Imitation Learning
&
Dagger

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning

Fall 2022

Today

The Bitter Lesson ('19)

- Recap++
  Examples + Videos

- Today:
  1. Imitation Learning ~~with~~
  2. DAgger

Recap++

Some Helpful Notation: Visitation Measures

- Visitation probability at time h: $\mathbb{P}_h(s_h, a_h \mid \mu, \pi)$
- Average Visitation Measure:

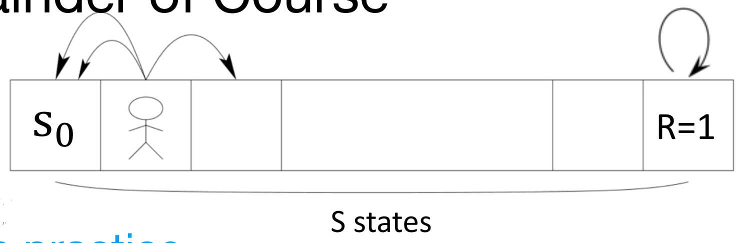
$$d_\mu^\pi(s, a) = \frac{1}{H} \sum_{h=0}^{H-1} \mathbb{P}_h(s, a \mid \mu, \pi)$$

$d^\pi_\mu(s)$

“Lack of Exploration” leads to Optimization and Statistical Challenges

- Suppose $|S| \approx H$ or $|S| \approx 1/(1-\gamma)$ & $\mu(s_0) = 1$ (i.e. we start at s_0).
- A randomly initialized policy has prob. $O(1/3^{|S|})$ of hitting the goal state in a single trajectory.
- Implications:
  - Any sample based policy iteration approach (starting with this policy) requires $O(3^{|S|})$ trajectories to make progress at the very first step.
  - Same for any sample based PG method.
  - Related: even if we had exact gradients, the “landscape” is such that these gradients are exponentially small, at randomly initialized policy (see AJKS Ch 11).

Implications/Comments/Remainder of Course



Thrun '92

- Sometimes exploration is (or can be made) "easier" in practice
  - Random strategies can reach "rewarding milestones"
  - We can design/"shape" the reward function to help us out.
- We can try to make the distribution μ to have better coverage.
  - For small problems, μ being uniform would make all these issues go away. (for large problems, μ being uniform may not help at all. Why?)
  - Ideally, μ having support on where a good policy tends to visit is helpful (sometimes we can't design μ)
- Course:
  - A little theory with regards to μ and PG. (today) PG has better guarantees than approx DP methods (in terms of μ).
  - Imitation learning (starting today). An expert gives us samples from a "good" μ.
  - Explicit Exploration: for the "tabular case" (we will mix UCB with VI!)

Fitted Policy Improvement Guarantees (optional)

- Let $s_0, a_0 \sim \mu$ now be the starting "state-action" distribution. $J(\pi) = E_{s_0,a_0\sim\mu}[Q^\pi(s,a)]$ (the theory is better suited to this. See AJKS).
- Approximation error: For all policies, suppose that for all π,

$$\min_\theta E_{s,a\sim\mu}\left[\left(Q^\pi(s,a) - \theta^\top\phi(s,a)\right)^2\right] \le \delta, \text{ and } \quad \min_\theta \|Q^\pi - \theta^\top\phi\|_\infty \le \delta_\infty$$

  - δ: the average case supervised learning error (reasonable to expect this can be made small)
    δ_∞: the worse case error (often unreasonable to expect to be small)

[Theorem:] (informal, see AJKS Ch 4+13)

- Suppose that we use a # samples that is poly in d & $1/\epsilon_{stat}$ for both fittedPI and NPG.
- FittedPI will return a policy π^{FPI} with the performance guarantee:

$$J(\pi^{FPI}) \ge J(\pi^\star) - \epsilon_{stat} - 2H^2\delta_\infty$$

- NPG has the same guarantee.
- NPG also has a stronger guarantee: Suppose μ has "reasonable support" on where $\pi^\star$ tends to visit, i.e. suppose:

$$\max_{s,a}\left(\frac{d_\mu^{\pi^\star}(s,a)}{\mu(s,a)}\right) \le C$$

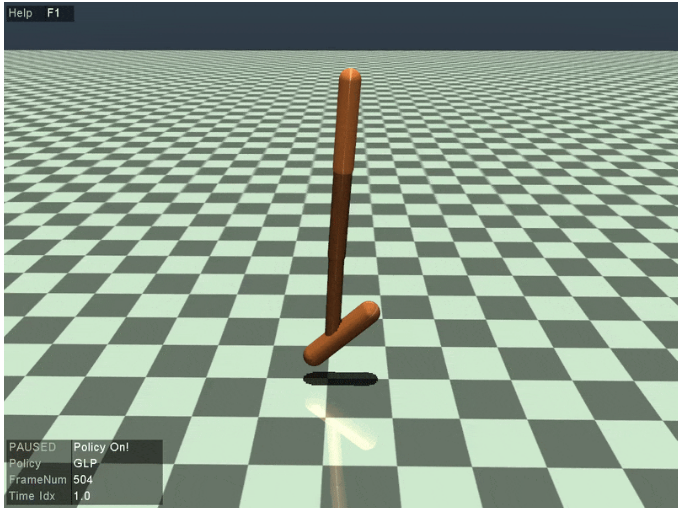
  then NPG will return a policy with sub-optimality determined by C and the average case error δ:

$$J(\pi^{NPG}) \ge J(\pi^\star) - \epsilon_{stat} - 2H^2C\delta$$

Aside: Brittle policies if we train starting from only from one configuration!

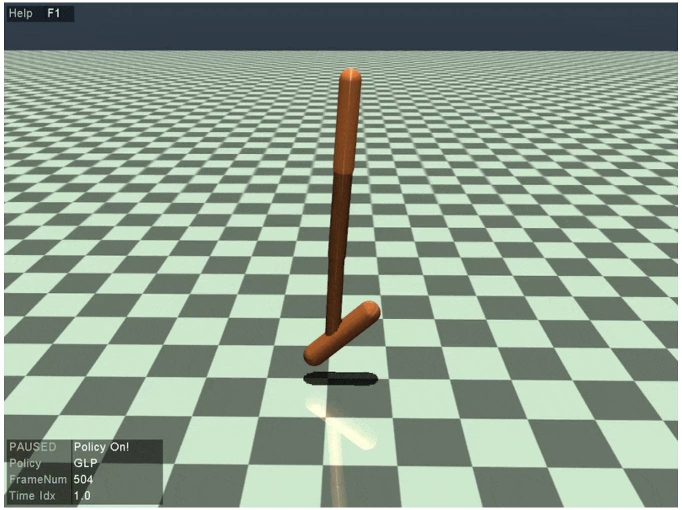
- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!

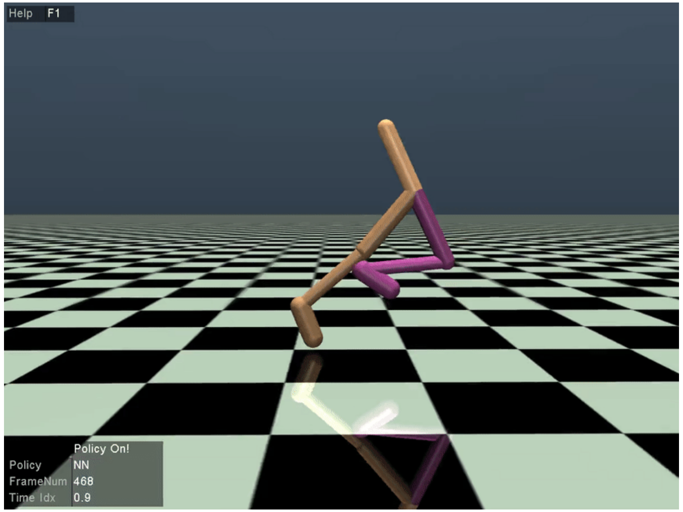
Aside: Brittle policies if we train starting from only from one configuration!



- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!

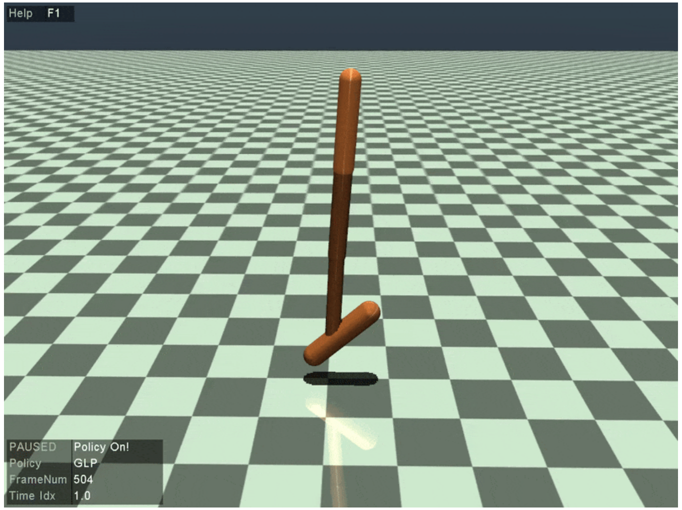
Aside: Brittle policies if we train starting from only from one configuration!

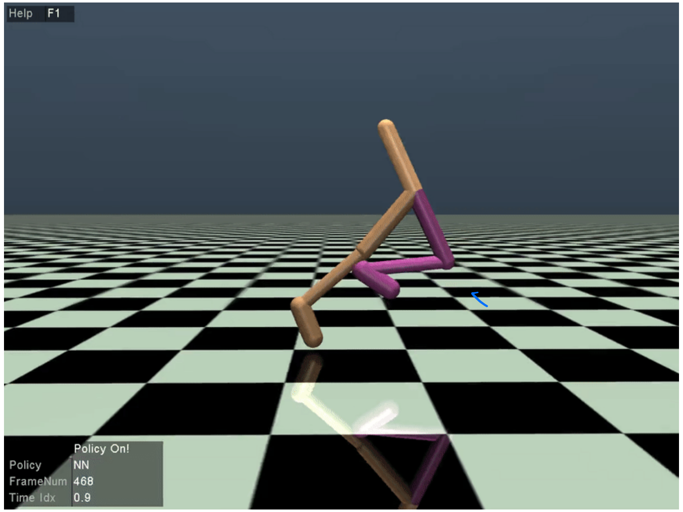




- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!

Aside: Brittle policies if we train starting from only from one configuration!





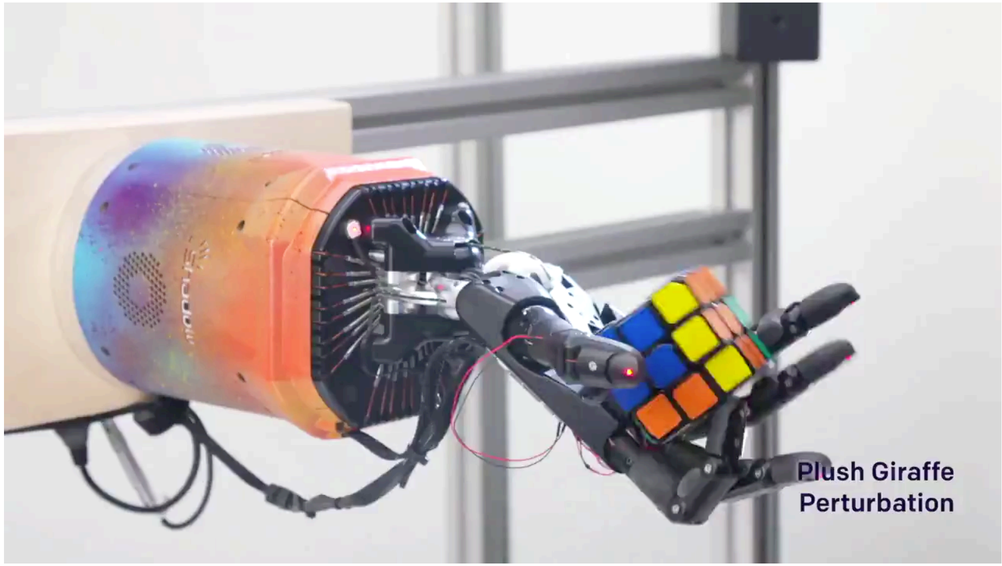
- [Rajeswaran, Lowrey, Todorov, K. 2017]: showed policies optimized for a single starting configuration s_0 are not robust!
- How to fix this?
  - Training from different starting configurations sampled from $s_0 \sim \mu$ fixes this.

$$\max_{\theta} E_{s_0 \sim \mu}[V^{\theta}(s_0)]$$

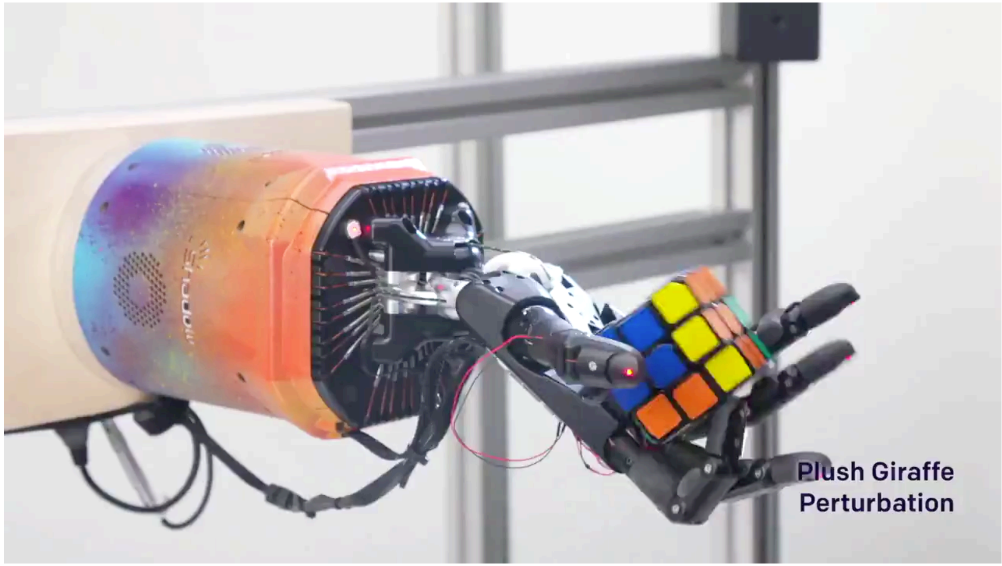
  - The measure μ is also relevant for robustness.

OpenAI: progress on dexterous hand manipulation

OpenAI: progress on dexterous hand manipulation



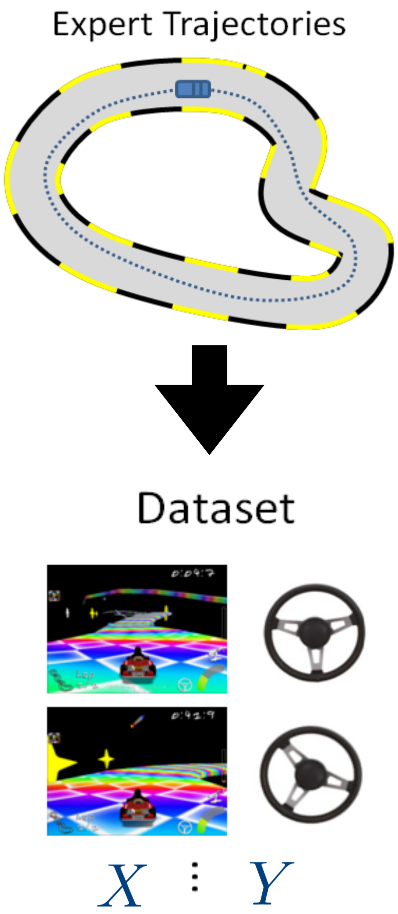
OpenAI: progress on dexterous hand manipulation



Trained with "domain randomization"

Basically, the measure $s_0 \sim \mu$ was diverse.

IL Setting and the Behavior Cloning algorithm



Discounted infinite horizon MDP $\mathscr{M} = \{S, A, \gamma, r, P, \rho, \pi^\star\}$

Ground truth reward $r(s, a) \in [0,1]$ is unknown;

For simplicity, let's assume expert is a (nearly) optimal policy $\pi^\star$

We have a dataset $\mathscr{D} = (s_i^\star, a_i^\star)_{i=1}^M \sim d^{\pi^\star}$

Goal: learn a policy from $\mathscr{D}$ that is as good as the expert $\pi^\star$

Let's formalize the Behavior Cloning algorithm

BC Algorithm input: a restricted policy class $\Pi = \{\pi : S \mapsto \Delta(A)\}$

BC is a Reduction to Supervised Learning:

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \ell\left(\pi, s^\star, a^\star\right)$$

Many choices of loss functions:

1. Negative log-likelihood (NLL): $\ell(\pi, s, a^\star) = -\ln \pi(a^\star \mid s^\star)$

2. square loss (i.e., regression for continuous action): $\ell(\pi, s, a^\star) = \|\pi(s) - a^\star\|_2^2$

Performance Guarantee

Assumption: we are going to assume Supervised Learning succeeded

$$\mathbb{E}_{s \sim d_\mu^{\pi^\star}} \mathbf{1}\left[\widehat{\pi}(s) \neq \pi^\star(s)\right] \leq \epsilon \in \mathbb{R}^+$$

Theorem [BC Performance] With probability at least $1-\delta$, BC returns a policy $\widehat{\pi}$:

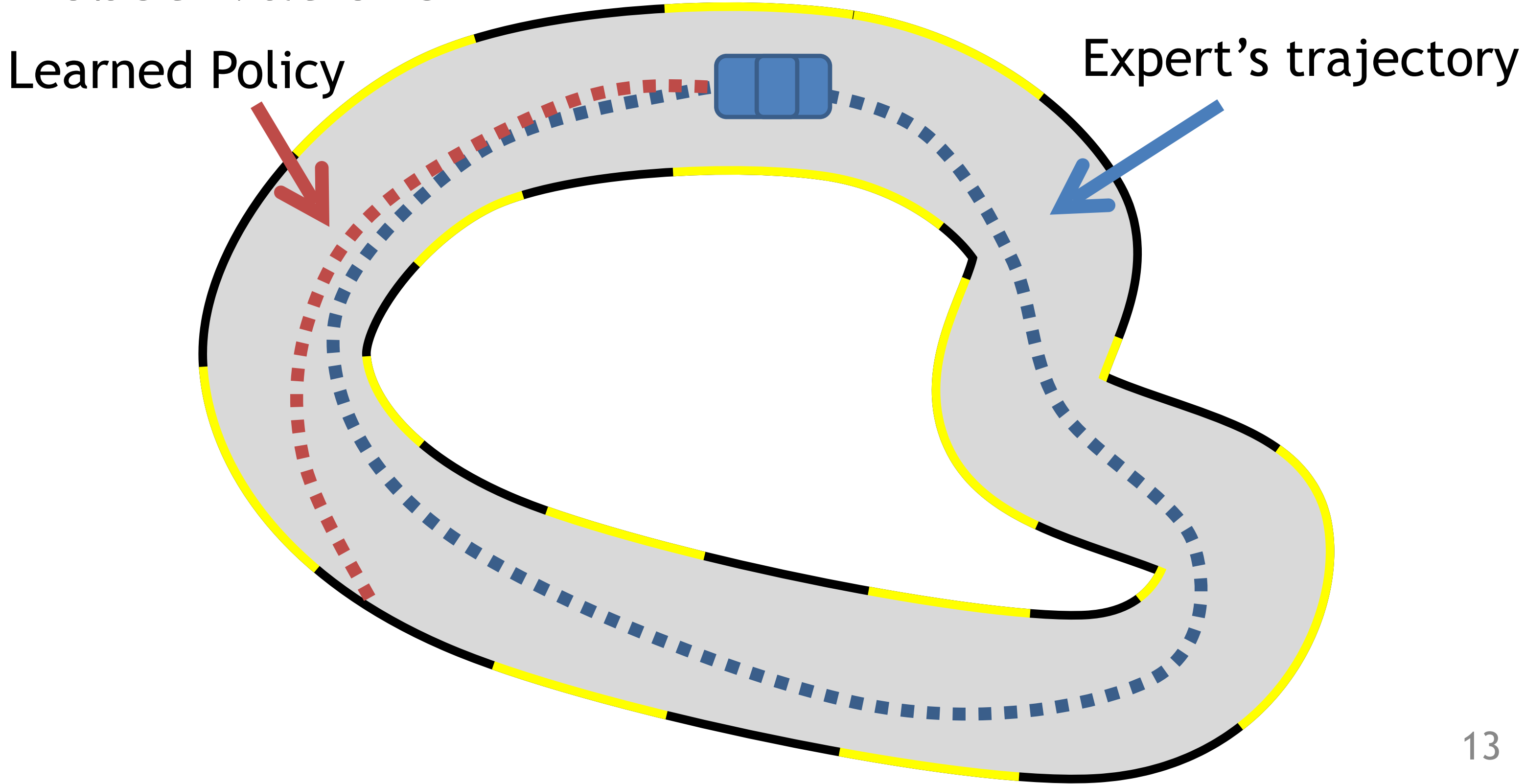
$$V^{\pi^\star} - V^{\widehat{\pi}} \leq \frac{2}{(1-\gamma)^2}\epsilon \qquad 2H^2 \epsilon$$

$$V_\mu^{\hat{\pi}} \geq V_\mu^{\star} - \frac{2}{(1-\gamma)^2}\epsilon$$

The quadratic amplification is annoying

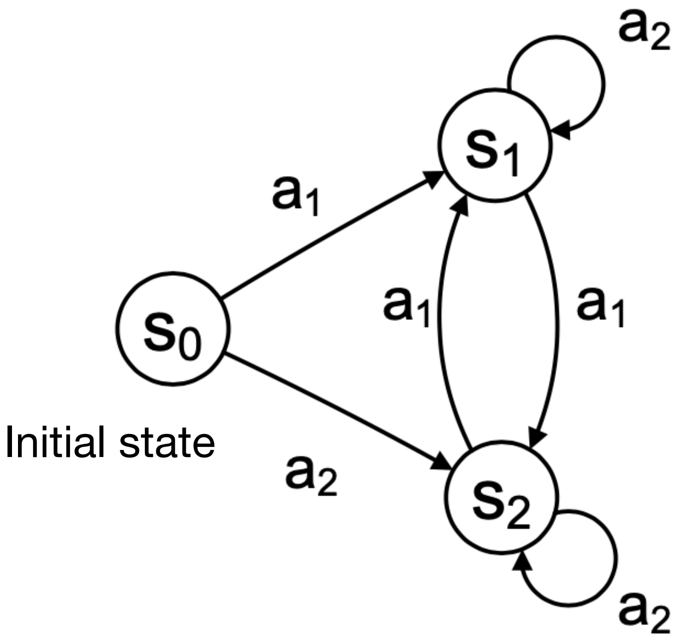
What could go wrong?

- Predictions affect future inputs/ observations



Distribution Shift: Example (finite horizon case)

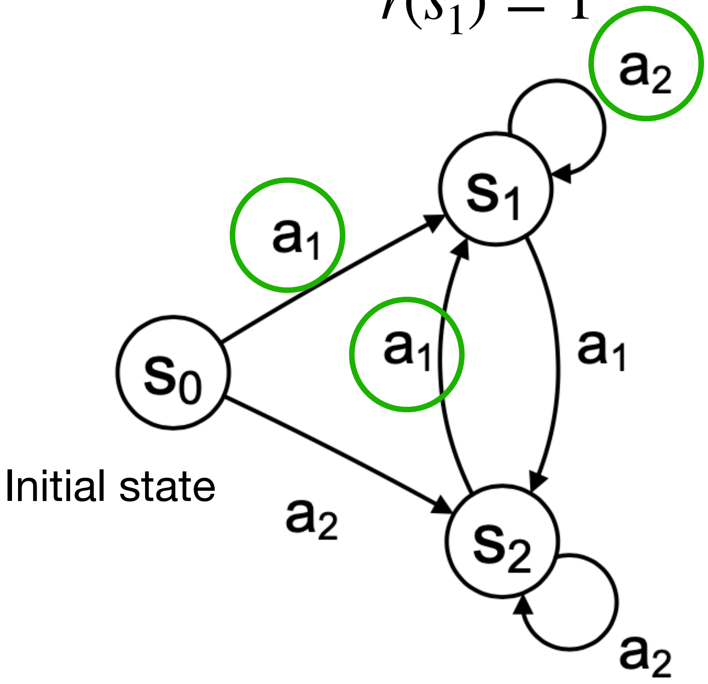
$r(s_1) = 1$



$d_{s_0}^{\pi^*}(s_0)$

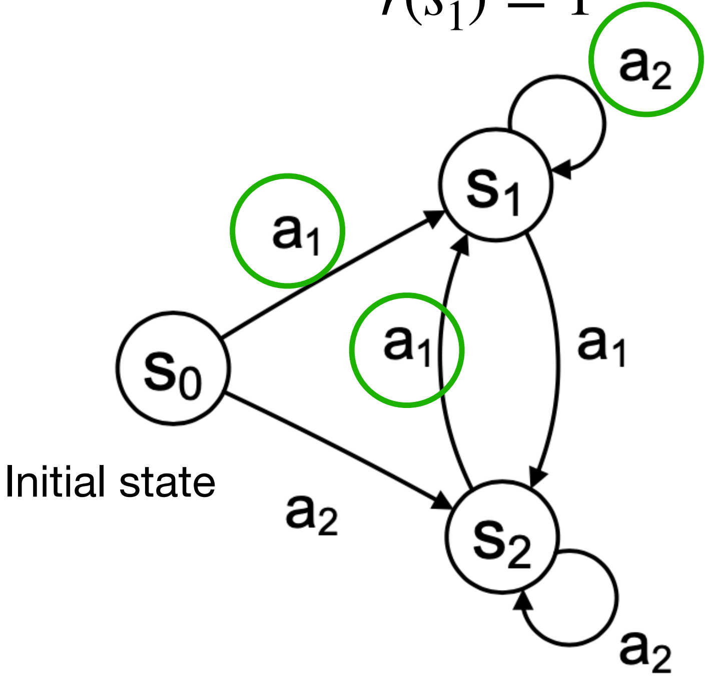
Distribution Shift: Example (finite horizon case)

$r(s_1) = 1$



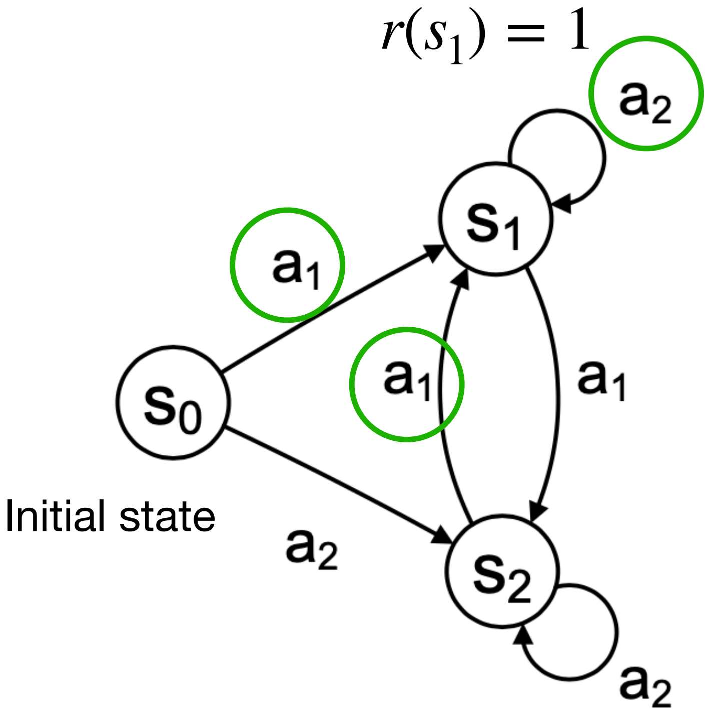
Distribution Shift: Example (finite horizon case)

$r(s_1) = 1$



$$d_{s_0}^{\pi^\star}(s_0) = \frac{1}{H}, \; d_{s_0}^{\pi^\star}(s_1) = \frac{H-1}{H}, \; d_{s_0}^{\pi^\star}(s_2) = 0$$

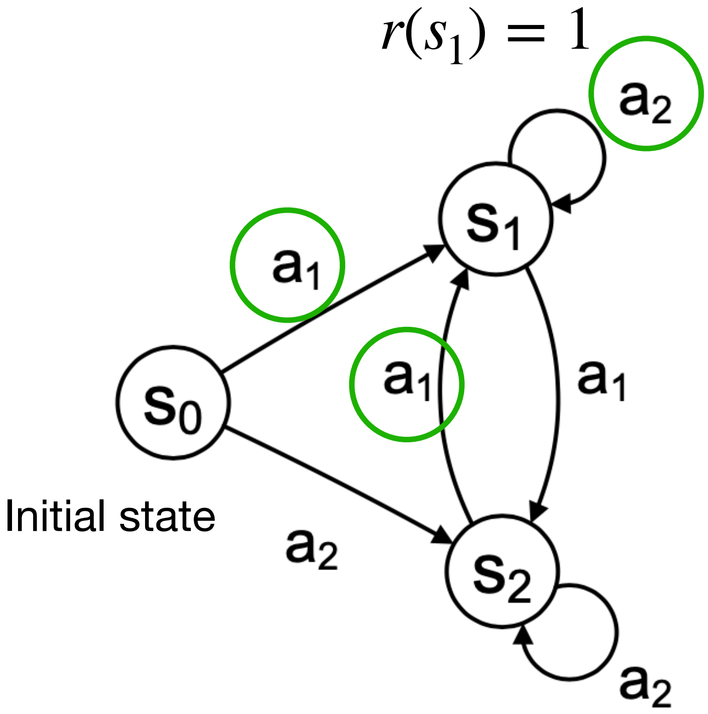
Distribution Shift: Example (finite horizon case)



$$d_{s_0}^{\pi^\star}(s_0) = \frac{1}{H}, \; d_{s_0}^{\pi^\star}(s_1) = \frac{H-1}{H}, \; d_{s_0}^{\pi^\star}(s_2) = 0$$

$$V_{s_0}^{\pi^\star} = H - 1$$

Distribution Shift: Example (finite horizon case)



$$d_{s_0}^{\pi^\star}(s_0) = \frac{1}{H},\ d_{s_0}^{\pi^\star}(s_1) = \frac{H-1}{H},\ d_{s_0}^{\pi^\star}(s_2) = 0$$

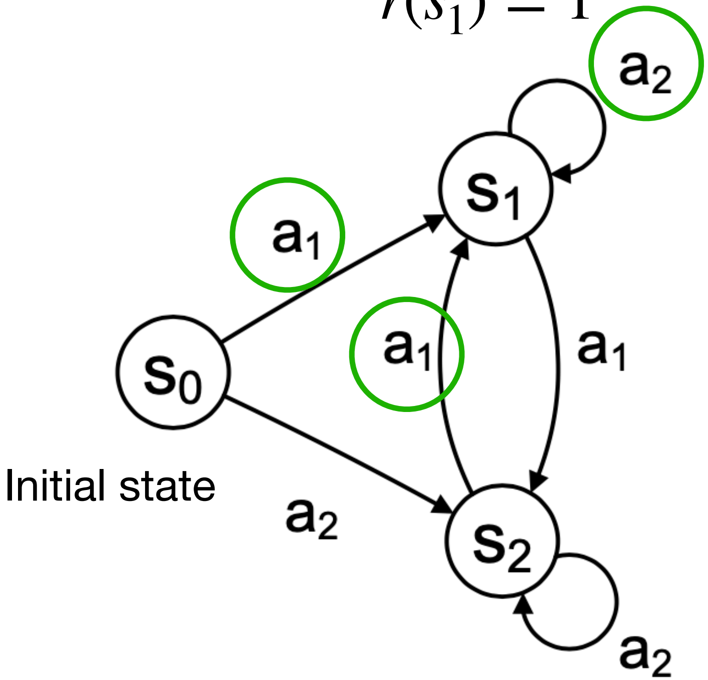
$$V_{s_0}^{\pi^\star} = H - 1$$

Assume SL returned such policy $\widehat{\pi}$

$$\widehat{\pi}(s_0) = \begin{cases} a_1 & \text{w/ prob } 1 - H\epsilon \\ a_2 & \text{w/ prob } H\epsilon \end{cases}, \quad \widehat{\pi}(s_1) = a_2, \widehat{\pi}(s_2) = a_2$$

Distribution Shift: Example (finite horizon case)

$r(s_1) = 1$



$$d_{s_0}^{\pi^\star}(s_0) = \frac{1}{H},\ d_{s_0}^{\pi^\star}(s_1) = \frac{H-1}{H},\ d_{s_0}^{\pi^\star}(s_2) = 0$$

$$V_{s_0}^{\pi^\star} = H - 1$$

Assume SL returned such policy $\widehat{\pi}$

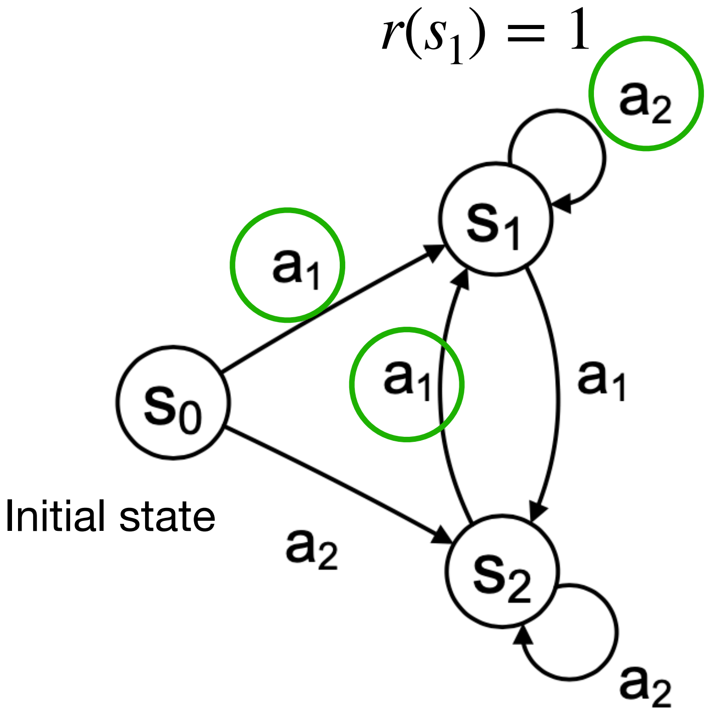
$$\widehat{\pi}(s_0) = \begin{cases} a_1 & \text{w/ prob } 1 - H\epsilon \\ a_2 & \text{w/ prob } H\epsilon \end{cases}, \quad \widehat{\pi}(s_1) = a_2, \widehat{\pi}(s_2) = a_2$$

This policy has good supervised learning error:

$$\mathbb{E}_{s\sim d_{s_0}^{\pi^\star}}\mathbb{E}_{a\sim\widehat{\pi}(\cdot|s)}\mathbf{1}\left(a \neq \pi^\star(s)\right) = \epsilon$$

$$\frac{1}{H}(H\varepsilon) + \frac{H-1}{H}\cdot 0 + 0\cdot 1 = \varepsilon$$

Distribution Shift: Example (finite horizon case)



$$d^{\pi^\star}_{s_0}(s_0) = \frac{1}{H},\ d^{\pi^\star}_{s_0}(s_1) = \frac{H-1}{H},\ d^{\pi^\star}_{s_0}(s_2) = 0$$

$$V^{\pi^\star}_{s_0} = H - 1$$

Assume SL returned such policy $\widehat{\pi}$

$$\widehat{\pi}(s_0) = \begin{cases} a_1 & \text{w/ prob } 1 - H\epsilon \\ a_2 & \text{w/ prob } H\epsilon \end{cases}, \quad \widehat{\pi}(s_1) = a_2, \widehat{\pi}(s_2) = a_2$$

This policy has good supervised learning error:

$$\mathbb{E}_{s\sim d^{\pi^\star}_{s_0}}\mathbb{E}_{a\sim\widehat{\pi}(\cdot|s)}\mathbf{1}\left(a \neq \pi^\star(s)\right) = \epsilon$$

$O(\varepsilon H^2)$

But we have quadratic error (in H) in performance:

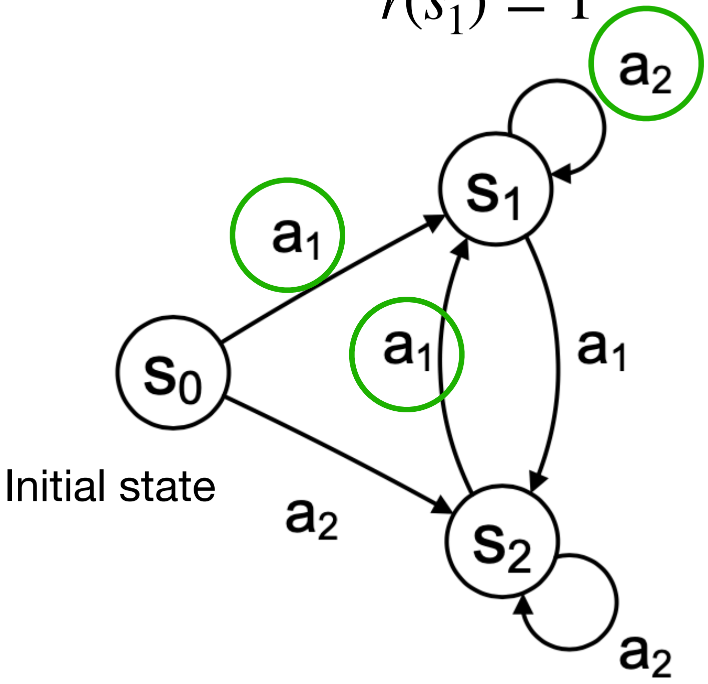
$$V^{\widehat{\pi}}_{s_0} = (1 - H\epsilon)\cdot V^{\pi^\star}_{s_0} + H\epsilon\cdot 0 = V^{\pi^\star}_{s_0} - \epsilon H(H-1)$$

$= (1 - H\varepsilon)V^{\pi^\star}_{s_0} + (H\varepsilon)\cdot 0$

$= V^\star_{s_0} - H\varepsilon V^\star_{s_0} =$

Distribution Shift: Example (finite horizon case)

$r(s_1) = 1$



$$d_{s_0}^{\pi^\star}(s_0) = \frac{1}{H},\ d_{s_0}^{\pi^\star}(s_1) = \frac{H-1}{H},\ d_{s_0}^{\pi^\star}(s_2) = 0$$

$$V_{s_0}^{\pi^\star} = H - 1$$

Assume SL returned such policy $\widehat{\pi}$

$$\widehat{\pi}(s_0) = \begin{cases} a_1 & \text{w/ prob } 1 - H\epsilon \\ a_2 & \text{w/ prob } H\epsilon \end{cases}, \quad \widehat{\pi}(s_1) = a_2, \widehat{\pi}(s_2) = a_2$$

This policy has good supervised learning error:

$$\mathbb{E}_{s\sim d_{s_0}^{\pi^\star}} \mathbb{E}_{a\sim\widehat{\pi}(\cdot|s)} \mathbf{1}\left(a \neq \pi^\star(s)\right) = \epsilon$$

But we have quadratic error (in H) in performance:

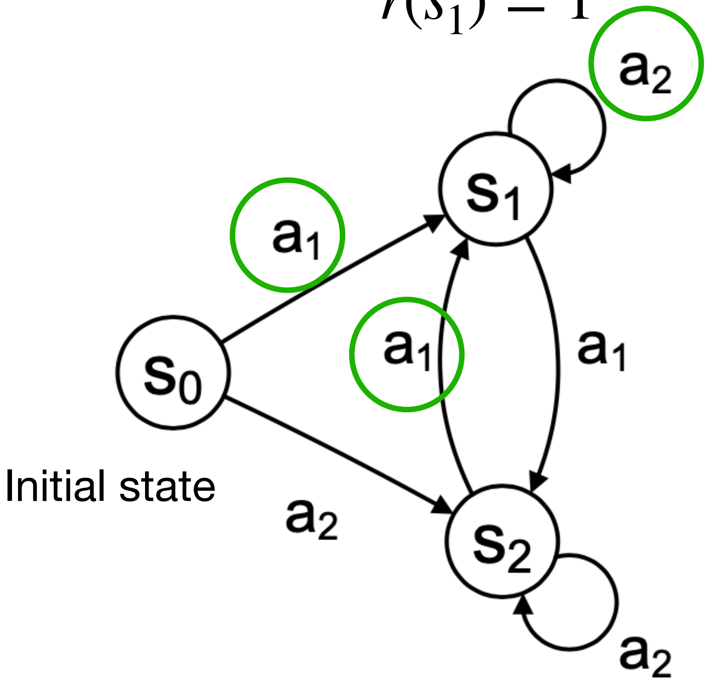
$$V_{s_0}^{\widehat{\pi}} = (1 - H\epsilon)\cdot V_{s_0}^{\pi^\star} + H\epsilon \cdot 0 = V_{s_0}^{\pi^\star} - \epsilon H(H-1)$$

Issue: once we make a mistake at s_0, we end up in s_2 which is not in the training data!

Distribution Shift: Example (discounted case)

look at offline

$r(s_1) = 1$



$d_{s_0}^{\pi^\star}(s_0) = 1 - \gamma,\ d_{s_0}^{\pi^\star}(s_1) = \gamma,\ d_{s_0}^{\pi^\star}(s_2) = 0$

$$V_{s_0}^{\pi^\star} = \frac{\gamma}{1-\gamma}$$

Assume SL returned such policy $\widehat{\pi}$

$$\widehat{\pi}(s_0) = \begin{cases} a_1 & \text{w/ prob } 1 - \epsilon/(1-\gamma) \\ a_2 & \text{w/ prob } \epsilon/(1-\gamma) \end{cases}, \quad \widehat{\pi}(s_1) = a_2, \widehat{\pi}(s_2) = a_2$$

We will have good supervised learning error:

$$\mathbb{E}_{s \sim d_{s_0}^{\pi^\star}} \mathbb{E}_{a \sim \widehat{\pi}(\cdot|s)} \mathbf{1}\left(a \neq \pi^\star(s)\right) = \epsilon$$

But we have quadratic error in performance:

$$V_{s_0}^{\widehat{\pi}} = \frac{\gamma}{1-\gamma} - \frac{\epsilon\gamma}{(1-\gamma)^2} = V_{s_0}^{\pi^\star} - \frac{\epsilon\gamma}{(1-\gamma)^2}$$

Issue: once we make a mistake at s_0, we end up in s_2 which is not in the training data!


Ready!

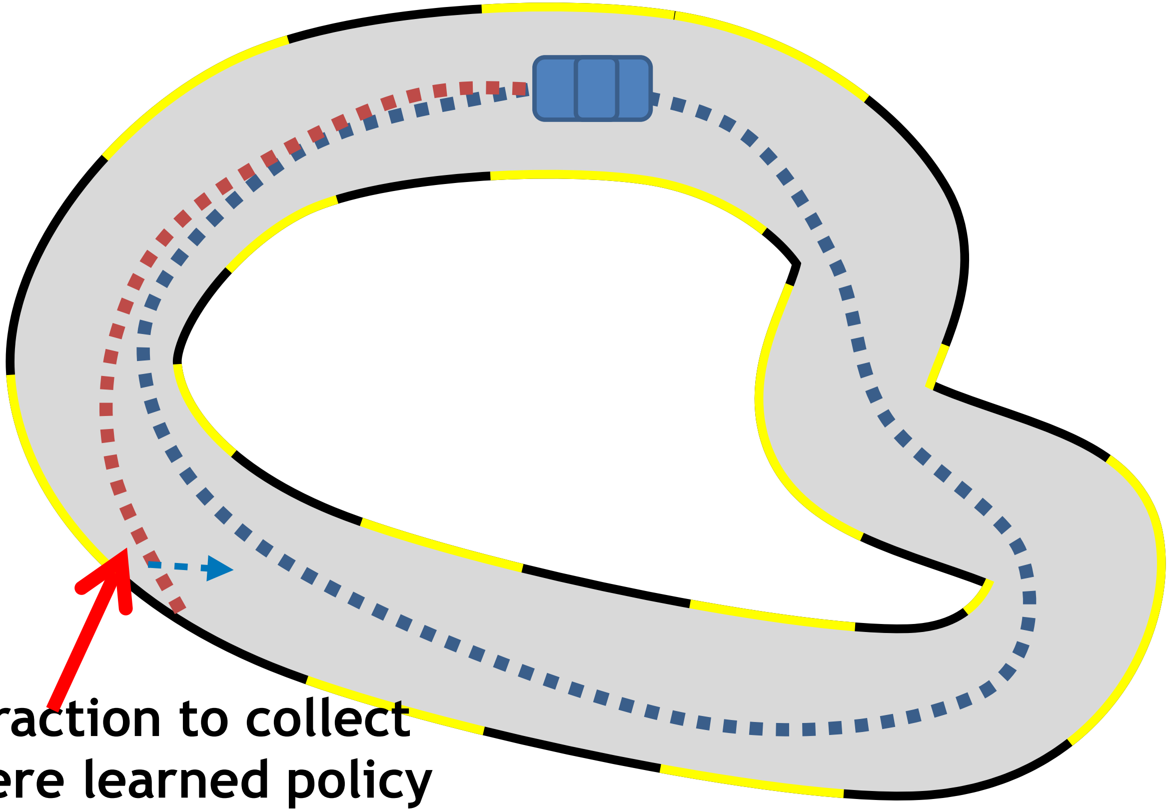

Ready!


Ready!

Today:

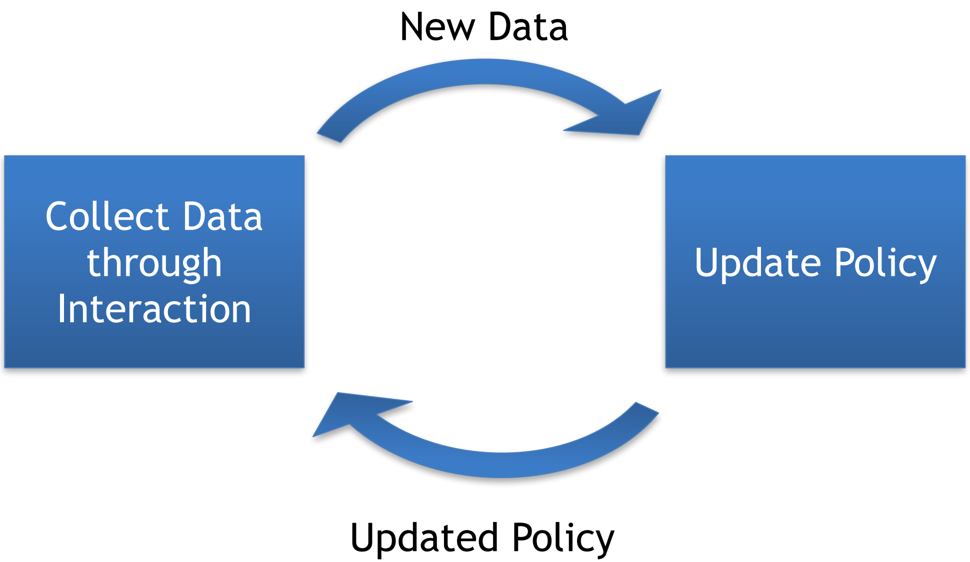
More Imitation Learning

Intuitive solution: **Interaction**



Use interaction to collect data where learned policy goes

General Idea: Iterative Interactive Approach



All DAgger slides credit: Drew Bagnell, Stephane Ross, Arun Venktraman

Outline for today:

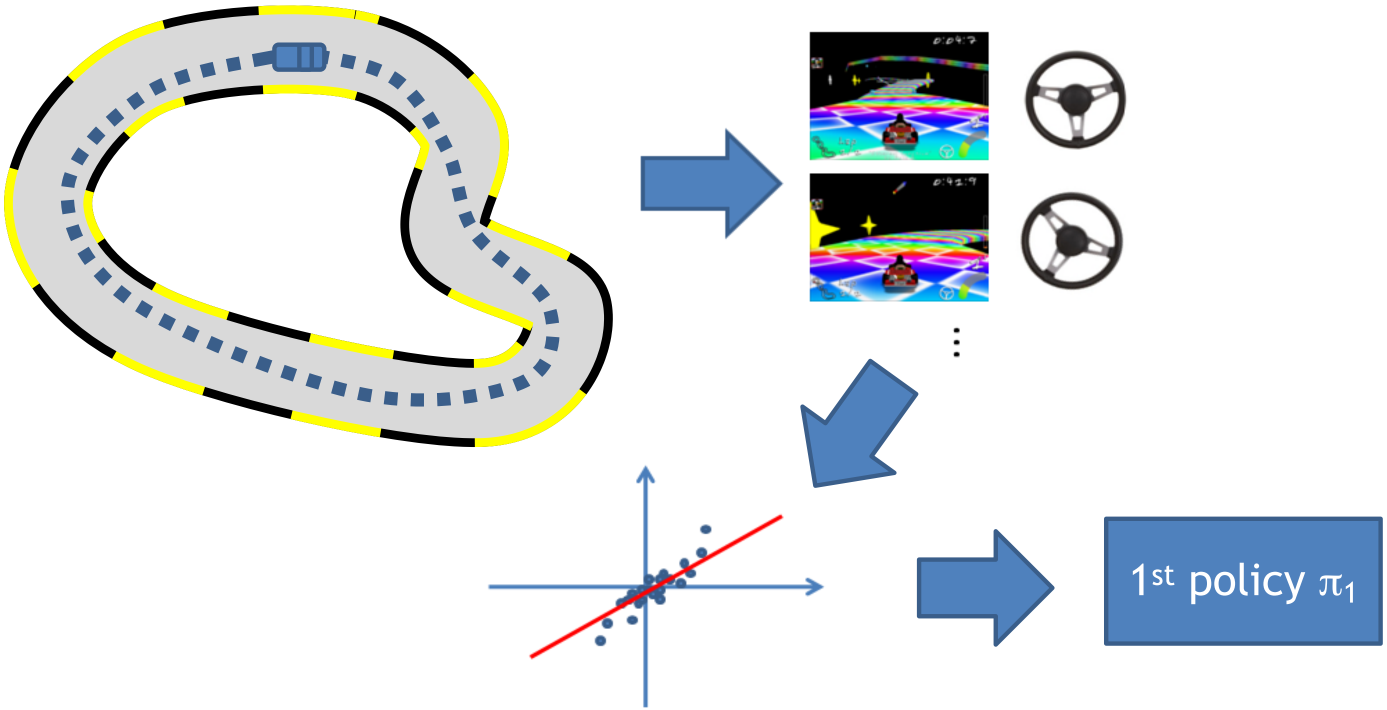
1. The DAgger (Data Aggregation) Algorithm

[Ross11a]

DAgger: Dataset Aggregation

0th iteration

Expert Demonstrates Task **Dataset**

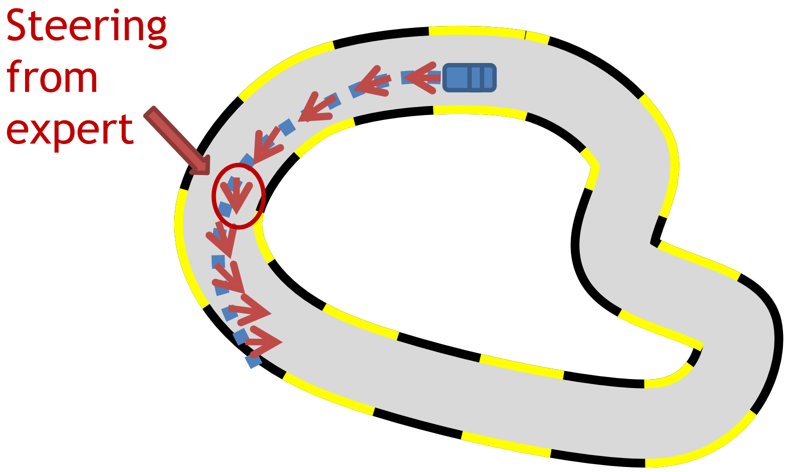


Supervised Learning

DAgger: Dataset Aggregation [Ross11a]

1st iteration

Execute π_1 and Query Expert

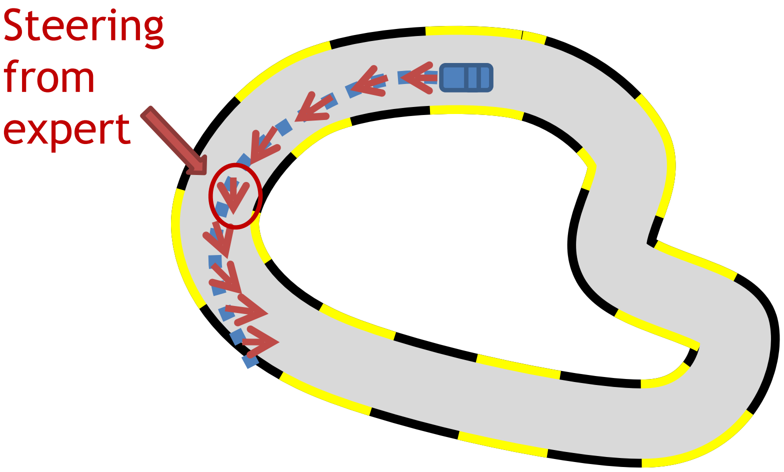


[Ross11a]

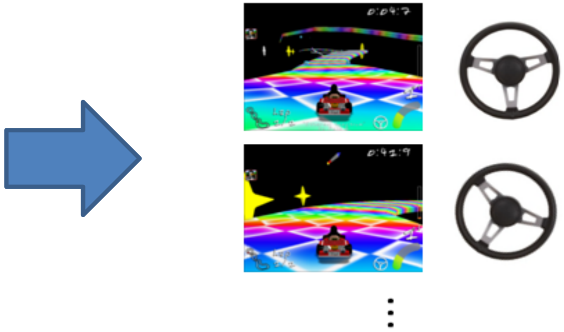
DAgger: Dataset Aggregation

1st iteration

Execute π_1 and Query Expert



New Data



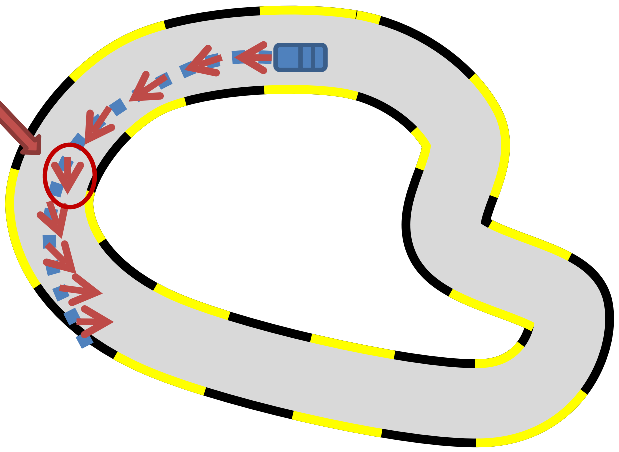
[Ross11a]

DAgger: Dataset Aggregation

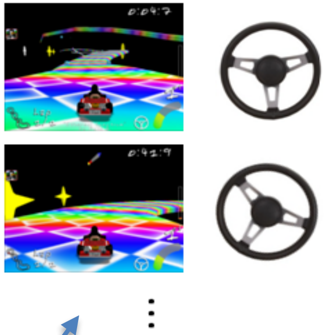
1st iteration

Execute π_1 and Query Expert

Steering from expert



New Data

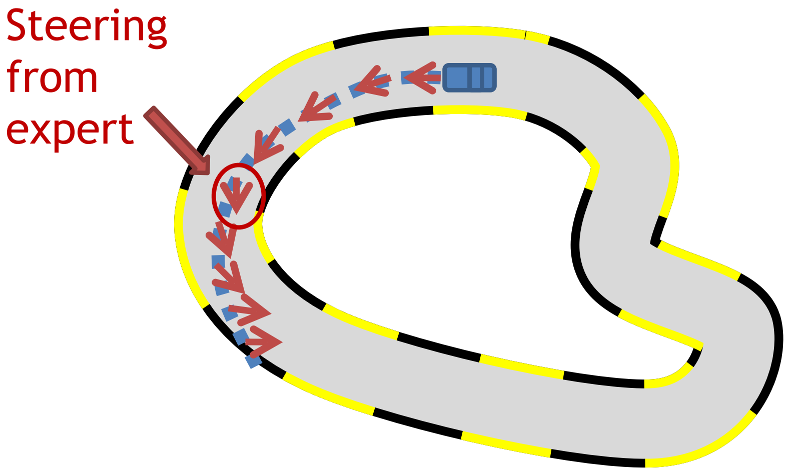


States from
the learned policy

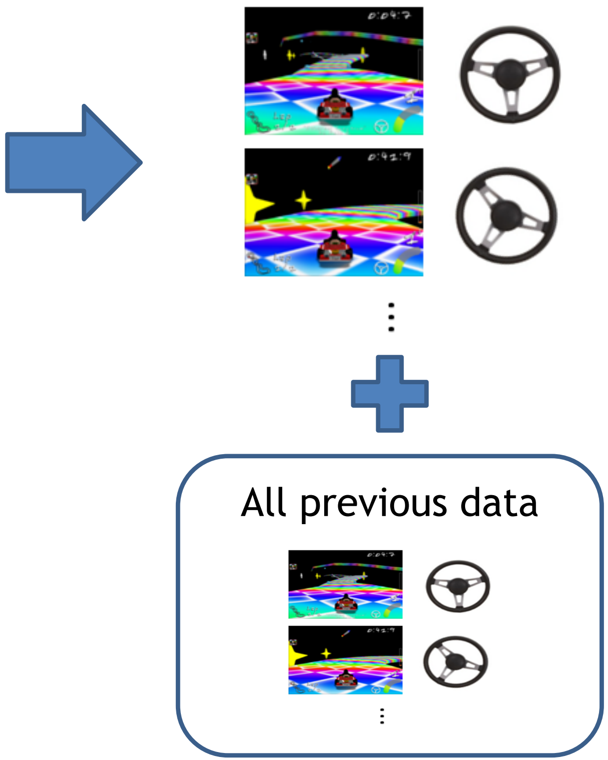
DAgger: Dataset Aggregation

1st iteration

Execute π_1 and Query Expert



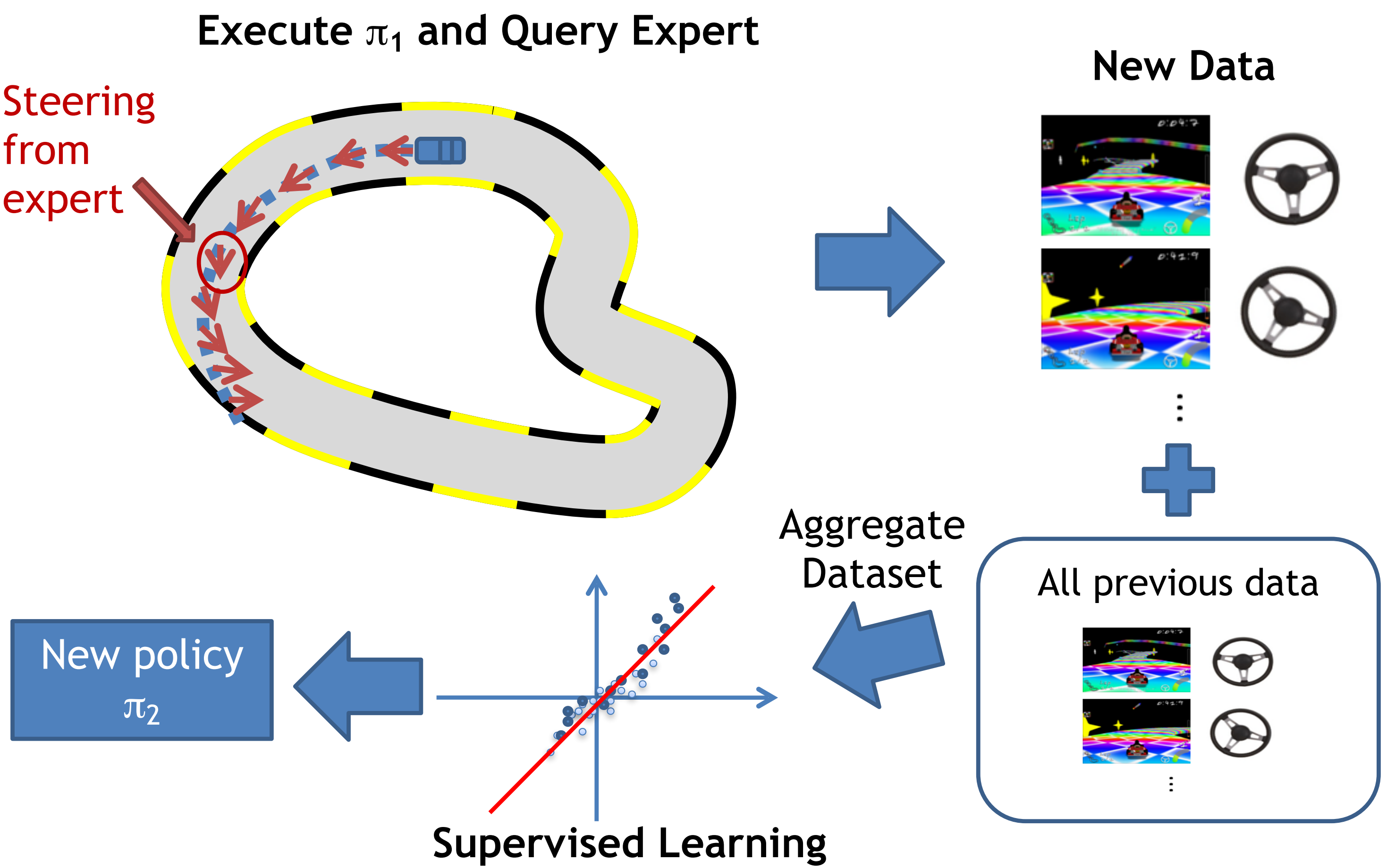
New Data



DAgger: Dataset Aggregation

[Ross11a]

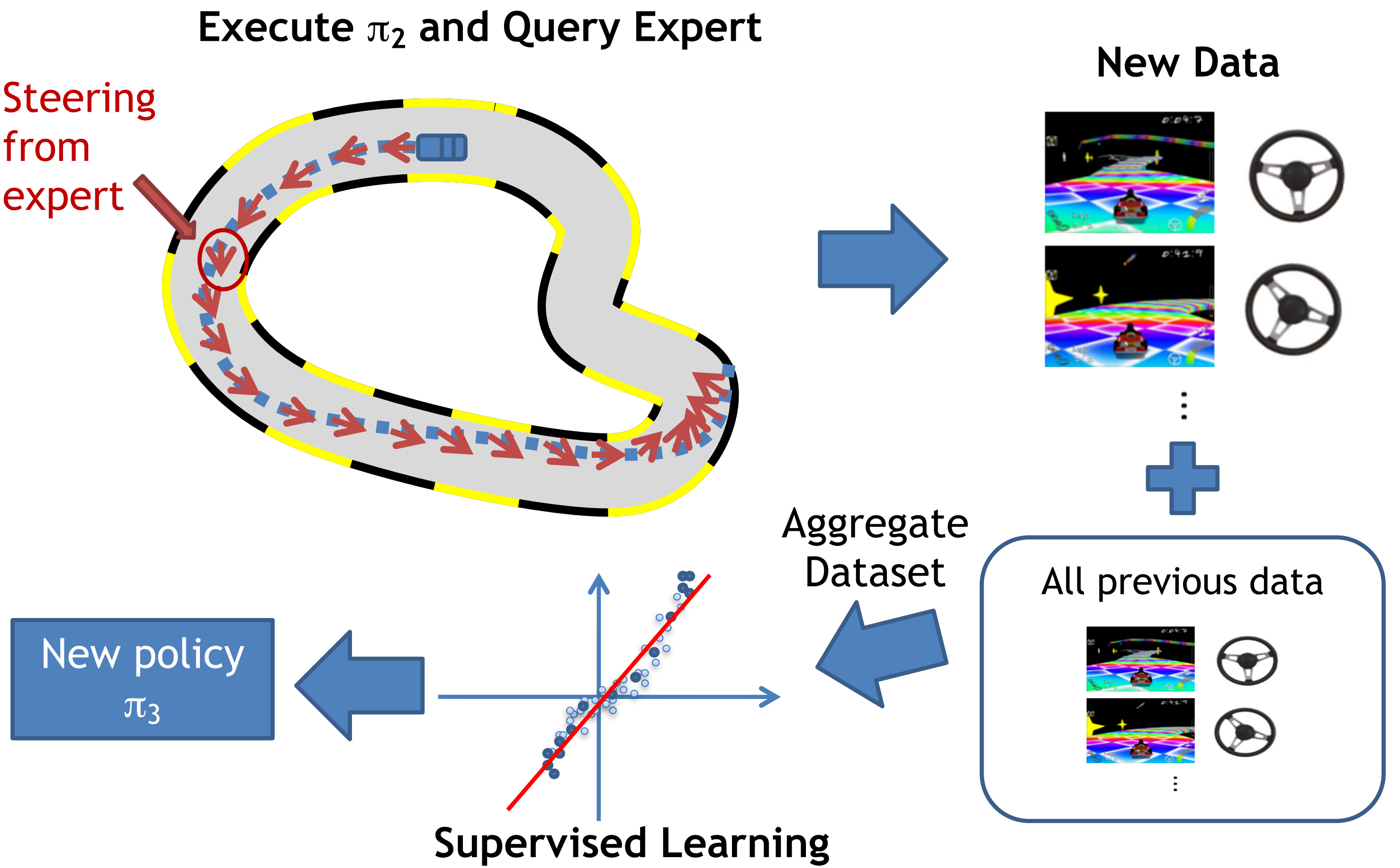
1st iteration



DAgger: Dataset Aggregation

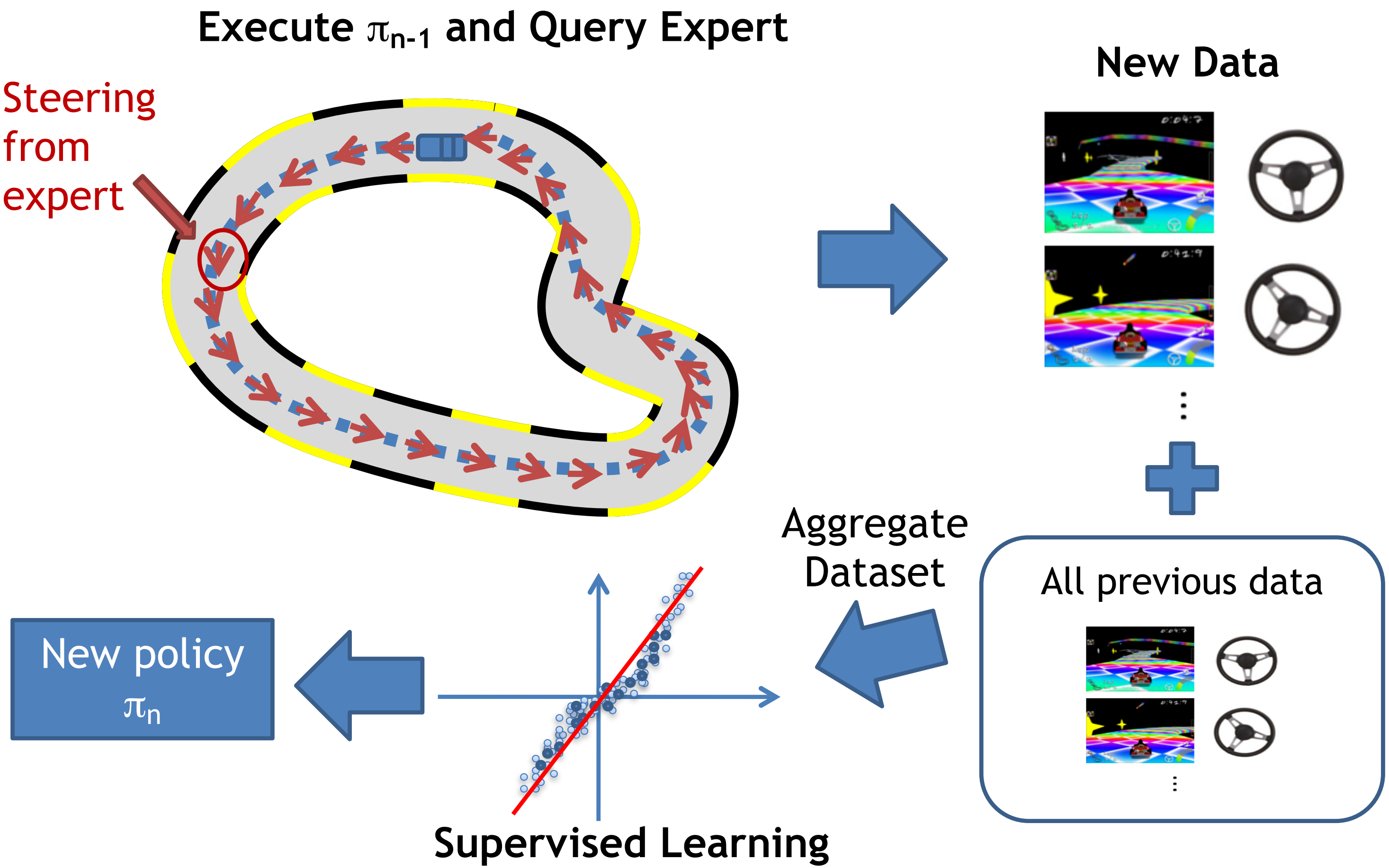
[Ross11a]

2nd iteration



DAgger: Dataset Aggregation [Ross11a]

n^{th} iteration



The DAgger algorithm

need a stronger oracle model than in BC.

Initialize π^0, and dataset $\mathscr{D} = \emptyset$

For $t = 0 \rightarrow T-1$:

1. W/ π^t, generate dataset $\mathscr{D}^t = \{s_i, a_i^\star\}, s_i \sim d_\mu^{\pi^t}, a_i^\star = \pi^\star(s_i)$
2. **Data aggregation:** $\mathscr{D} = \mathscr{D} \cup \mathscr{D}^t$
3. **Update policy via Supervised-Learning**: $\pi^{t+1} = \text{SL}\left(\mathscr{D}\right)$

Success!

[Ross AISTATS 2011]

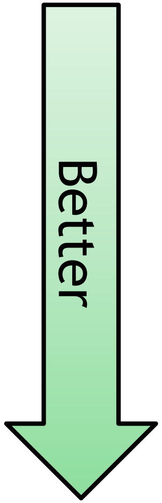
Success!

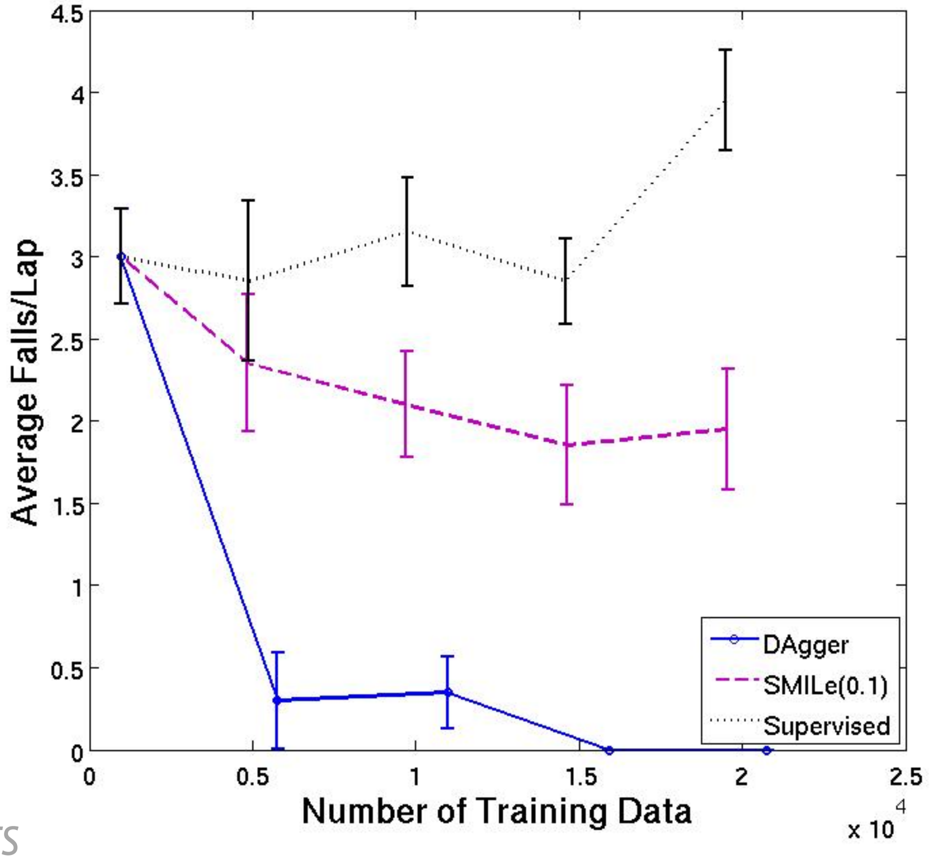
[Ross AISTATS 2011]

Success!

[Ross AISTATS 2011]

Average Falls/Lap





[Ross AISTATS

Roughly, the DAgger algorithm requires less human labeled data than BC.

[Informal Theorem]

Assuming ϵ SL error is achievable.
The DAgger algorithm has error:

$$V^{\pi^\star} - V^{\hat{\pi}} \le \frac{2}{(1-\gamma)}\epsilon \quad 2H\epsilon$$

while BC has error:

$$V^{\pi^\star} - V^{\hat{\pi}} \le \frac{2}{(1-\gamma)^2}\epsilon$$

$2H^2\epsilon$

Imitation Learning from Observation Alone

	BC
s^* and $d^{\pi^*}_{s_0}$	s^*, a^* and $d^{\pi^*}_{s_0}$

+ ① we can act in the world.
+ ② suppose we don't see any rewards.

Goal: learn π s.t. $d^{\pi}_{s_0} \approx d^{\pi^*}_{s_0}$

have features $\vec{\phi}(s)$

consider loss

$$L(\pi) = \left| \mathbb{E}_{s \sim d^{\pi^*}_{s_0}} [\vec{\phi}(s)] - \mathbb{E}_{s \sim d^{\pi}_{s_0}} [\vec{\phi}(s)] \right|$$

so we want to "match" expected feature with that seen in the data. [where $|\vec{x}| = \sum_i |x_i|$]

---

for $\mathbb{E}_{s \sim d^{\pi^*}_{s_0}} [\vec{\phi}(s)]$ $\leftarrow$ can estimate from observed expert state trajectory

---

optimization: use $\{\pi_\theta \mid \theta \in \mathbb{R}^d\}$ and REINFORCE

$$\theta \leftarrow \theta - \eta \nabla L(\theta)$$

(REINFORCE can be used to compute this gradient).

example features for "small" problems.

$\vec{\phi} \in \mathbb{R}^{|S|}$ and $\vec{\phi}(s) = e_s = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ "one hot" encoding with 1 in the s-th position.

$$E_{s \sim d^{\pi}_{s_0}}[\vec{\phi}] = \vec{d^{\pi}_{s_0}}$$ ← do you see why?

so if $L(\pi) = 0$

$\Downarrow$

$$d^{\pi}_{s_0}(s) = d^{\pi^*}_{s_0}(s) \quad \forall s.$$

how do we learn $\vec{\phi}$?

- GAN approach to ILO

- Learner $\leftarrow$ generator

make $L(\pi)$ to

be small

- feature update

by adversary

update ϕ to make

$$\left| E_{s \sim d_{s_0}^{\pi^*}} [\vec{\phi}(s)] - E_{s \sim d_{s_0}^{\pi}} [\vec{\phi}(s)] \right|$$ to be large.

Summary:

1. Example of error amplification
2. The DAgger algorithm

1-minute feedback form: https://bit.ly/3RHtIxy