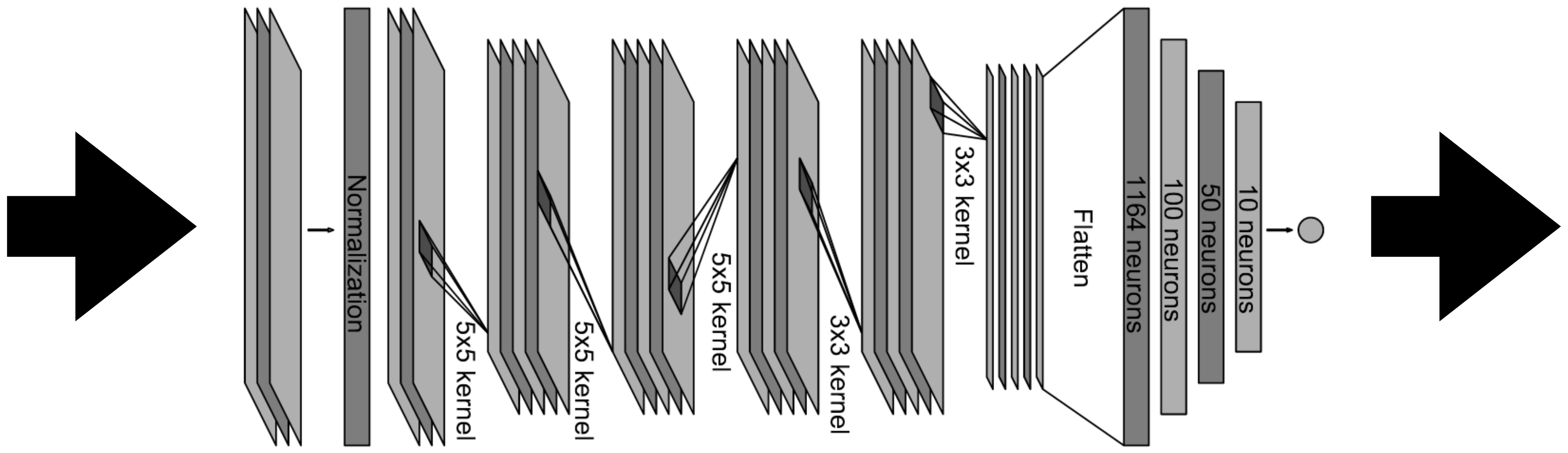


$$\pi(a_t \mid s_t)$$

A policy is a (stochastic) function







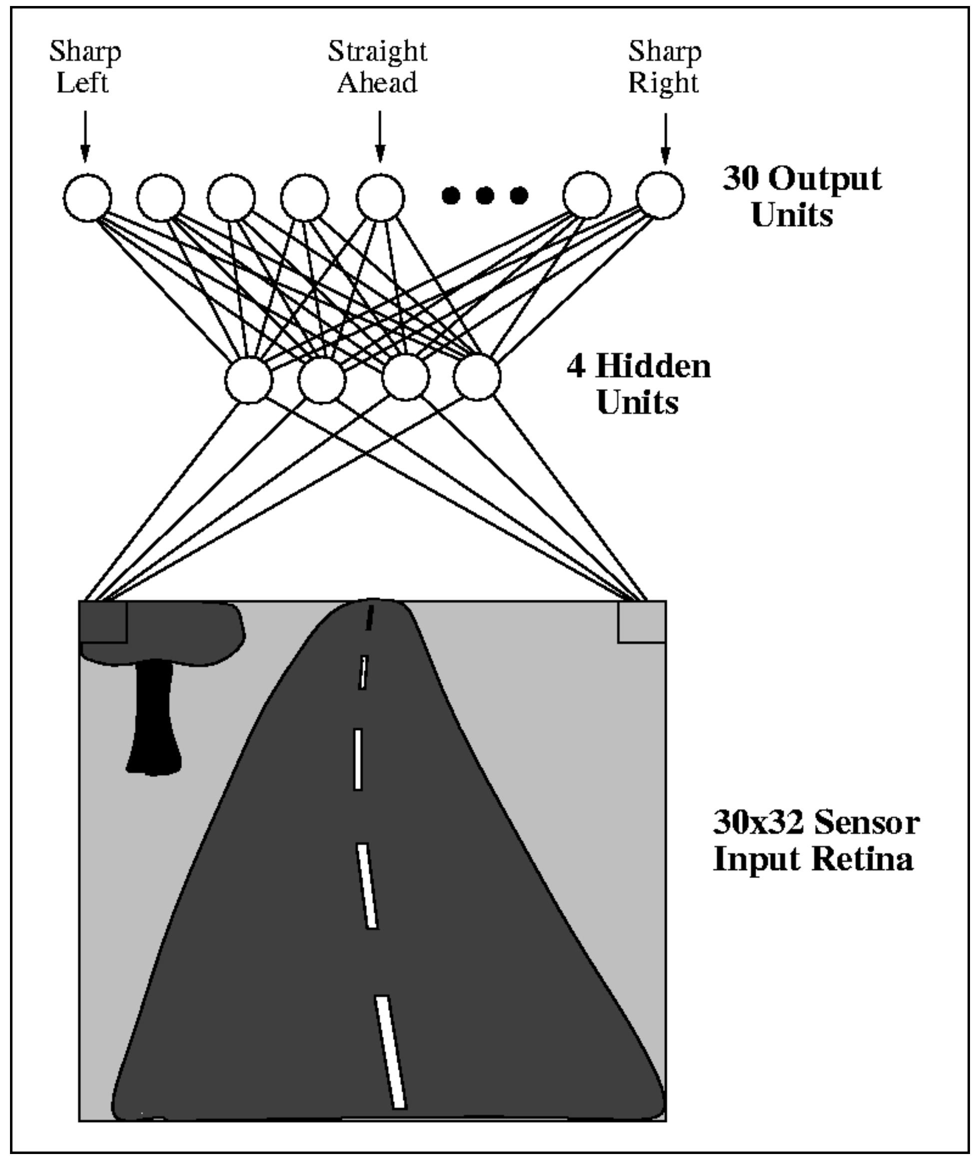
$\pi(a_t \mid o_t)$

observation

action

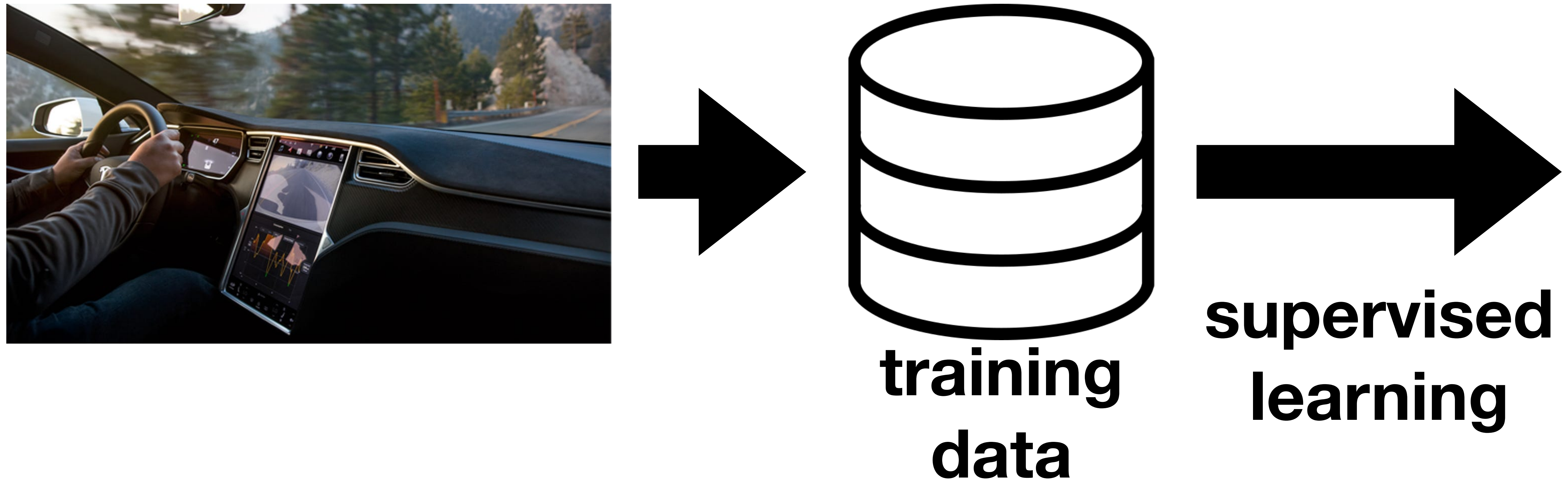
ALVINN

- Autonomous Land Vehicle in a Neural Network (ALVINN, 1989)



[Pomerleau, 1989]

Inaccuracy in BC



$\pi_\theta(a|s)$

- 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 $p_\theta(\xi) \approx p^*(\xi)$
- But errors accumulate over time
  - May reach states not seen in the training dataset

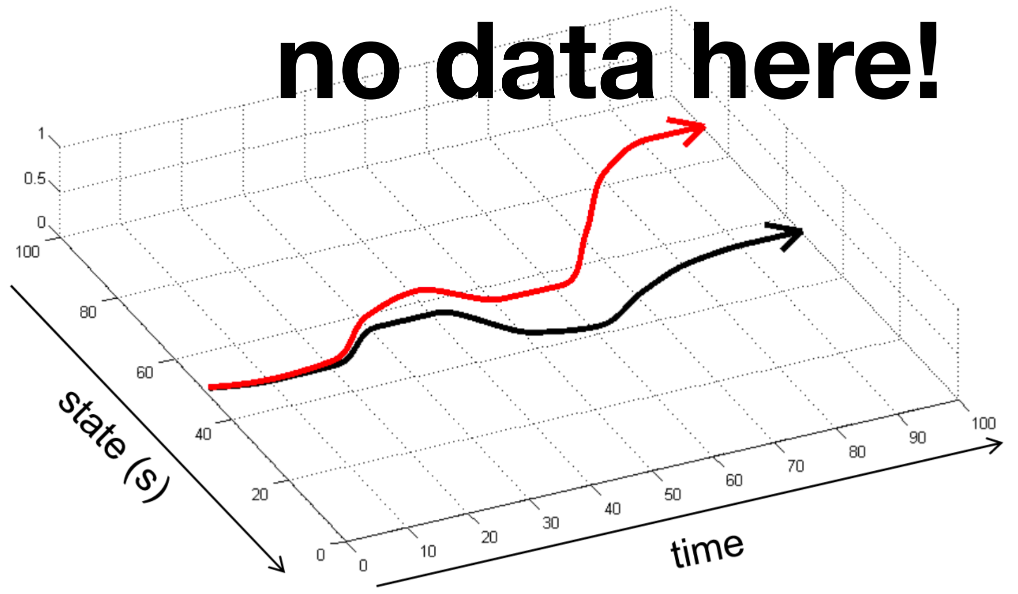
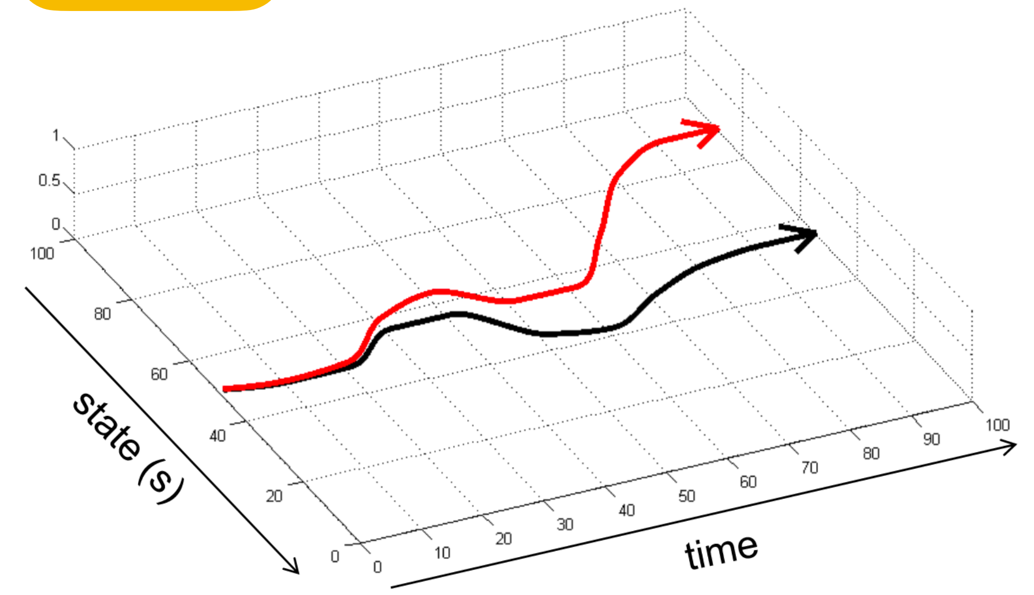


Image: Sergey Levine

The impact of inaccurate dynamics

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



- Errors in learning are unavoidable
- What impact do they have on sequential behavior?
- Bounded one-step error in a dynamical model $\sum_{s'} \left| p_\theta(s' \mid s) - p^*(s' \mid s) \right| \leq \epsilon$
  - Can lead to growing error over time $\sum_{s_t} \left| p_\theta(s_t) - p^*(s_t) \right| \leq \epsilon t$
  - Not too bad by itself, but can drift outside training distribution $\mathscr{D}$

Today's lecture

Basic RL concepts

Behavior Cloning

Better behavior modeling

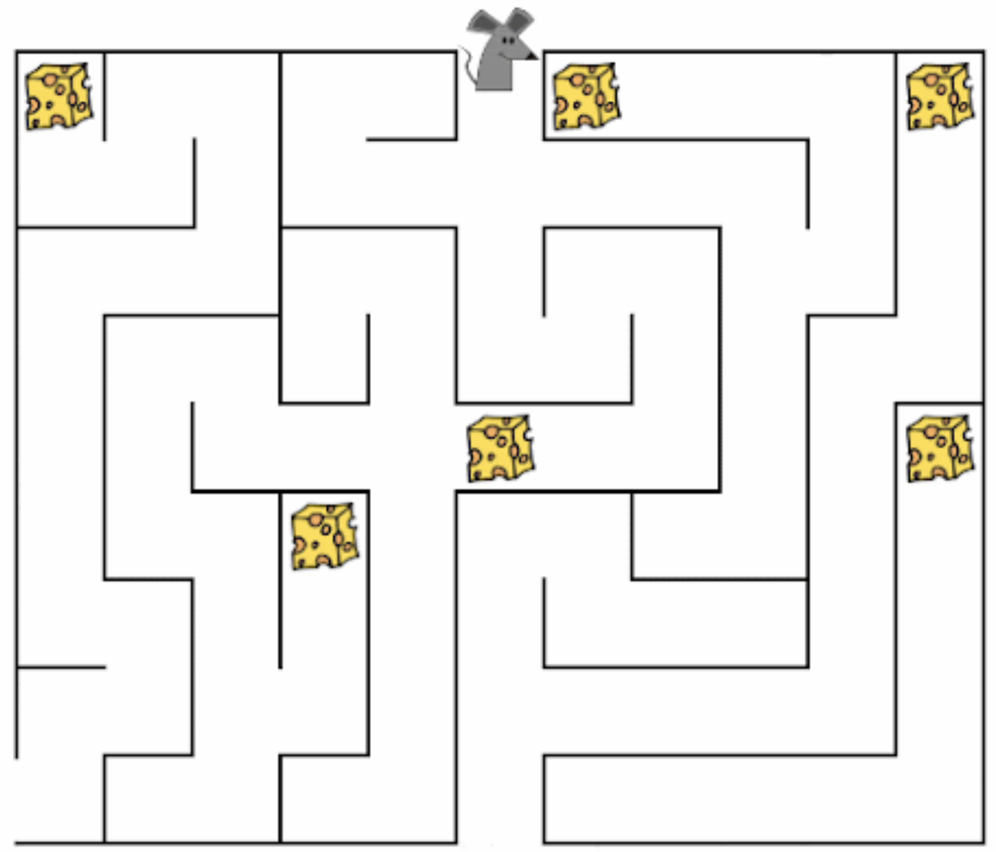
Alleviating train–test mismatch

Modeling other agents is hard

- Is there sufficient data? Demonstrating puts a burden on the teacher
- Are demonstrations correct? Humans are fallible, some supervision is hard
- Are demonstrations consistent? Some tasks can be done in multiple ways
- Is the state partially observable? $o_t \stackrel{?}{=} s_t$
- Are the learner and teacher observations the same? $o_t \stackrel{?}{=} o_t^*$

Inconsistent demonstrations: multiple goals

- What if the task is to reach one of multiple goals?
  - Different episodes can successfully reach different goals
  - We need to train one policy to reach multiple goals



- If we know the goal, condition on it
  - Goal-conditioned policy: $\pi_\theta(a_t \mid s_t, g)$
- More generally: task-conditioned policy $\pi_\theta(a_t \mid s_t, \tau)$
  - Goal = desired final state; but how to represent other kinds of tasks?

Image: puzzlesandriddles.com

Goal-conditioned Behavior Cloning

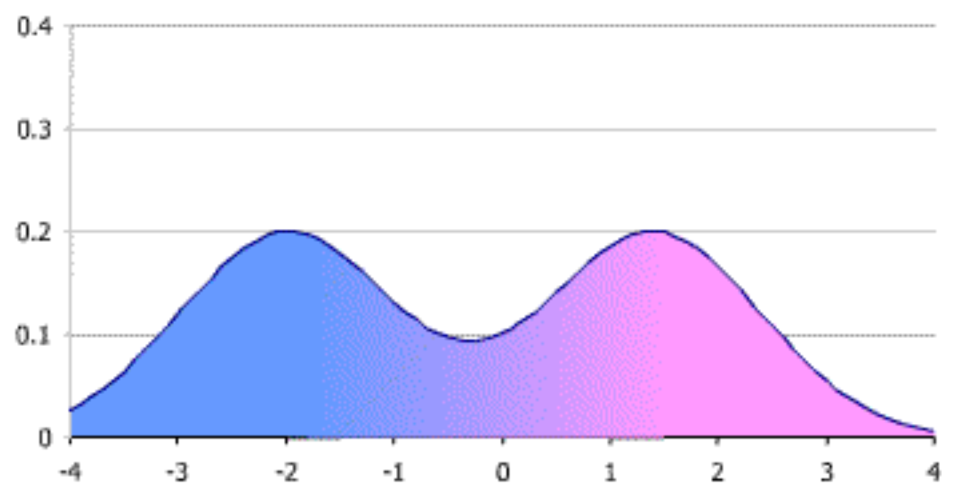
- Can we train a goal-conditioned policy $\pi_\theta(a_t \mid s_t, g)$ from demonstrations?
  - Assume goal = state that the agent should reach
- How can we know the goal in demonstrations $\xi = s_0, a_0, s_1, a_1, \ldots$?
  - Manual labeling? $\mathcal{D} = \{(\xi^{(i)}, g^{(i)})\}_i$
- Hindsight: take each s_t as the goal of the trajectory leading to it

$$s_0, a_0, \ldots, s_{t-1}, a_{t-1}, s_t = g$$

  - Supervised learning of $\pi(a \mid s, g)$ from data points $((s_t, g = s_{t'}), a_t)$ for $t' > t$

Inconsistency due to multimodal behavior

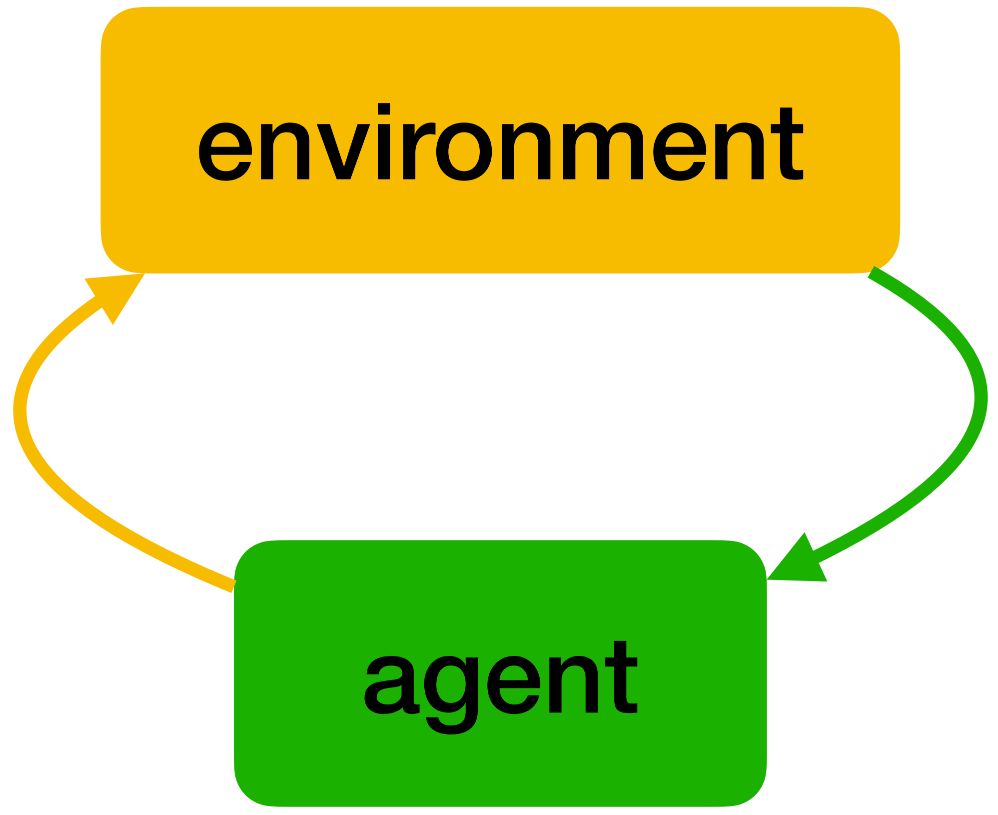
- Goal-conditioning assumes known goals
  - More generally, known behavior modifiers
- Usually, the behavior mode is unknown
  - Need multimodal policy $\pi(a \mid s)$
    - Mixture models (e.g. GMM)
    - Latent-variable models (e.g. normalizing flows)
  - Need to be consistent along a trajectory
    - Condition the policy on memory of past actions $\pi(a_t \mid s_t, a_{\leq t})$

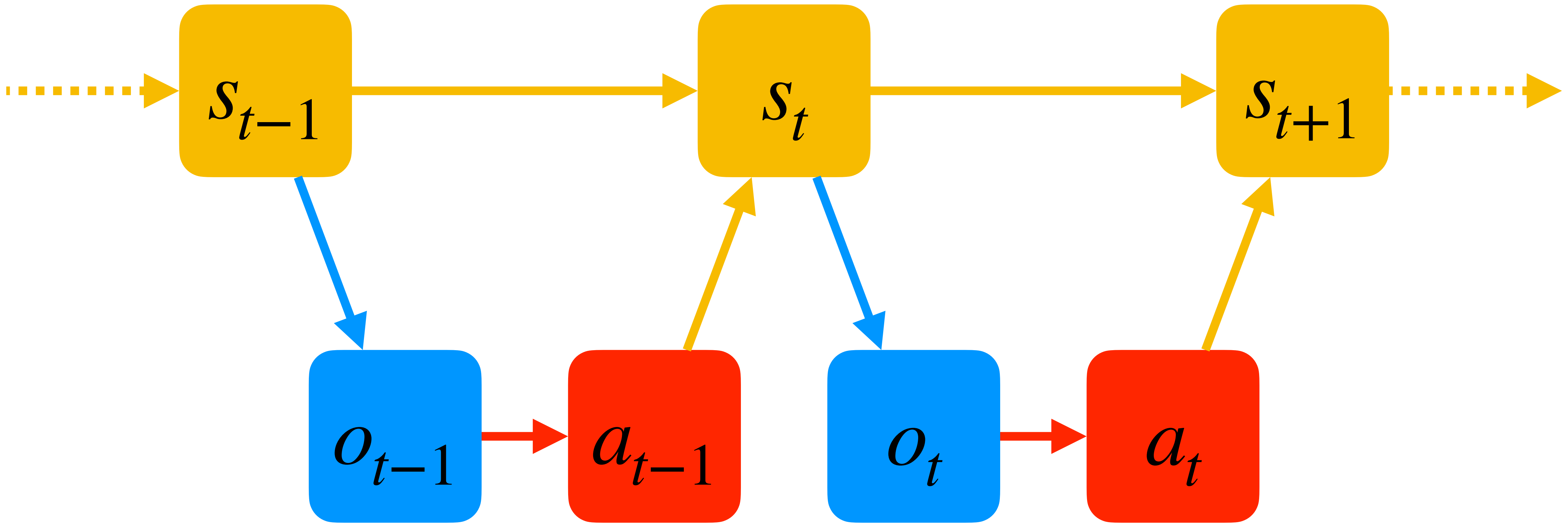


Modeling partially observable behavior

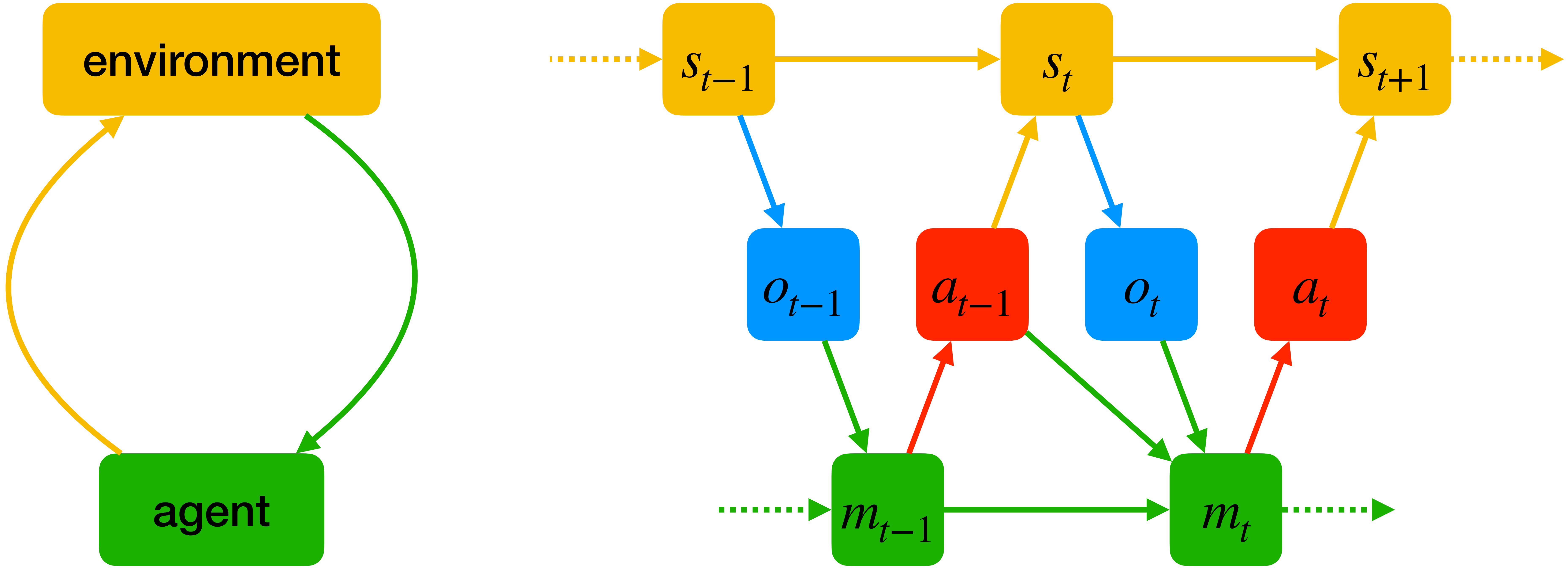
- Partial observations are not Markov
  - Generally, this means $p(o_{t+1} \mid o_t, a_t) \neq p(o_{t+1} \mid o_{\leq t}, a_{\leq t})$
  - Reactive policy $\pi_\theta(a_t \mid o_t)$ may not be optimal
    - May need $\pi_\theta(a_t \mid o_{\leq t})$, or even $\pi_\theta(a_t \mid o_{\leq t}, a_{<t})$; but how?
- Can use RNNs $f_\theta : (h_{t-1}, a_{t-1}, o_t) \mapsto h_t$, or other memory models
- But memory state is latent in demonstrations
  - Modeling memory is hard → prior structure may help; more on this later

Modeling memory





Modeling memory



$\pi_\theta(m_t, a_t \mid m_{t-1}, a_{t-1}, o_t)$

- A common architecture:
  - A recurrent model $m_t = f_\theta(m_{t-1}, a_{t-1}, o_t)$; and an action model $\pi_\theta(a_t \mid m_t)$

Today's lecture

Basic RL concepts

Behavior Cloning

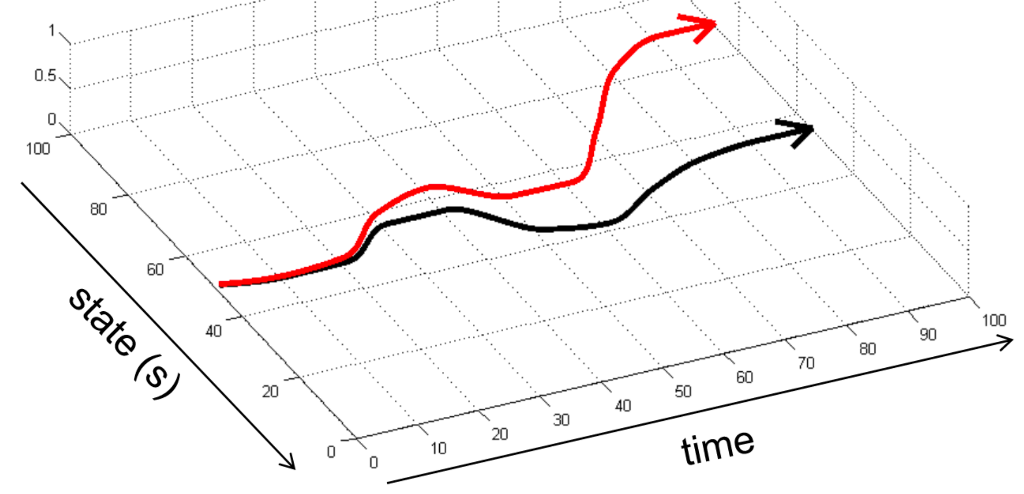
Better behavior modeling

Alleviating train–test mismatch

Alleviating train–test mismatch

- ML promises generalization when training distribution = test distribution
  - But this is challenging in IL: errors accumulate
  - We can quickly get to error states that we haven't seen fixed
  - Train–test distribution mismatch = covariate shift

no data here!

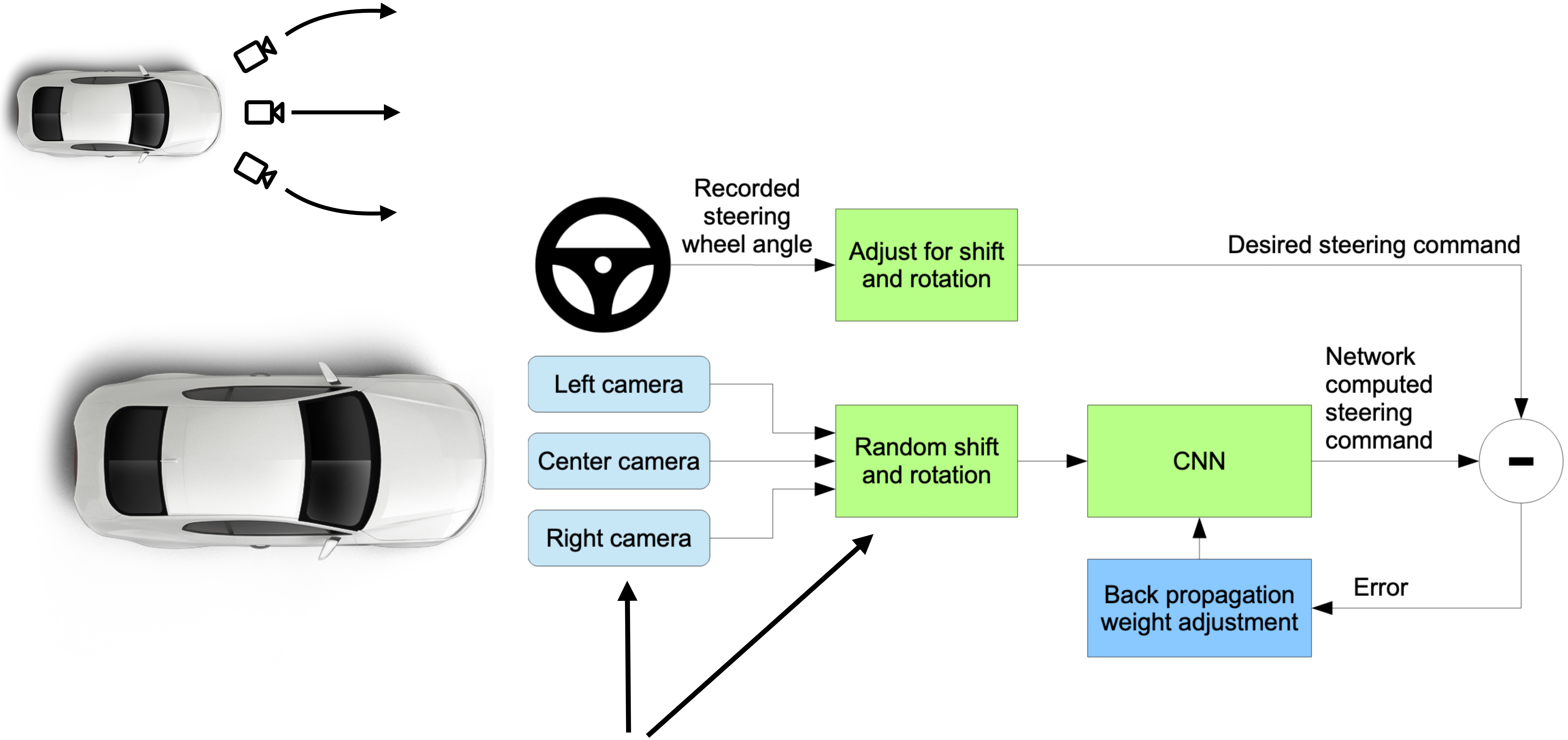


- Ideas:
  - Augment the training dataset to expand the distribution
  - Update train distribution → test distribution
  - Intervene during demonstrations to expand the distribution

Imitation Learning can work

Video: NVIDIA

How did they do it?



augmented data to better cover test distribution

[Bojarski et al., 2016]

DAgger: Dataset Aggregation

- Can we collect demonstration data from the test distribution?
  - We don't know $p_\theta(\xi)$ until we're done training θ
  - But we get closer and closer during training

Algorithm DAgger

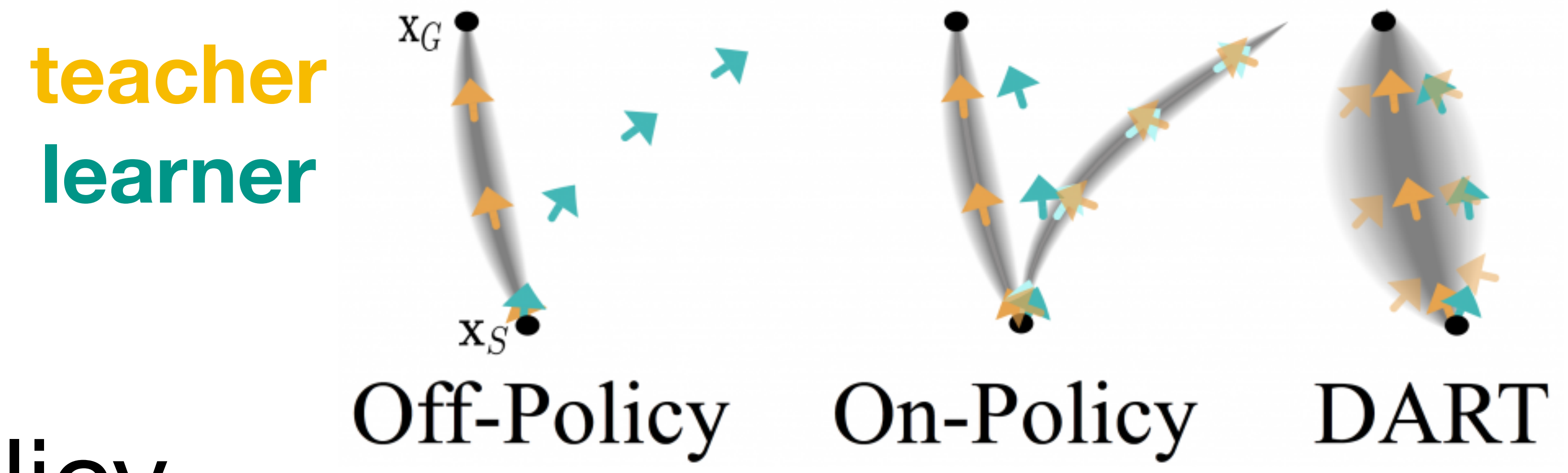
Collect dataset $\mathcal{D}$ of teacher demonstrations $\xi \sim p^*$
repeat
    Train π_θ on $\mathcal{D}$
    Execute π_θ to get $\xi \sim p_\theta$
    $\boxed{\text{Ask teacher to label } (a_t^*|s_t) \sim \pi^*}$
    Aggregate $\{(s_t, a_t^*)\}_t$ into $\mathcal{D}$

but how? challenging...

DAgger demo

Video: Stéphane Ross

DART: Disturbances Augmenting Robot Training



- Off-policy vs. on-policy
  - On-policy = data comes from the learner's current policy
  - Off-policy = data comes from another policy (another agent or past learner)
- In off-policy IL (e.g. BC) learner may go off the teacher's support
- In on-policy IL (e.g. DAgger) learner initially goes off, until corrected
- DART: increase the data support by injecting noise during demonstrations
  - Force teacher into slight-error states, to see how they are fixed

[Laskey et al., 2017]

DART

- Noise = perturbation of actions $q(\tilde{a} \,|\, a)$
  - New effective dynamics: $\tilde{p}(s' \,|\, s, a) = \sum_{\tilde{a}} q(\tilde{a} \,|\, a) p(s' \,|\, s, \tilde{a})$
  - For example, in continuous actions: $\tilde{a} = a + \epsilon; \quad \epsilon \sim \mathcal{N}(0,\Sigma)$

Algorithm DART

repeat

Collect dataset $\mathcal{D}$ of teacher demonstrations $\xi \sim \tilde{p}^*$

Train π_θ on $\mathcal{D}$

Update noise q such that p_θ is better supported by $\tilde{p}^*$

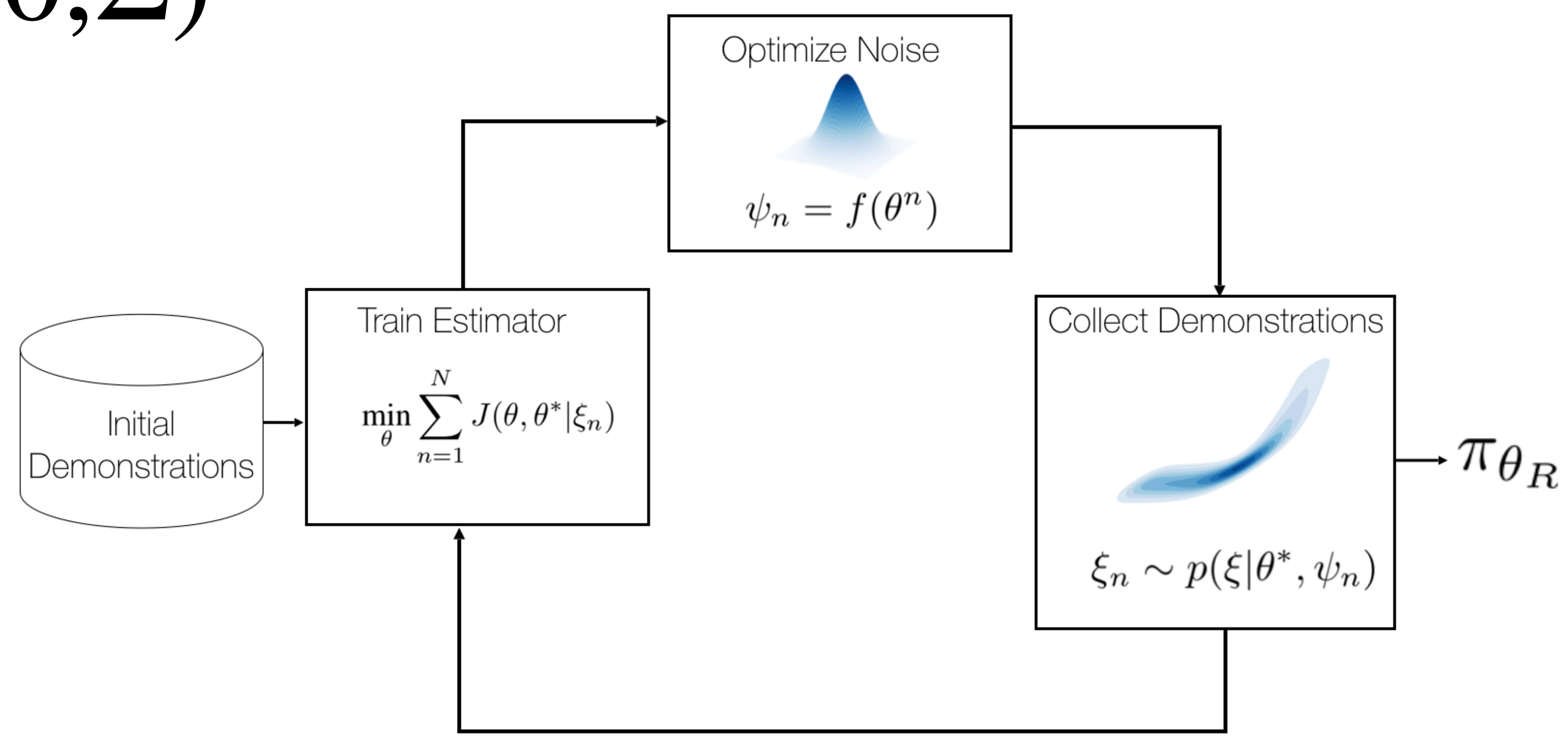


Image: Michael Laskey

Grasping task

Behavior Cloning

DART

[Laskey et al., 2017]

Recap

- Imitation Learning = Learning from Demonstrations
  - Learn policy $\pi(a \mid s)$ from teacher demonstrations
- Behavior Cloning: supervised learning
  - Minimize loss, e.g. NLL, on training set of trajectories
- Accurate imitation is crucial
  - Improve imitation through goal-conditioning, multimodal policies, memory, etc.
- Errors accumulate and cause train–test distribution mismatch
  - Can be alleviated through augmentation, on-policy data collection, noise injection