# DAgger: Interactive Experts and No-Regret Learning

Sanjiban Choudhury

Cornell Bowers CIS Computer Science

# Let's travel to the INFINITE data limit!

*The Three Regimes of Covariate Shift*

# Easy

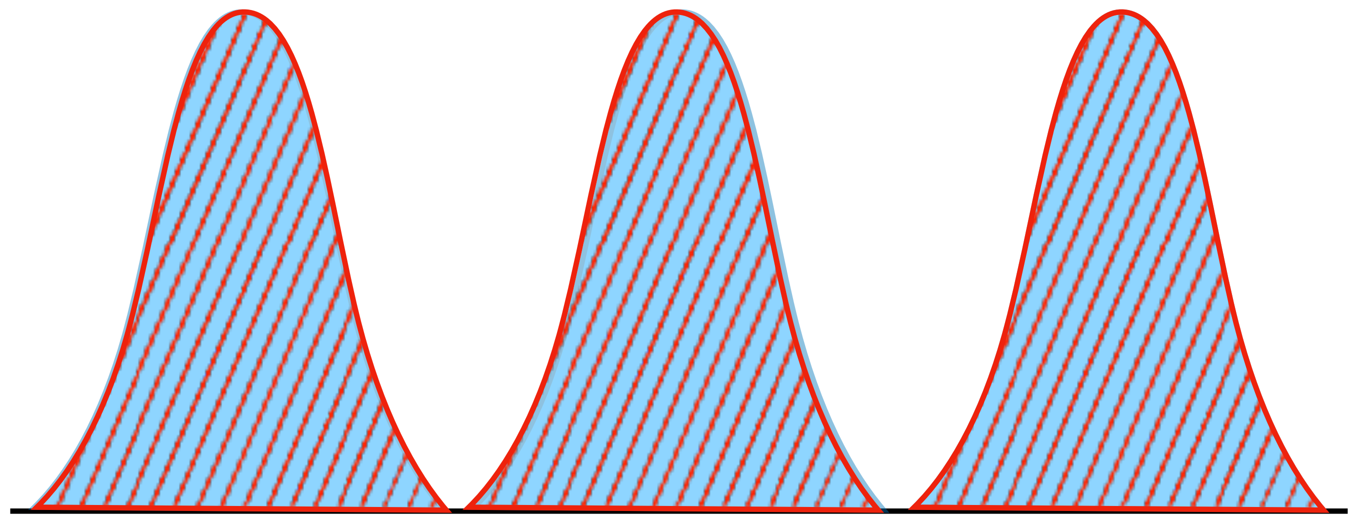
$$\text{Expert } \rho^{\pi^E}(s) \quad \approx \quad \text{Learner } \rho^{\pi}(s)$$

**Setting**

Expert is realizable

$$\pi^E \in \Pi$$

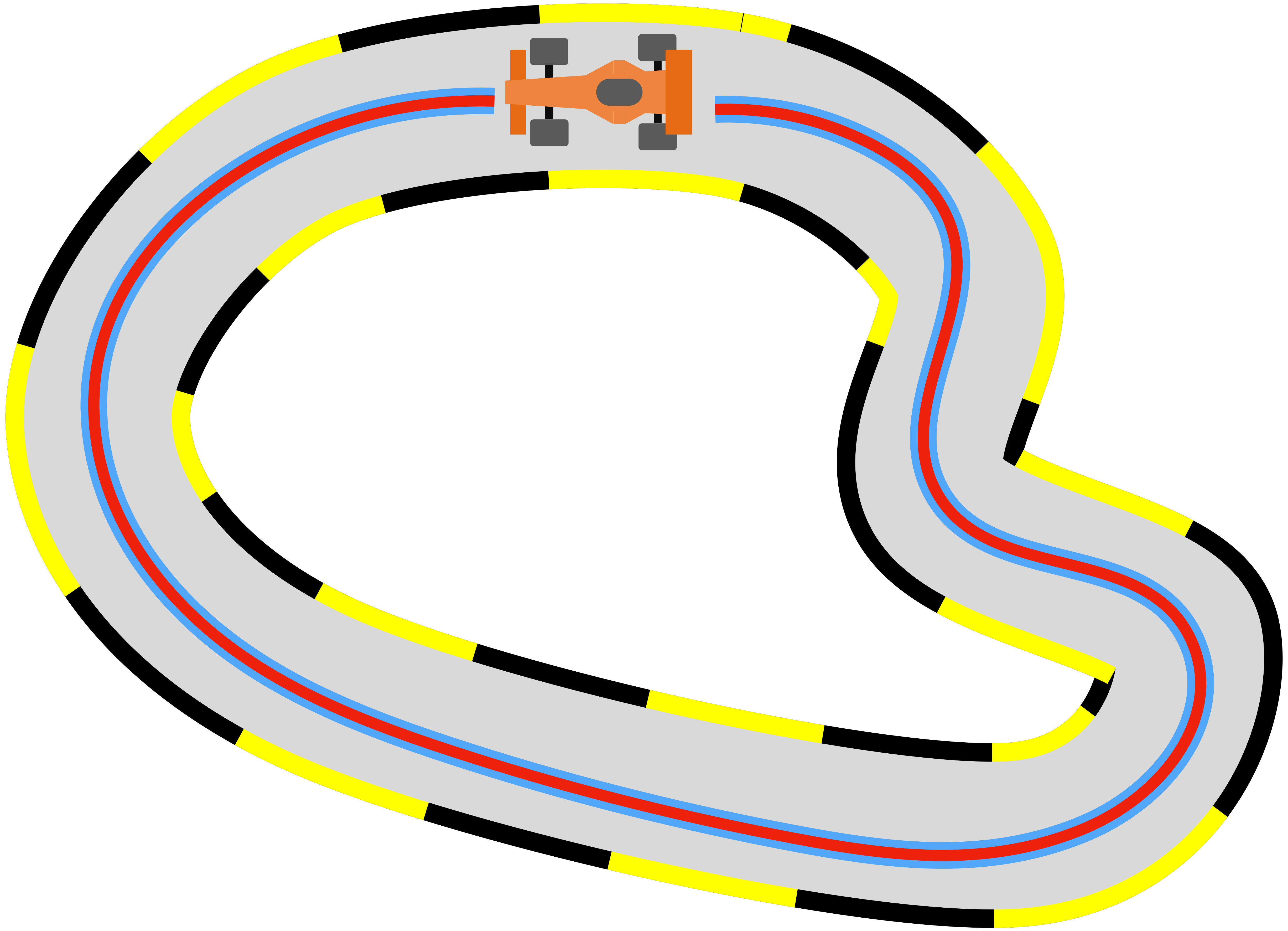
As $N \to \infty$, drive down $\epsilon = 0$ (or Bayes error)



**Solution**

Nothing special.

Collect lots of data and do Behavior Cloning



## Easy

## Hard

**Setting**

Expert is realizable
$\pi^E \in \Pi$

As $N \to \infty$, drive down
$\epsilon = 0$ (or Bayes error)

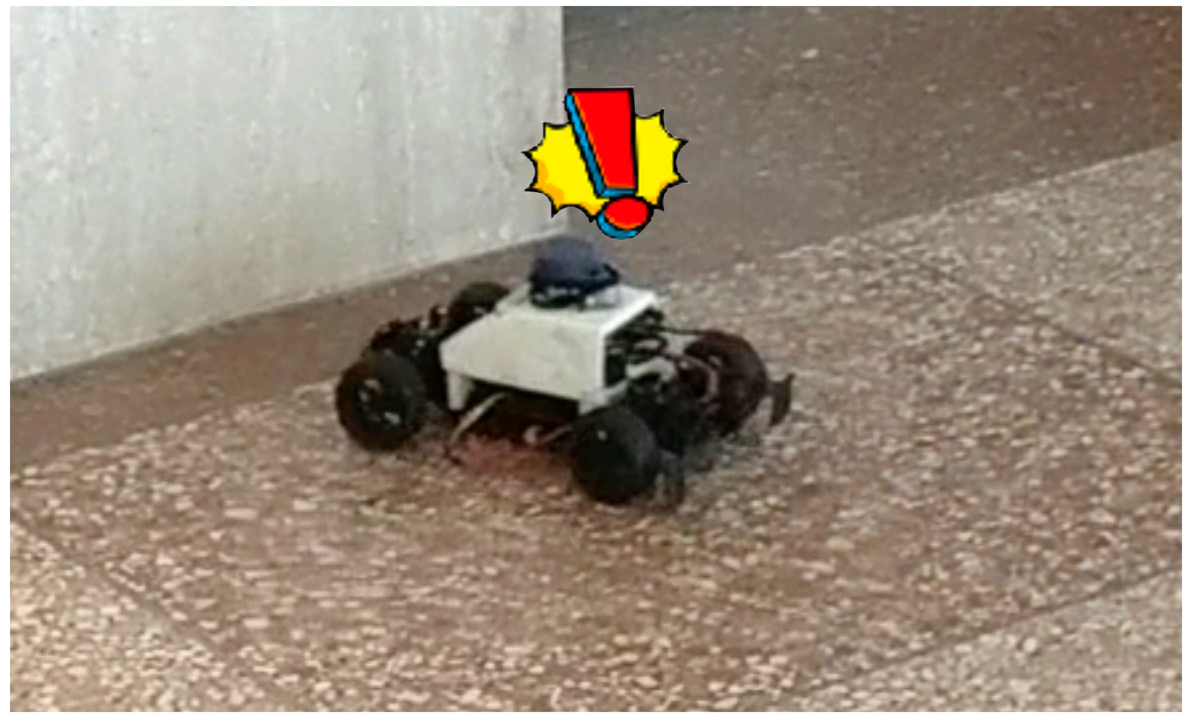
Non-realizable expert +
limited expert support

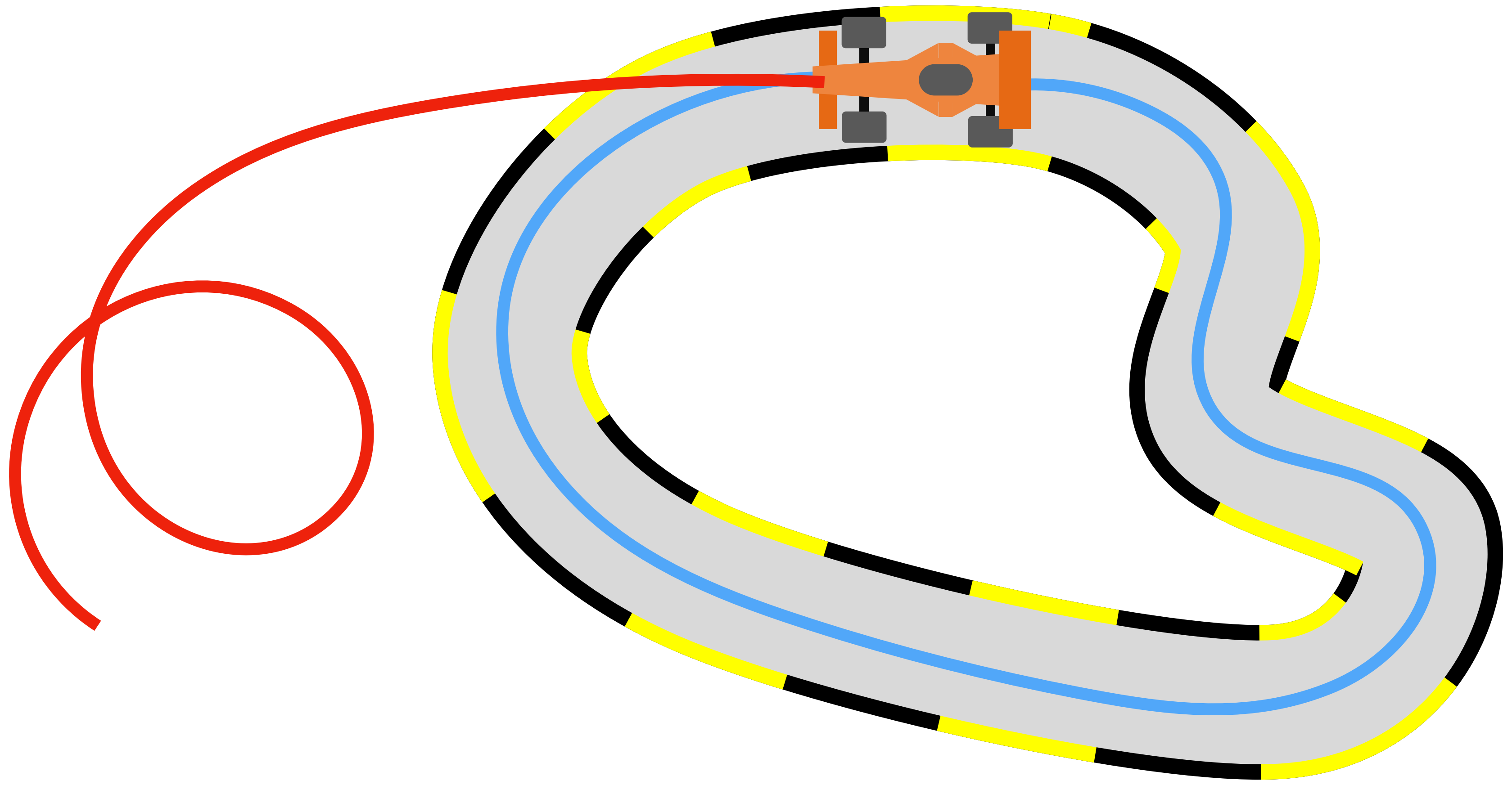


**Solution**

Nothing special.
Collect lots of data and
do Behavior Cloning

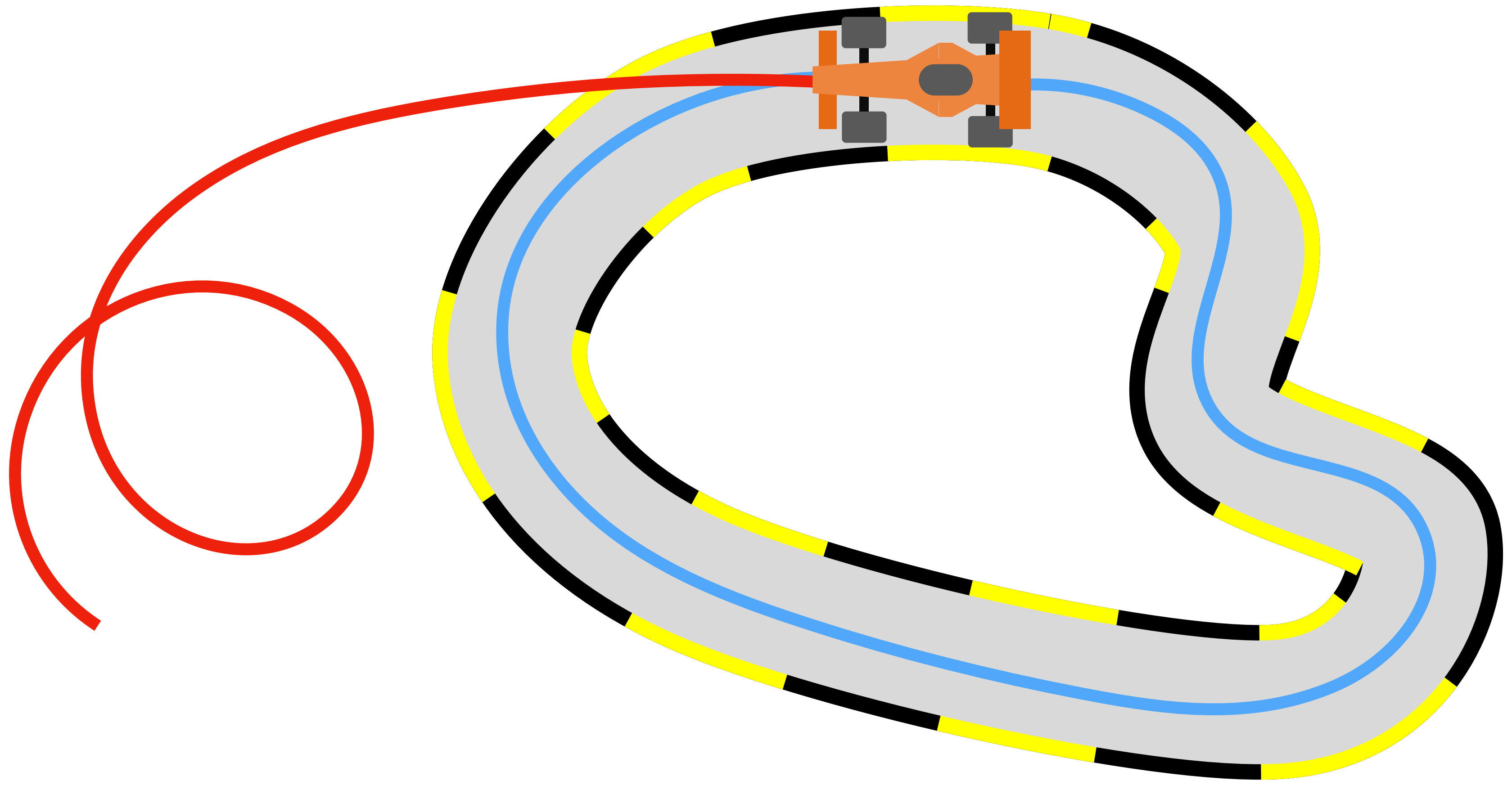
# Non-realizable expert + limited support?



No label for what to do
in this state!

# Non-realizable expert + limited support?



**Hard**

Behavior Cloning
compounds in error $O(\epsilon T^2)$

[Ross & Bagnell '10]

Setting

Solution

## Easy

Expert is realizable

$\pi^E \in \Pi$

As $N \to \infty$, drive down

$\epsilon = 0$ (or Bayes error)



Nothing special.

Collect lots of data and

do Behavior Cloning

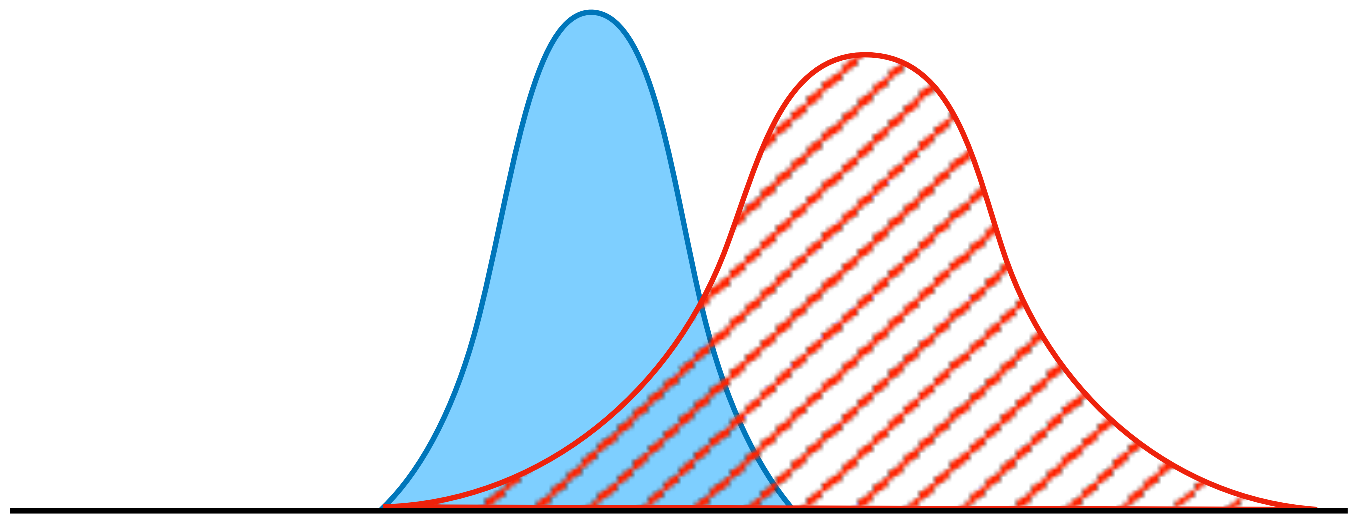
## Hard

Non-realizable expert +

limited expert support

Even as $N \to \infty$,

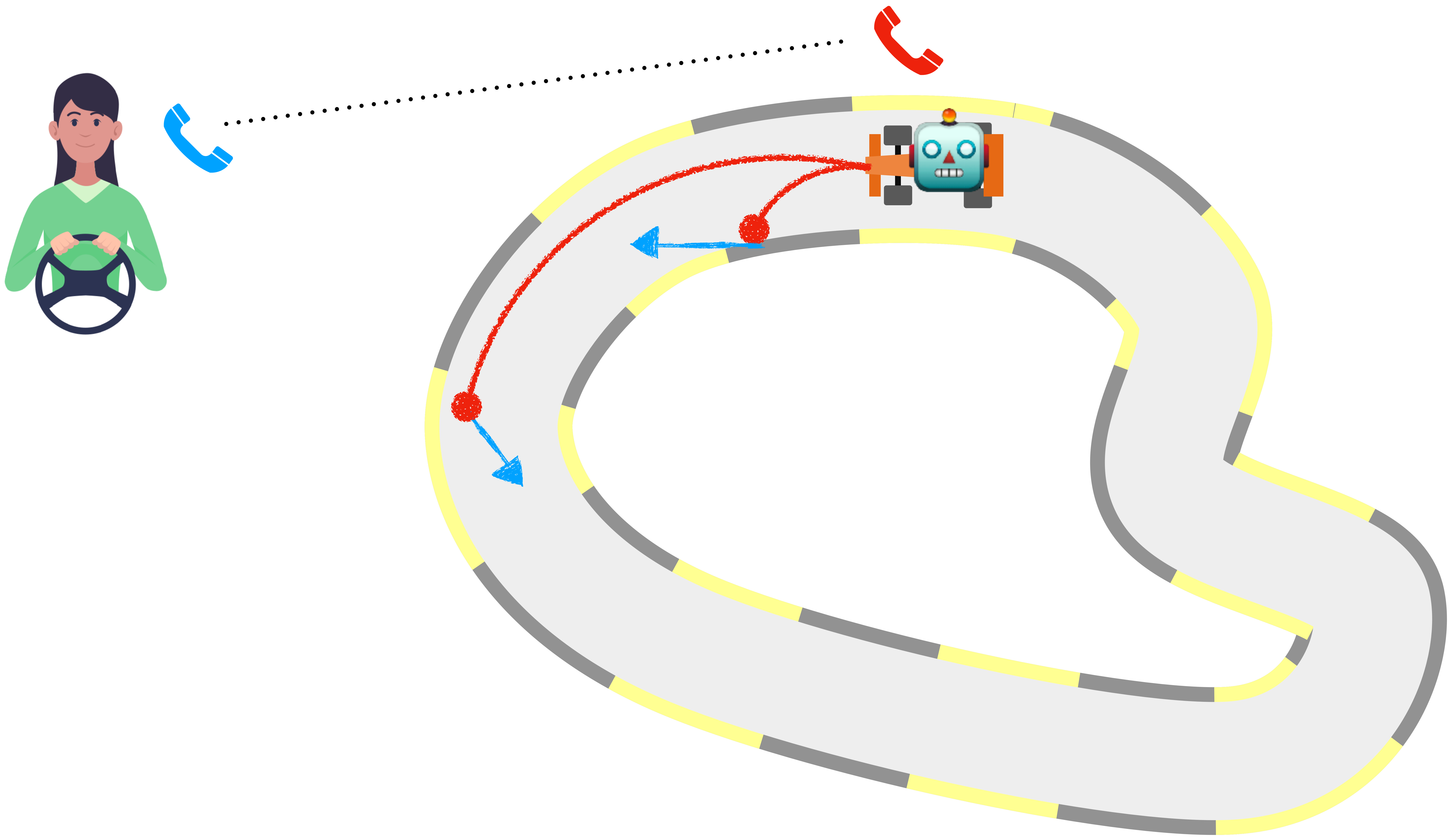
behavior cloning $O(\epsilon T^2)$



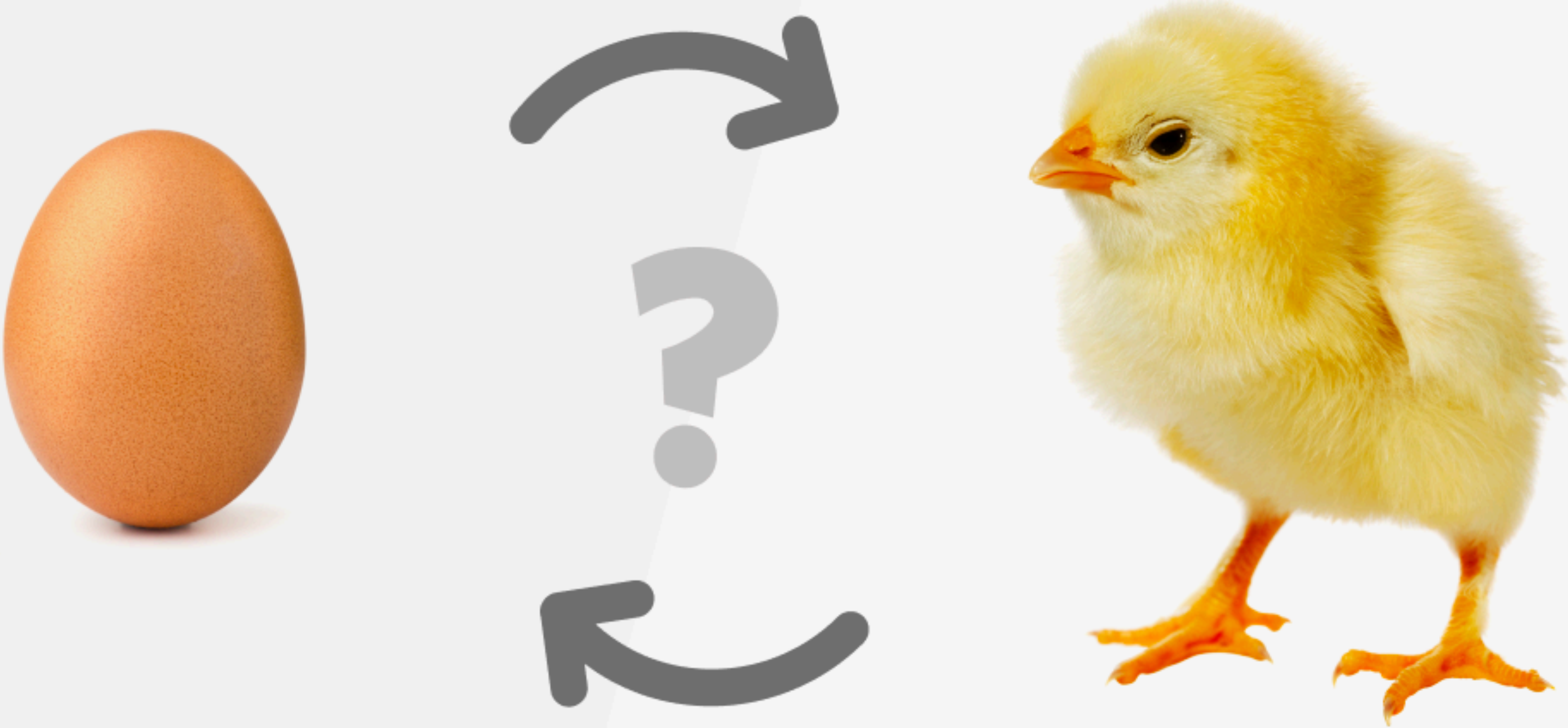
?

Why can't we just collect data $(s, a^*)$ on the distribution of states the learner visits?

# Introducing an *interactive* expert!



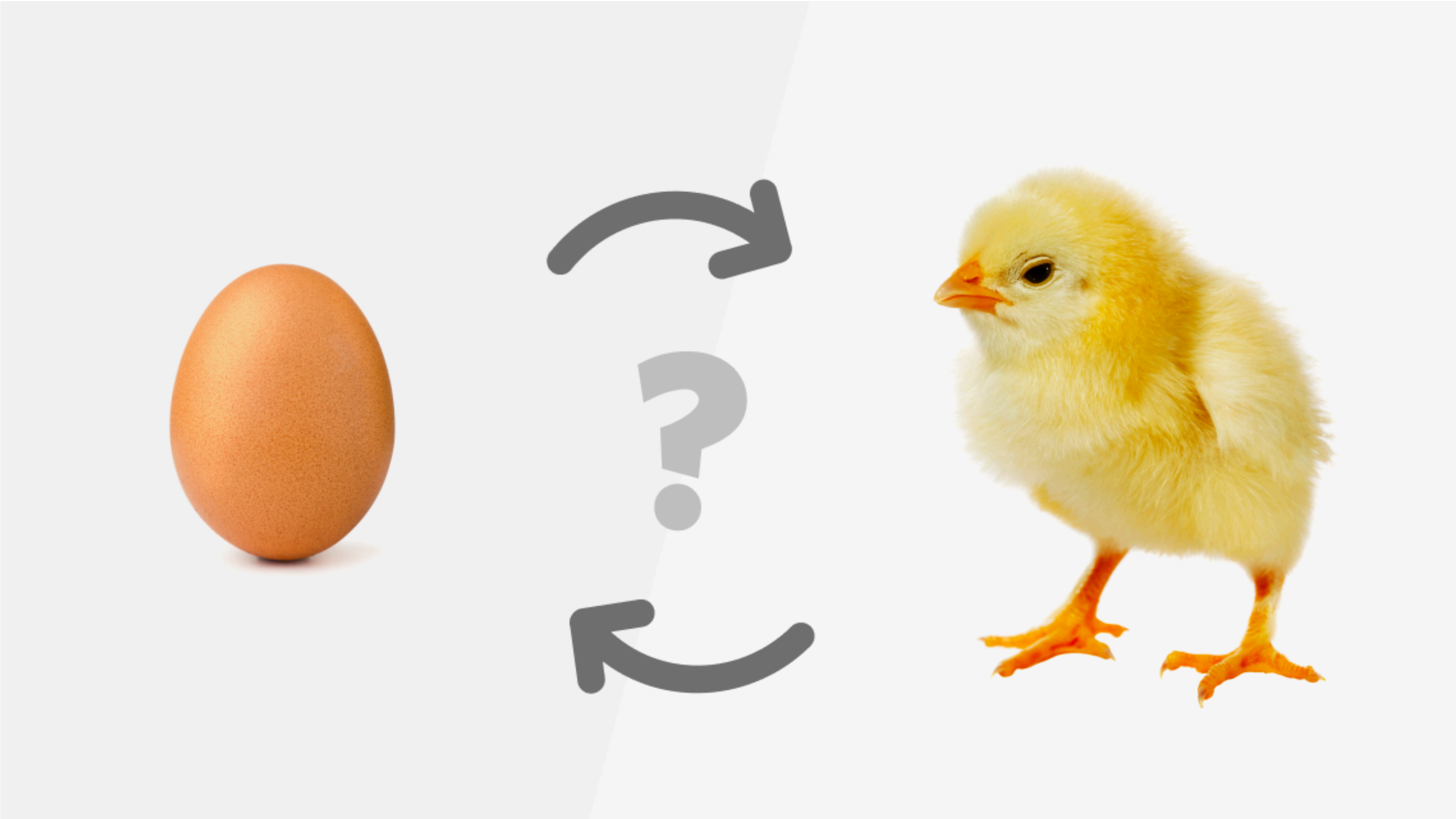
# To know the distribution, you need a learner
# To train a learner, you need a distribution



# Activity!

# Brainstorm

How can we solve the chicken and egg problem, i.e. train the learner on a distribution of states it visits?



# An *embarrassingly* simple algorithm: FORWARD

Idea: Train a different learner policy at every timestep by interactively querying expert

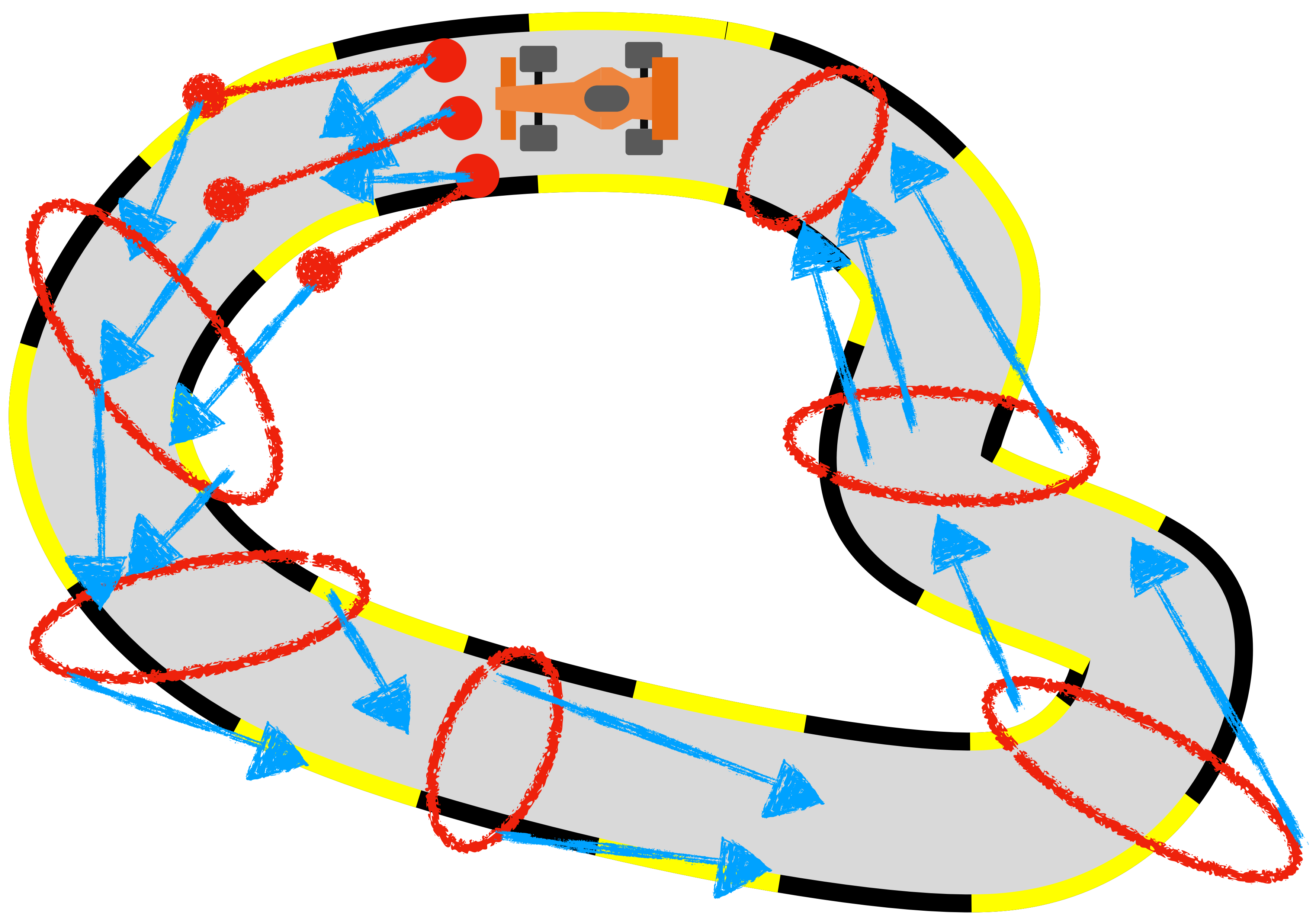
Get start state samples $s_0 \sim d^0(\,.\,)$

for t = 0 … T-1

Query interactive expert to get
$a_t^* = \pi^*(s_t)$

Train a learner policy at time $t$
$\pi^t = \text{Train}(s_t, a_t^*)$

Execute learner policy $\pi^t$ to get
next state samples $s_{t+1} \sim d_\pi^{t+1}(\,.\,)$



But what if we want ONE policy?

# DAGGER

Episode IV

A NEW HOPE

# DAGGER: A meta-algorithm for imitation learning

## A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning

**Stéphane Ross**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
stephaneross@cmu.edu

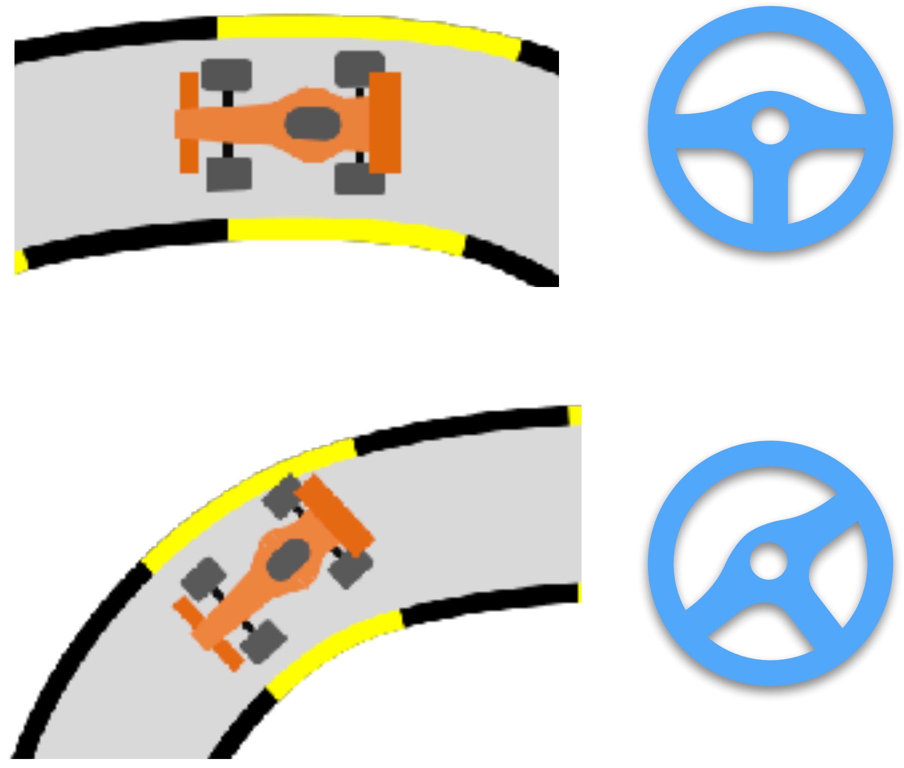
**Geoffrey J. Gordon**
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
ggordon@cs.cmu.edu

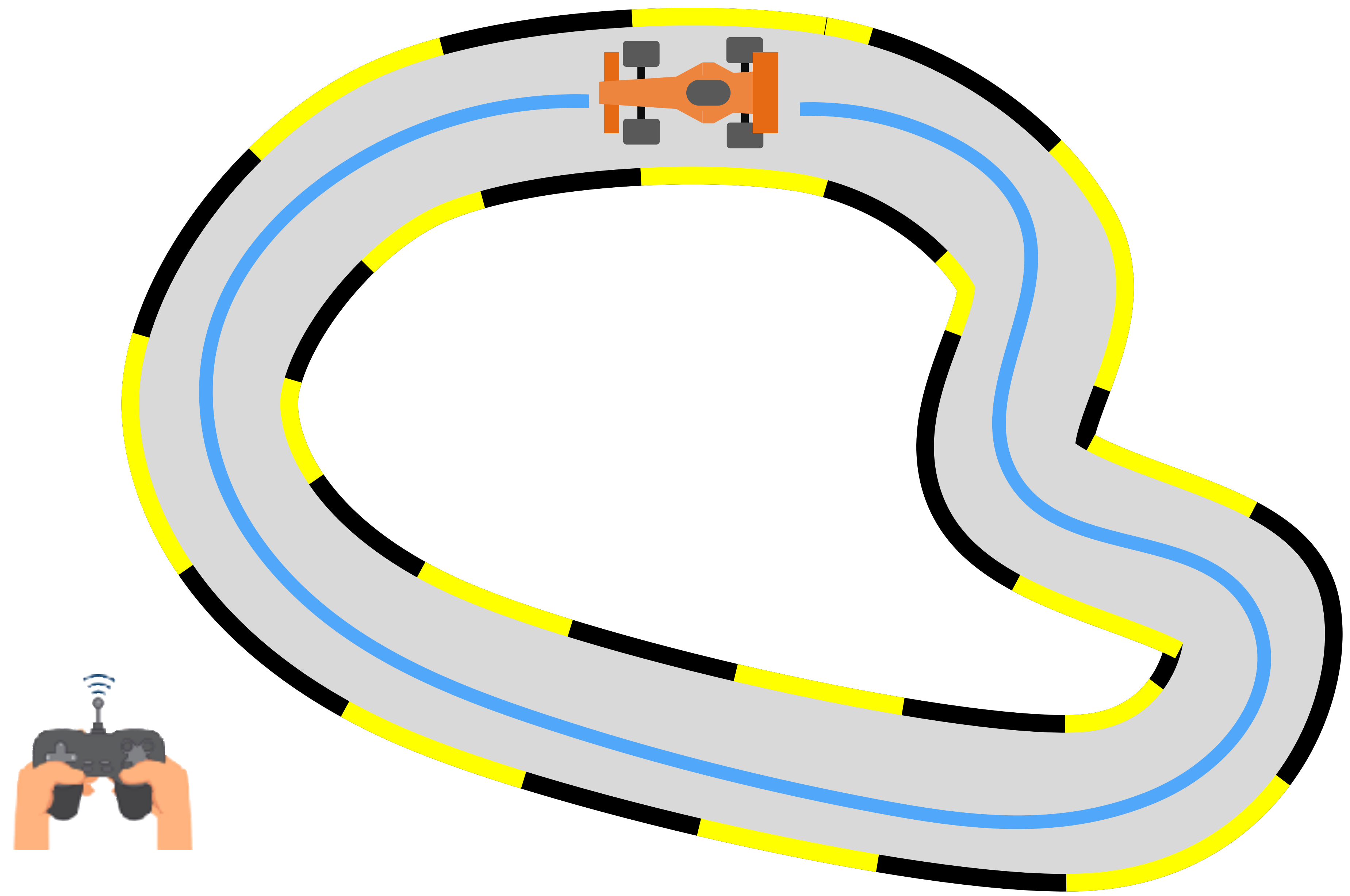
**J. Andrew Bagnell**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dbagnell@ri.cmu.edu

[Ross et al'11]

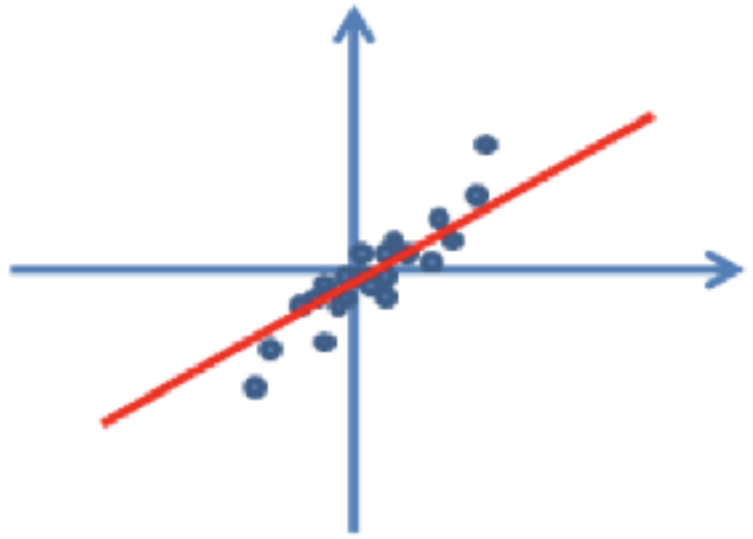
# DAgger: Iteration 0

Data



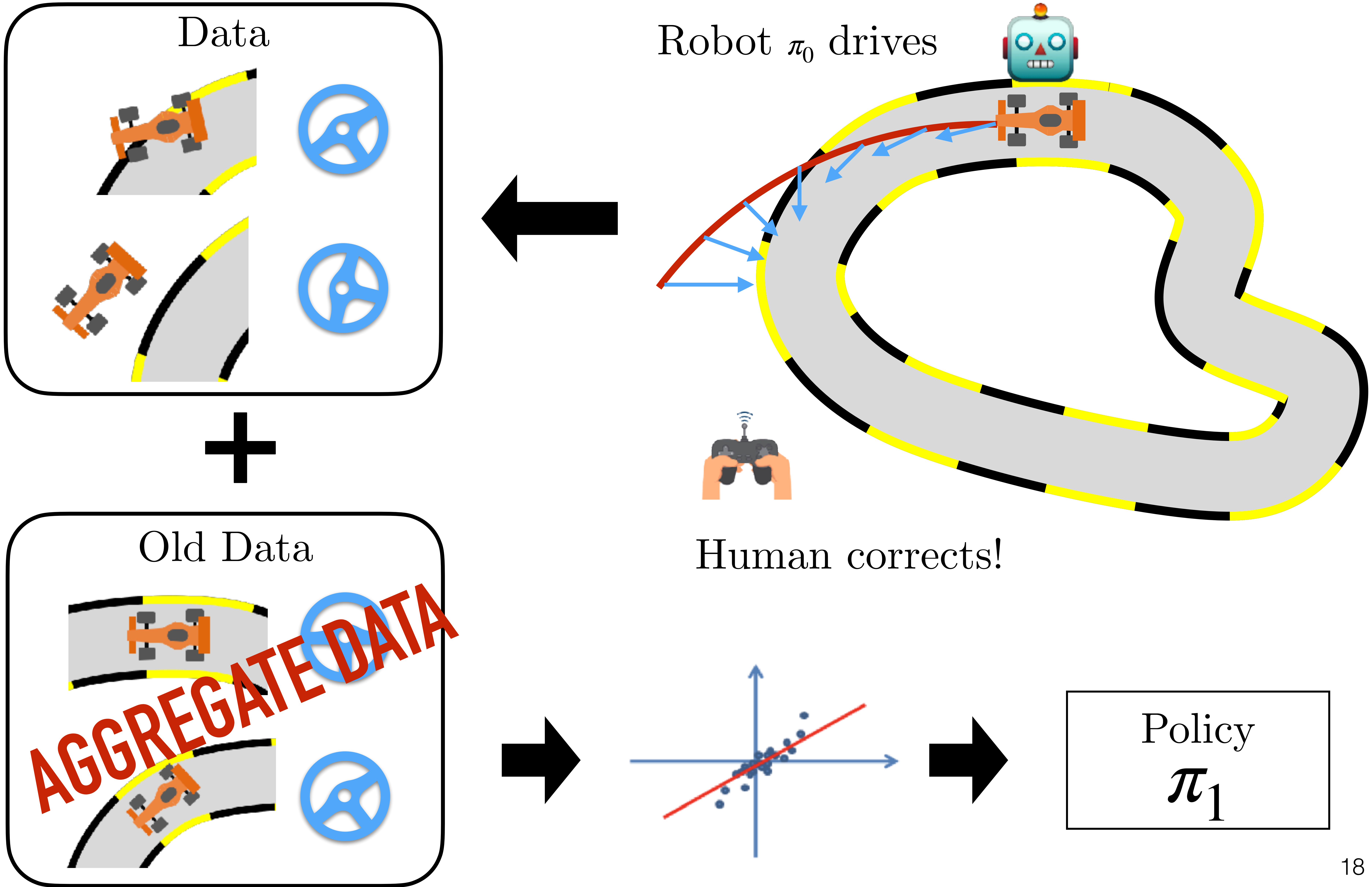


Human drives



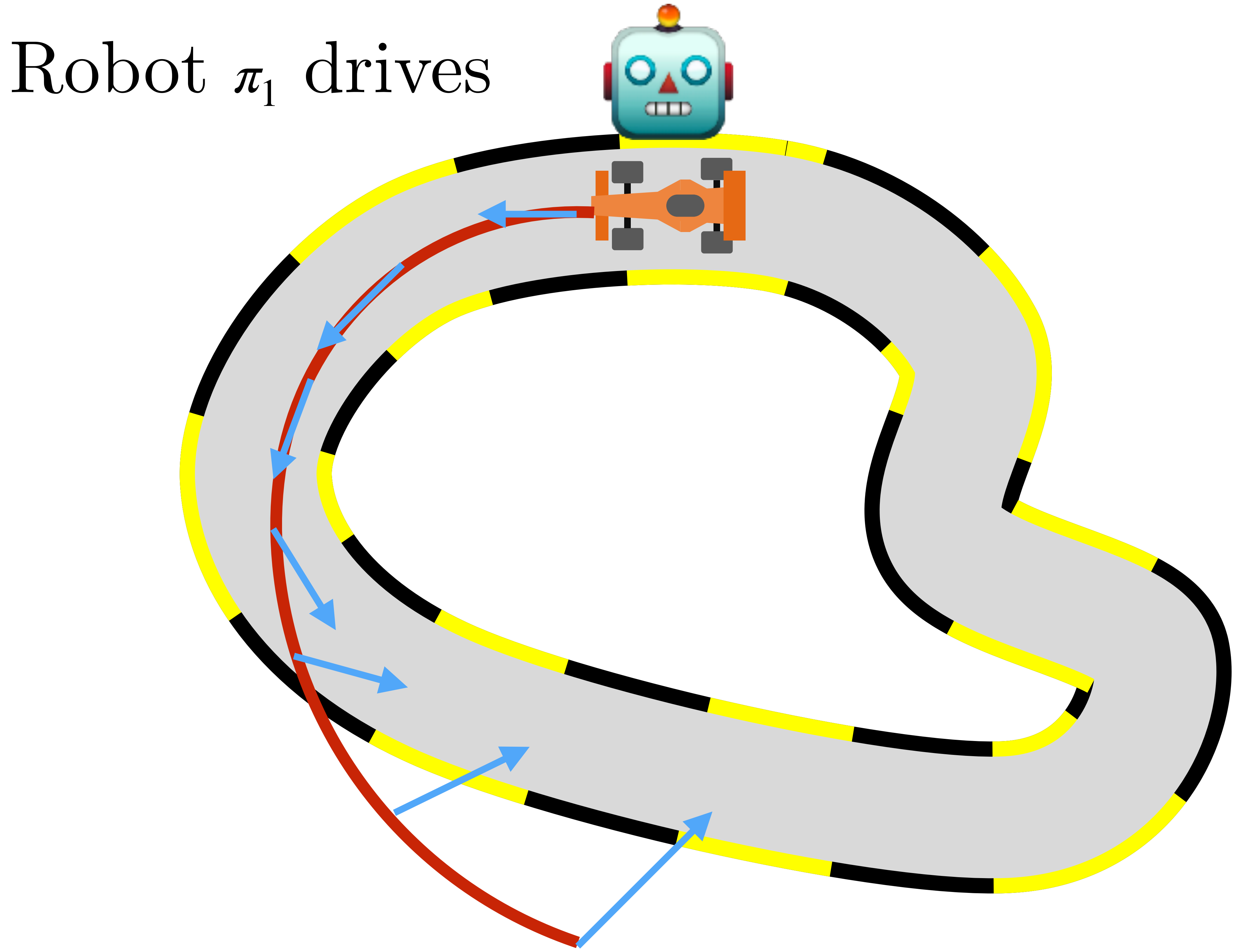
Policy
$\pi_0$

# DAgger: Iteration 1



[Ross et al'11]

# DAgger: Iteration 2

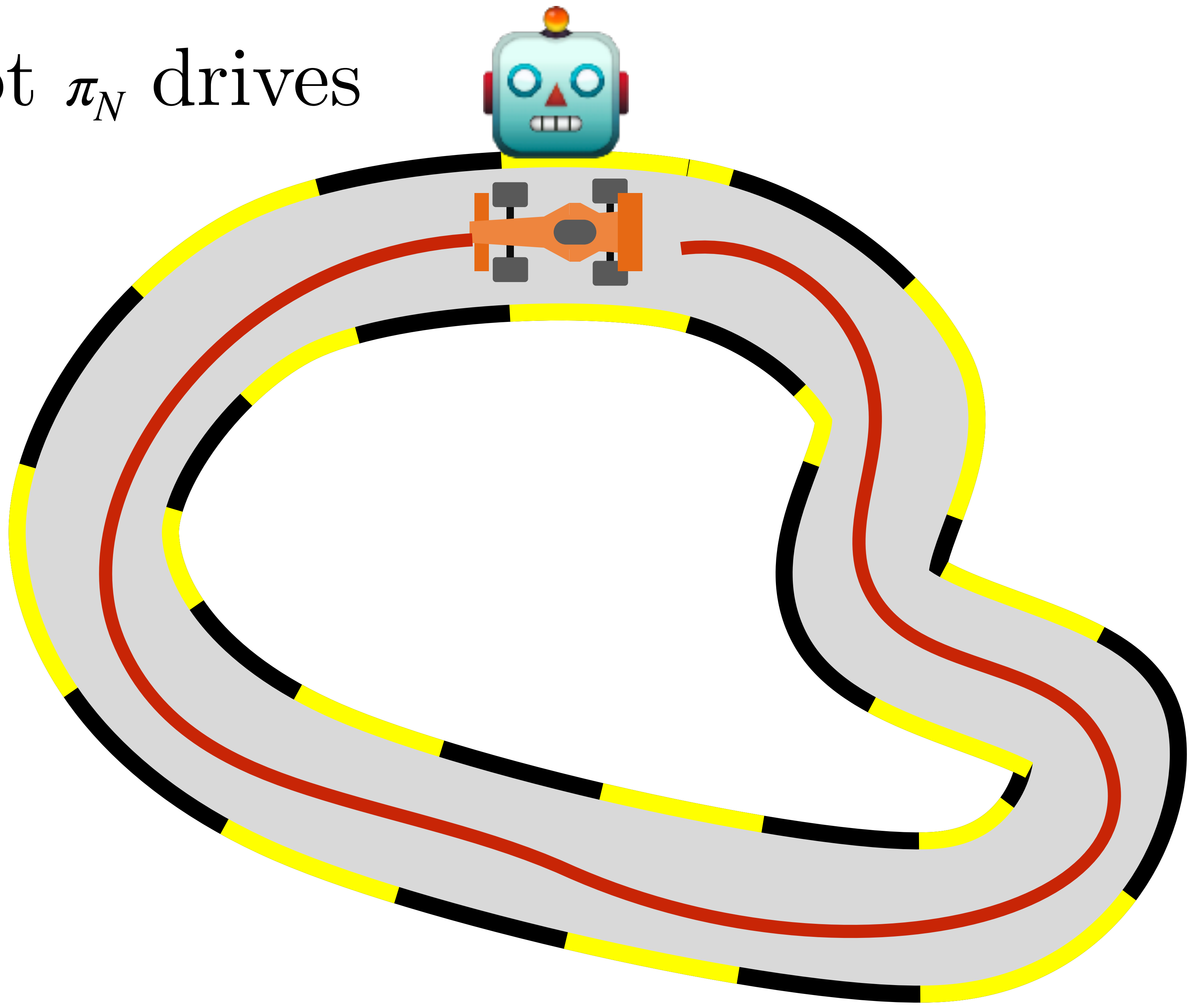



AGGREGATE DATA

[Ross et al'11]

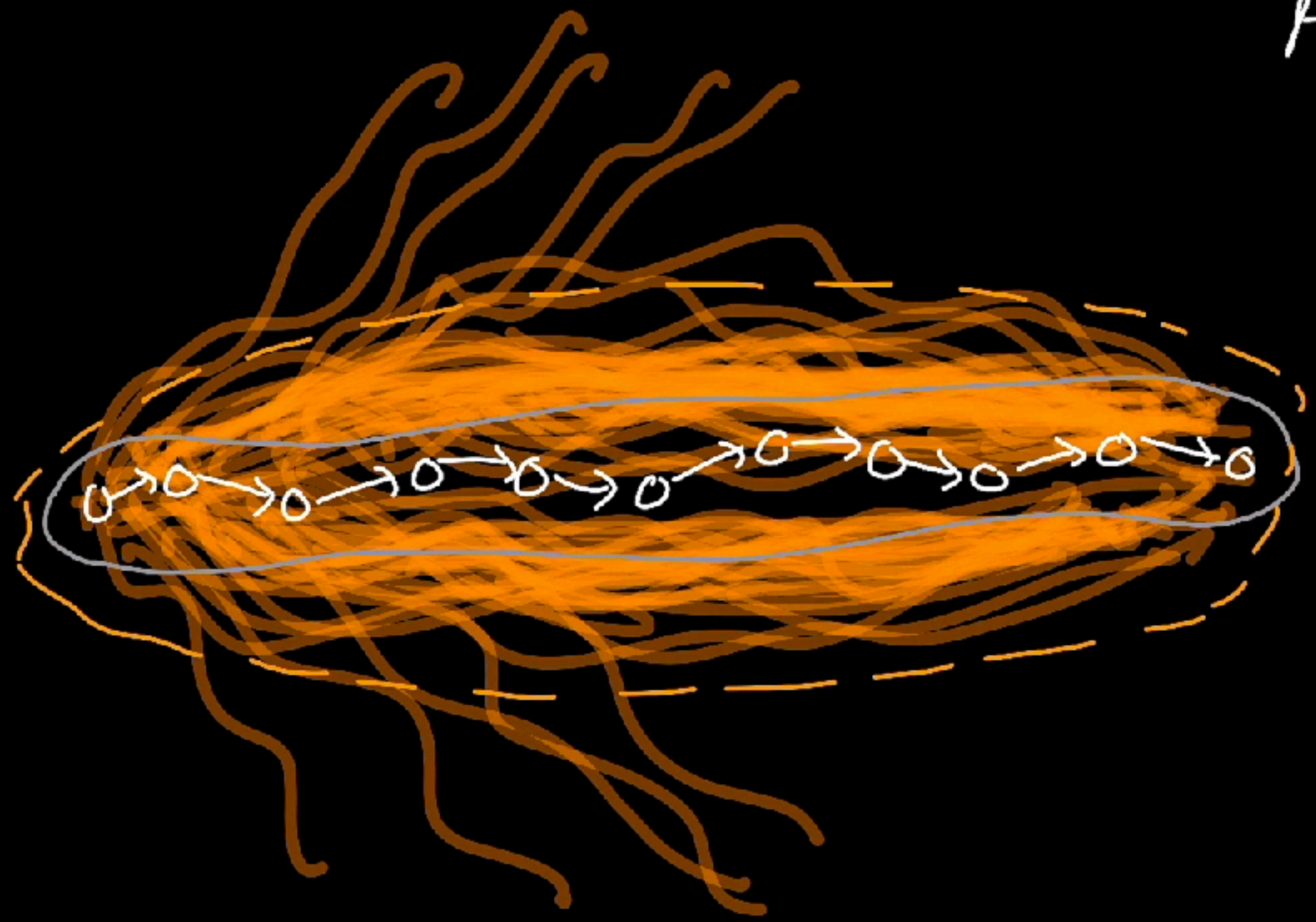
# DAgger: Iteration 1

Robot $\pi_N$ drives



After many iterations ....
we are able to drive like a human!

# Dagger (Dataset Aggregation)



Aggregated Training Distributio

$\approx$

Test Distribution

Human Distribution

But why does *aggregating* data work?

# The Imitation Game

| Learner | | Adversary |
| --- | --- | --- |
| Initialize policy | $\pi_1$ [policy] → | |
| | ← $l_1(\,.\,)$ [loss] | Chooses loss |
| Update policy | $\pi_2$ → | |
| | ← $l_2(\,.\,)$ | Chooses loss |
| ⋮ | | ⋮ |

Imitation learning is just a game

*Be stable*

Slowly change predictions

Let's prove!

How can I customize DAGGER to be more practical?

# Customizing your DAGGER

Q1. The policy iteration at step 1 is crappy and visits irrelevant states. What do I do?

Blend the expert and learner policy $\pi_i = (1 - \beta_i)\hat{\pi}_i + \beta_i \pi^*$

Q2. What if I can't afford to store all the aggregated data?

Online gradient descent!


Original results from DAGGER!

# DAGGER is a foundation

*Imitation under uncertainty*

SAIL

ExPLORE STROLL

Counterfactual Teaching

DPI LOLS

NRPI

*Reinforcement Learning*

DAEQUIL

AGGREVATE(D)

EIL

HG-DAGGER

SHIV

Agnostic SysID

DaaD

*Model learning*

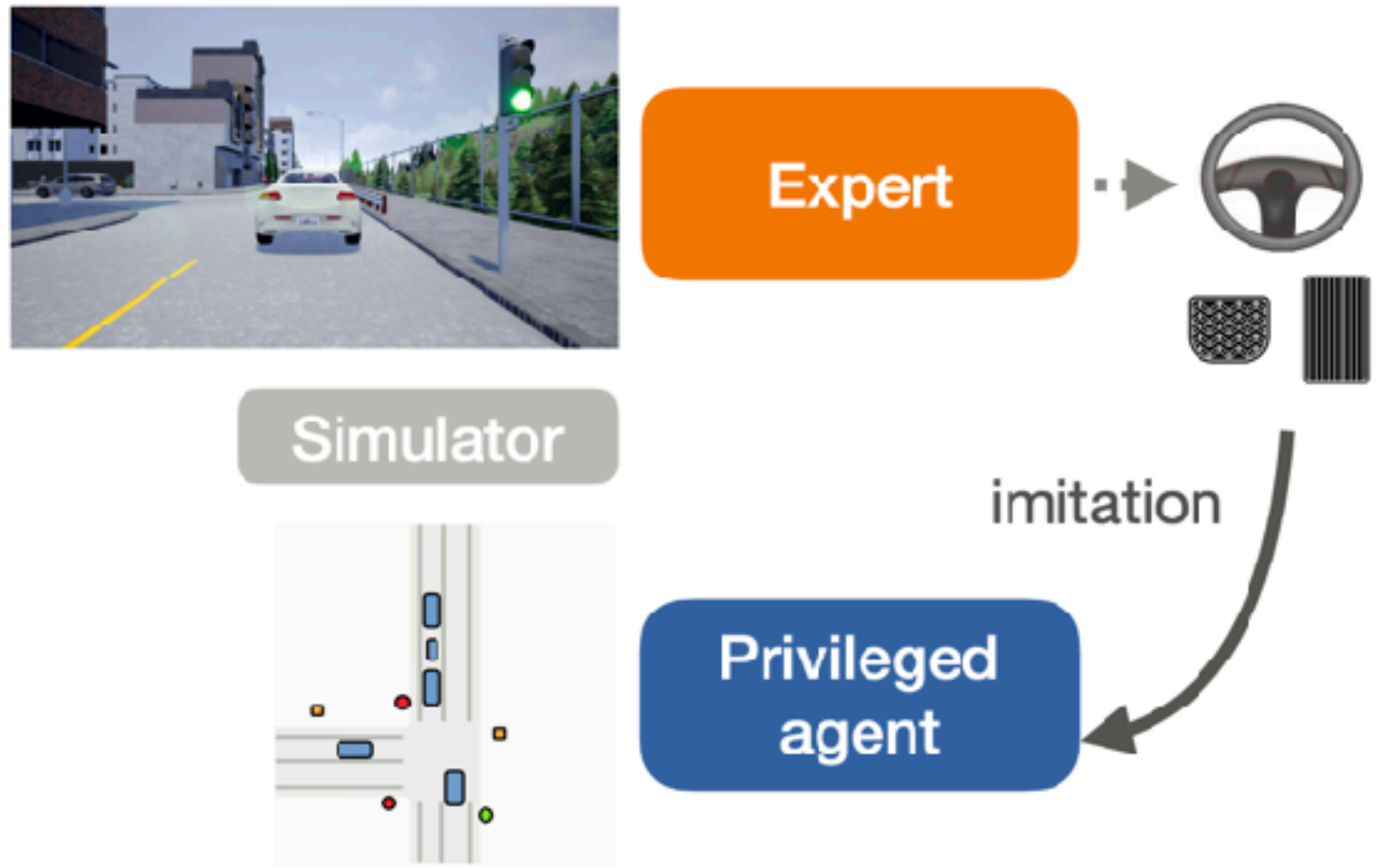
*Imitation learning*

*Query efficient imitation learning*

DAGGER

# Many cool applications of DAGGER in robotics

Lee et al, Learning quadrupedal locomotion over challenging terrain (2020)

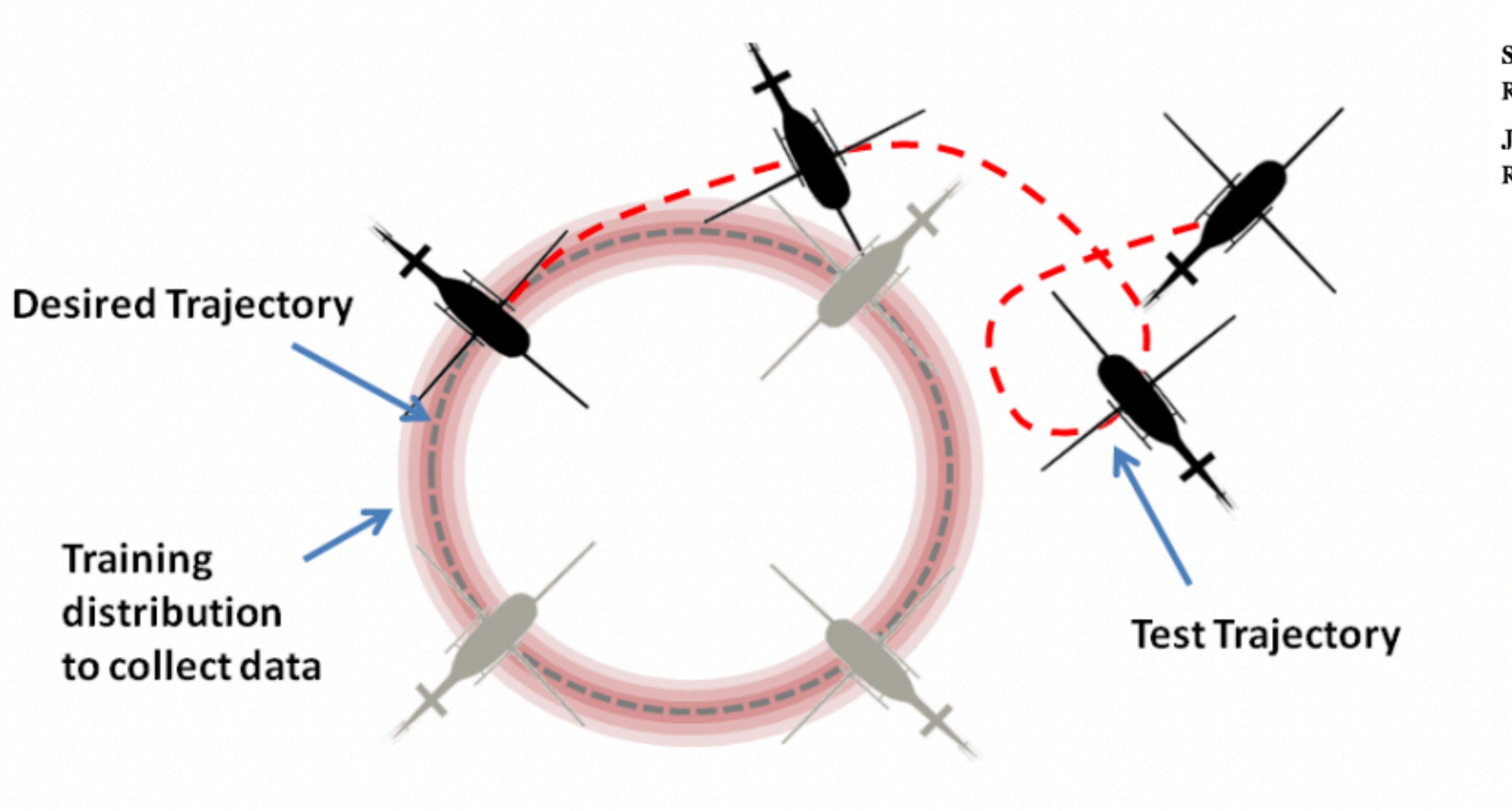


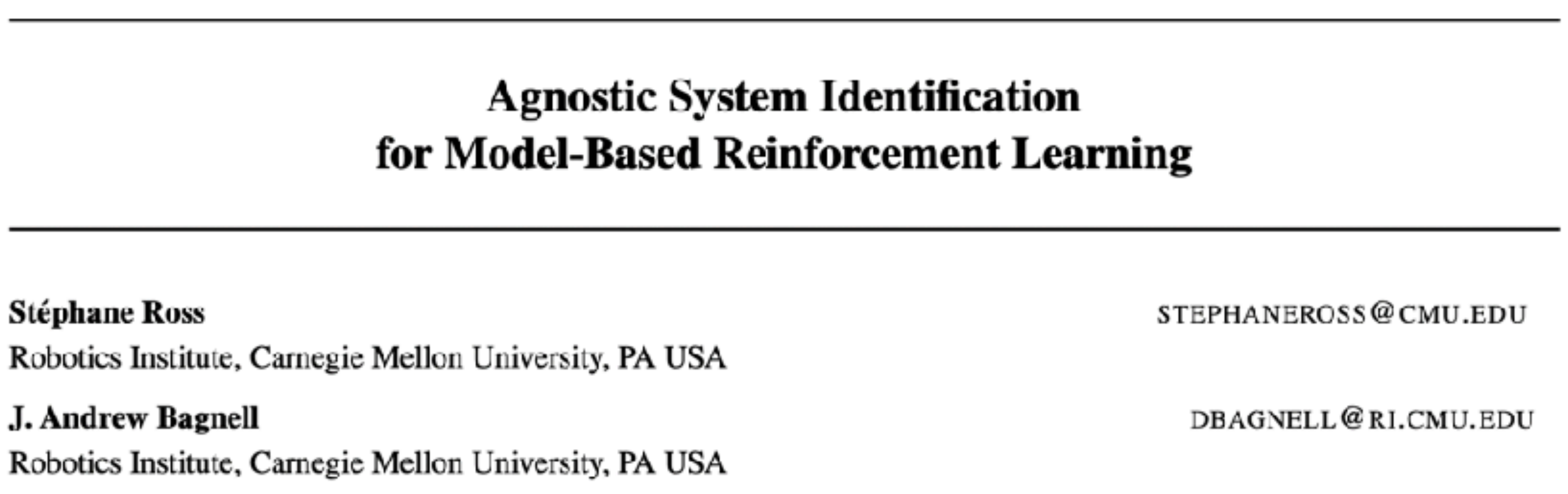

Chen et al Learning by Cheating(2020)

Choudhury et al, Data Driven Planning via Imitation Learning (2018)

Pan et al Imitation learning for agile autonomous driving (2019)

# DAGGER is not *just* for imitation learning!



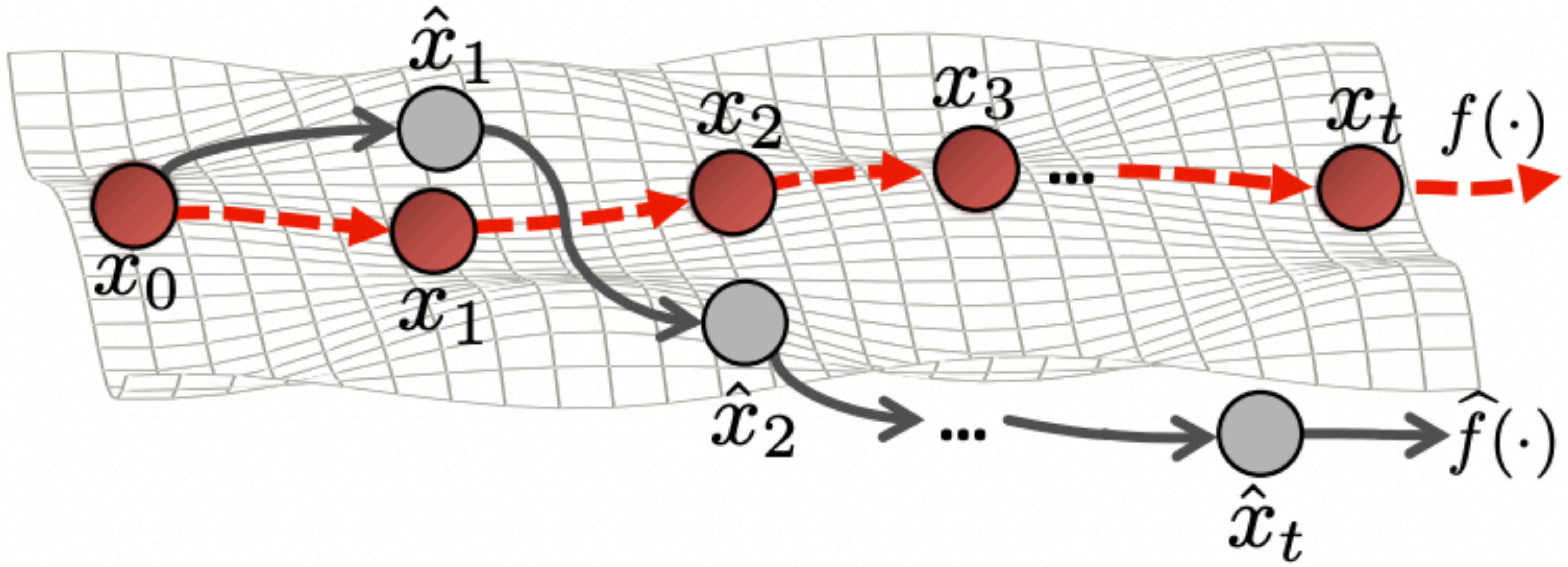



**Agnostic System Identification for Model-Based Reinforcement Learning**

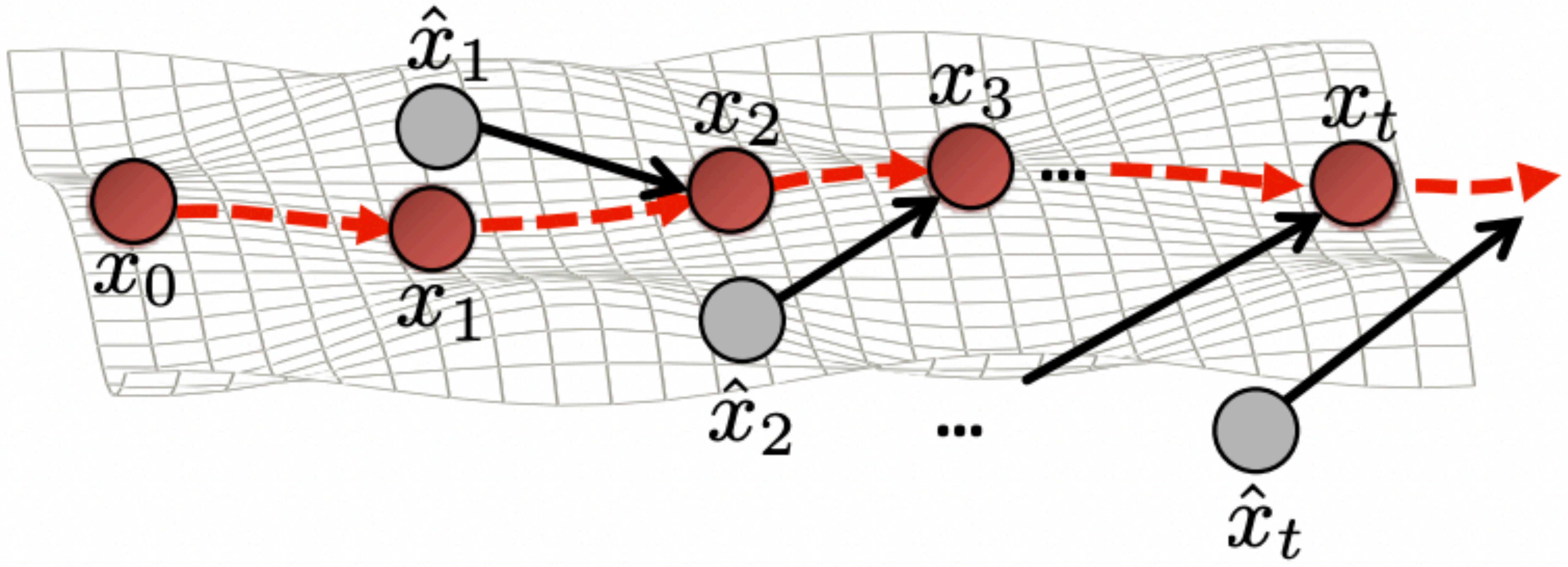

**Stéphane Ross** STEPHANEROSS@CMU.EDU
Robotics Institute, Carnegie Mellon University, PA USA
**J. Andrew Bagnell** DBAGNELL@RI.CMU.EDU
Robotics Institute, Carnegie Mellon University, PA USA


Model-based Reinforcement Learning

# DAGGER is not *just* for imitation learning!




(a) Forward simulation of learned model (gray) introduces error at each prediction step compared to the true time-series (red)




(b) Data provides a demonstration of corrections required to return back to proper prediction

## DATA AS DEMONSTRATOR with Applications to System Identification


**Arun Venkatraman**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
aruvenk@cs.cmu.edu

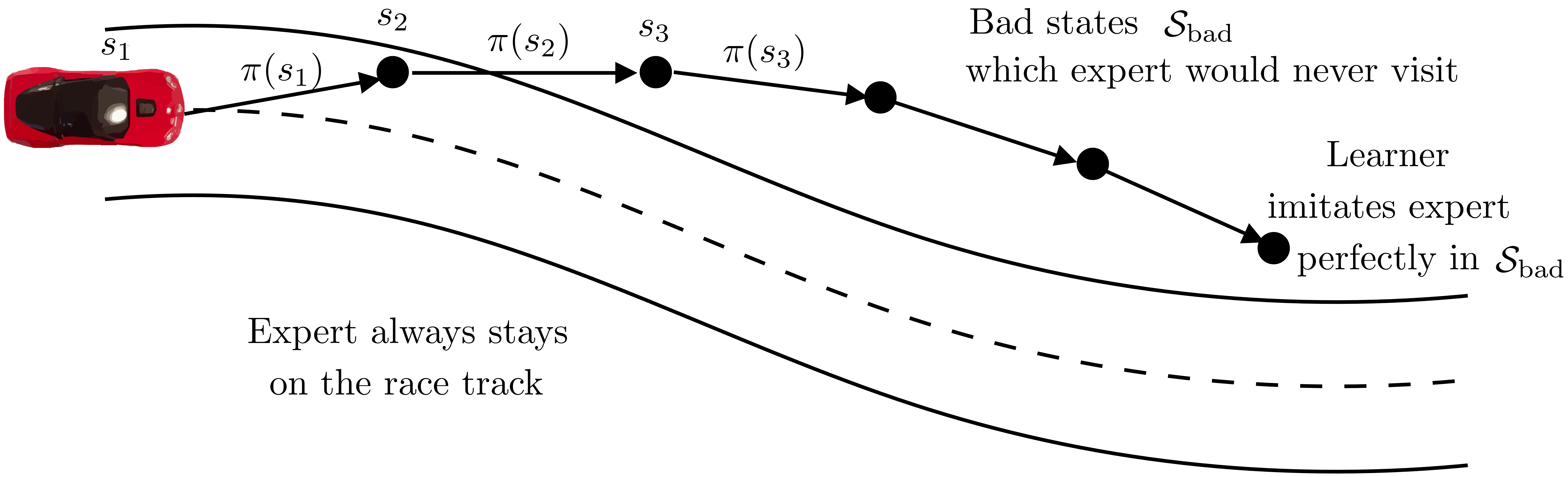
**Byron Boots**
School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332
bboots@cc.gatech.edu

**Martial Hebert**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
hebert@ri.cmu.edu

**J. Andrew Bagnell**
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
dbagnell@ri.cmu.edu

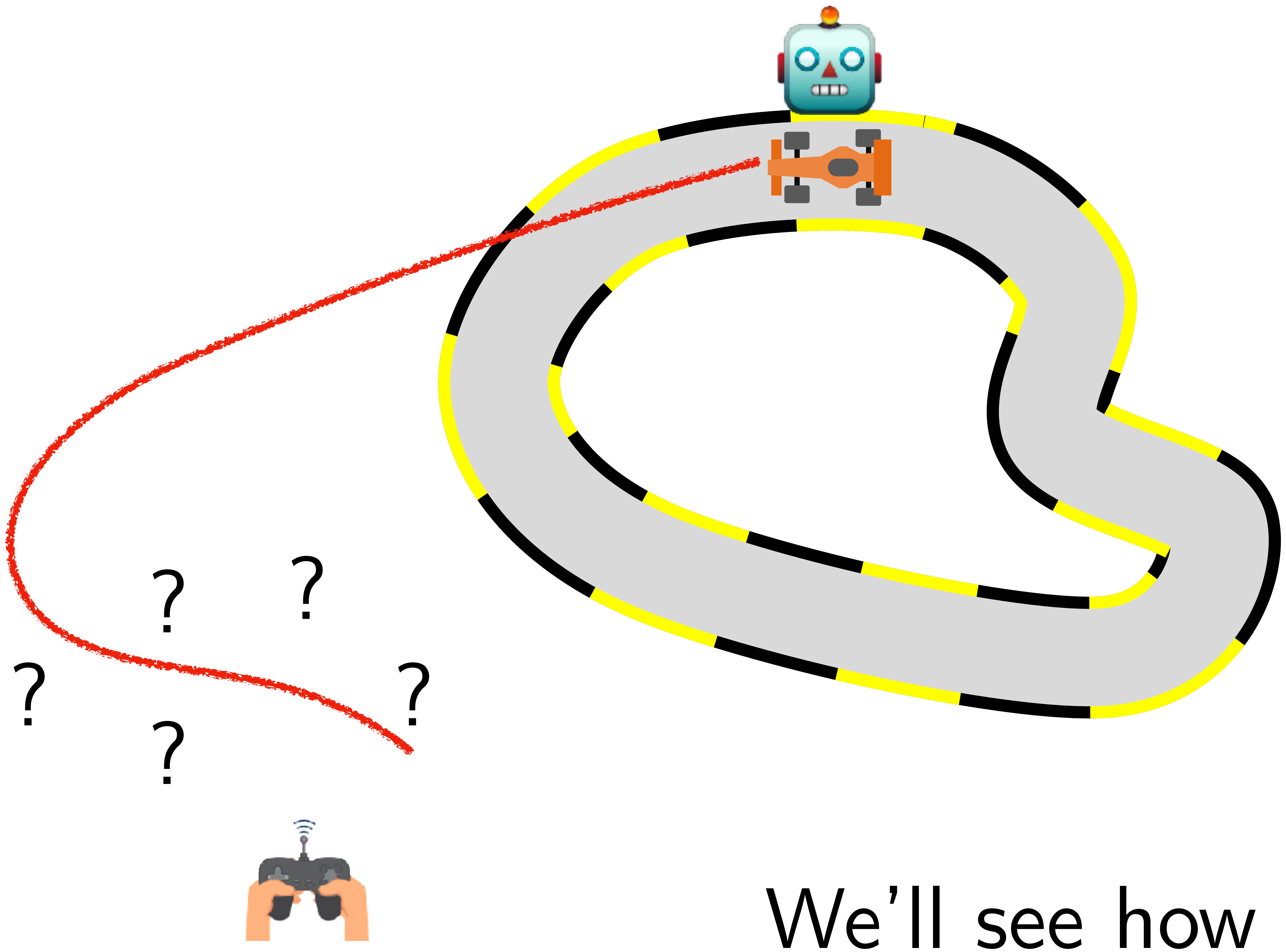
# Hidden charges from DAGGER

# Hidden charge #1: Not all mistakes are equal



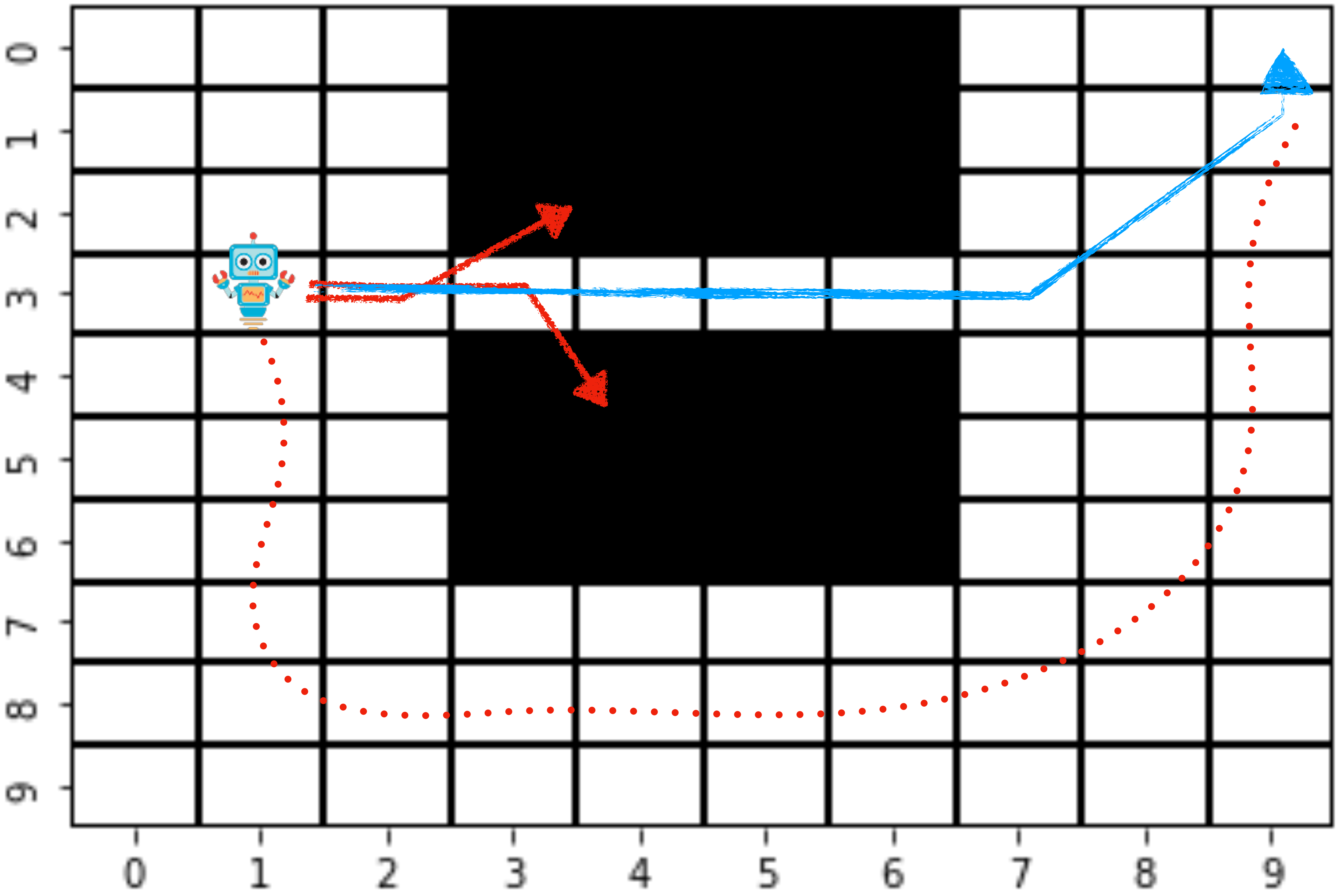

Dagger minimizes 0-1 loss, but what we really want to optimize are advantages! (More next lecture)

# Hidden charge #2: Dagger asks the expert for queries *everywhere*




We'll see how to learn from
limited human feedback (interventions)

# Hidden charge #3: Dagger expects at least one policy to be good *everywhere*

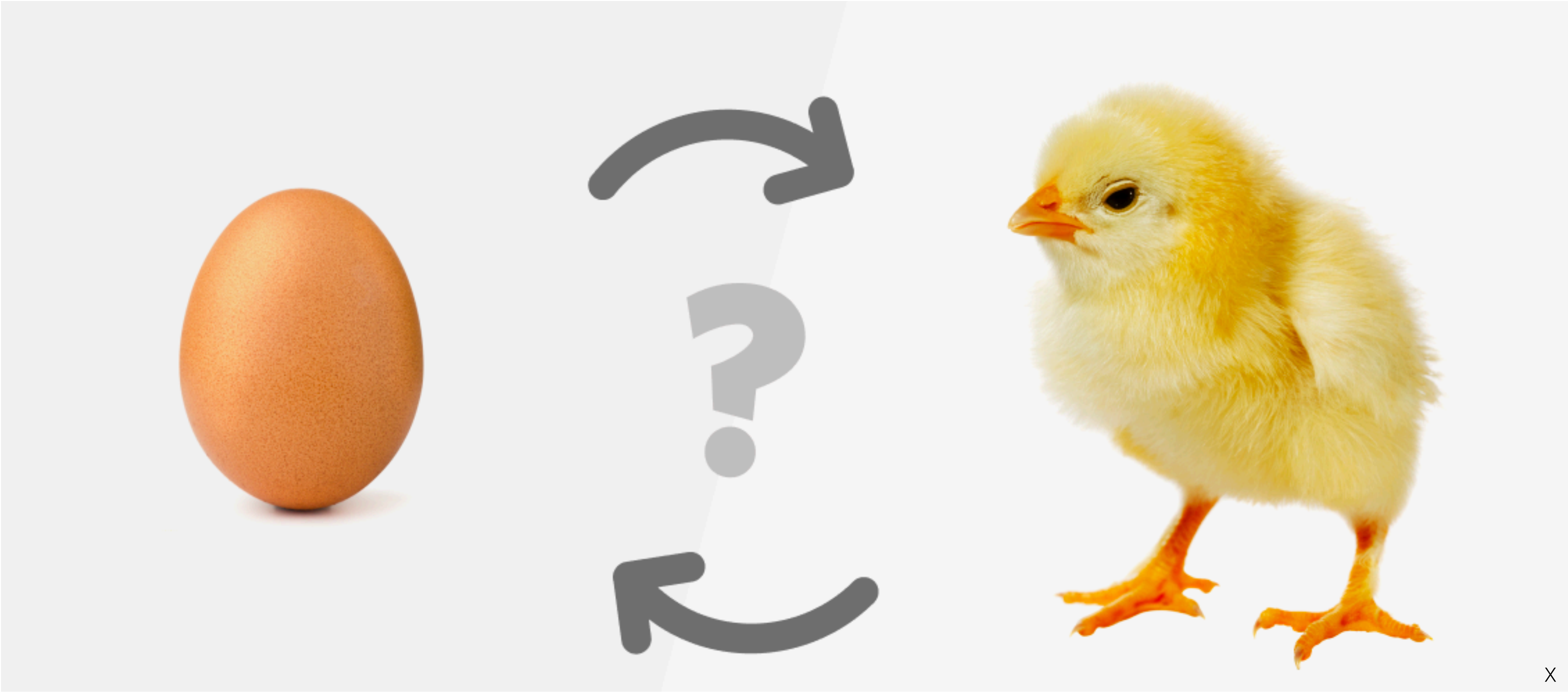


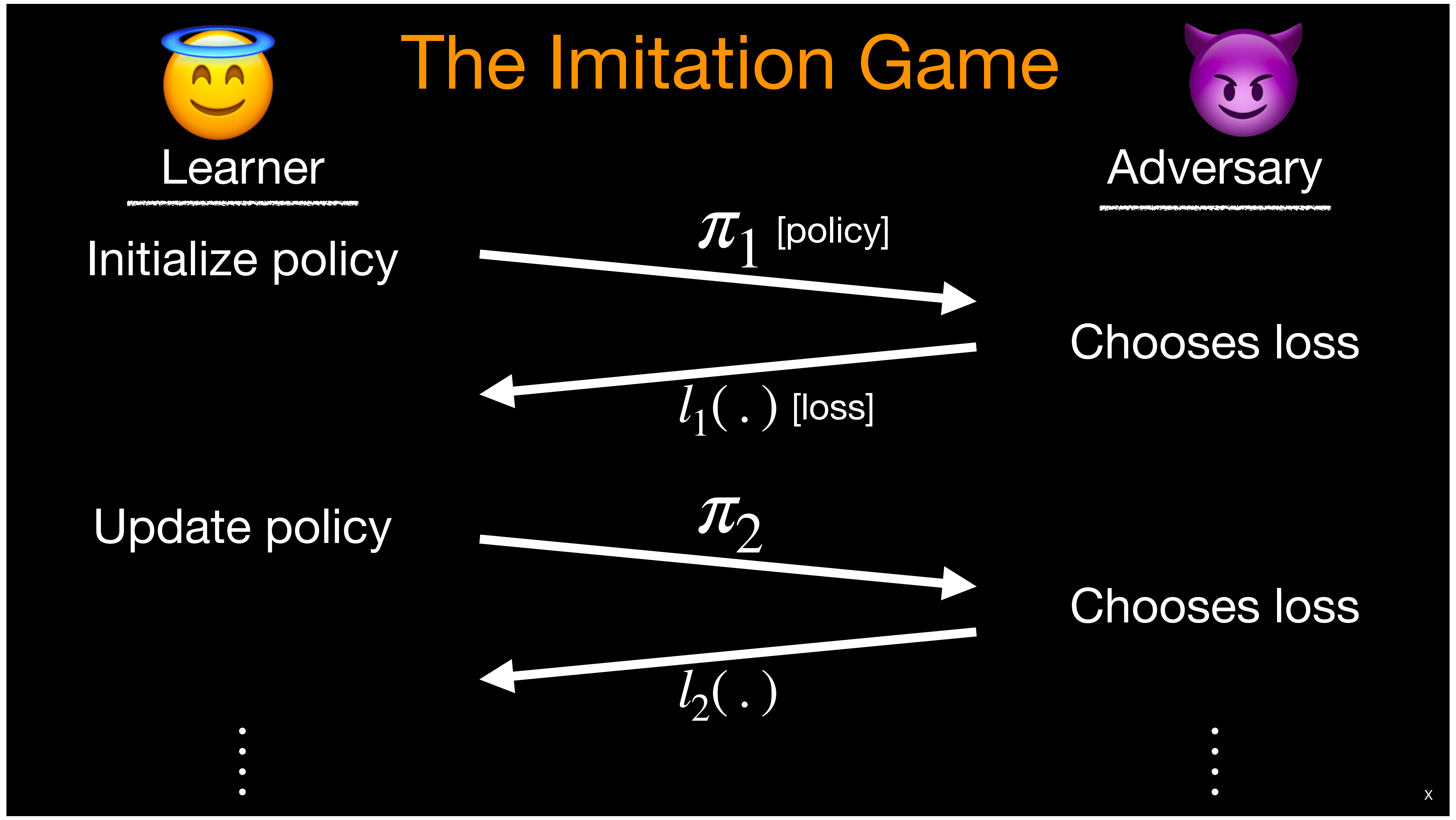

Learner simply can't cross the bridge! …

… but can take the long way round.

# tl;dr

To know the distribution, you need a learner
To train a learner, you need a distribution






[Ross et al'11]