Interactive Imitation Learning

Recap

The Behavior Cloning algorithm:

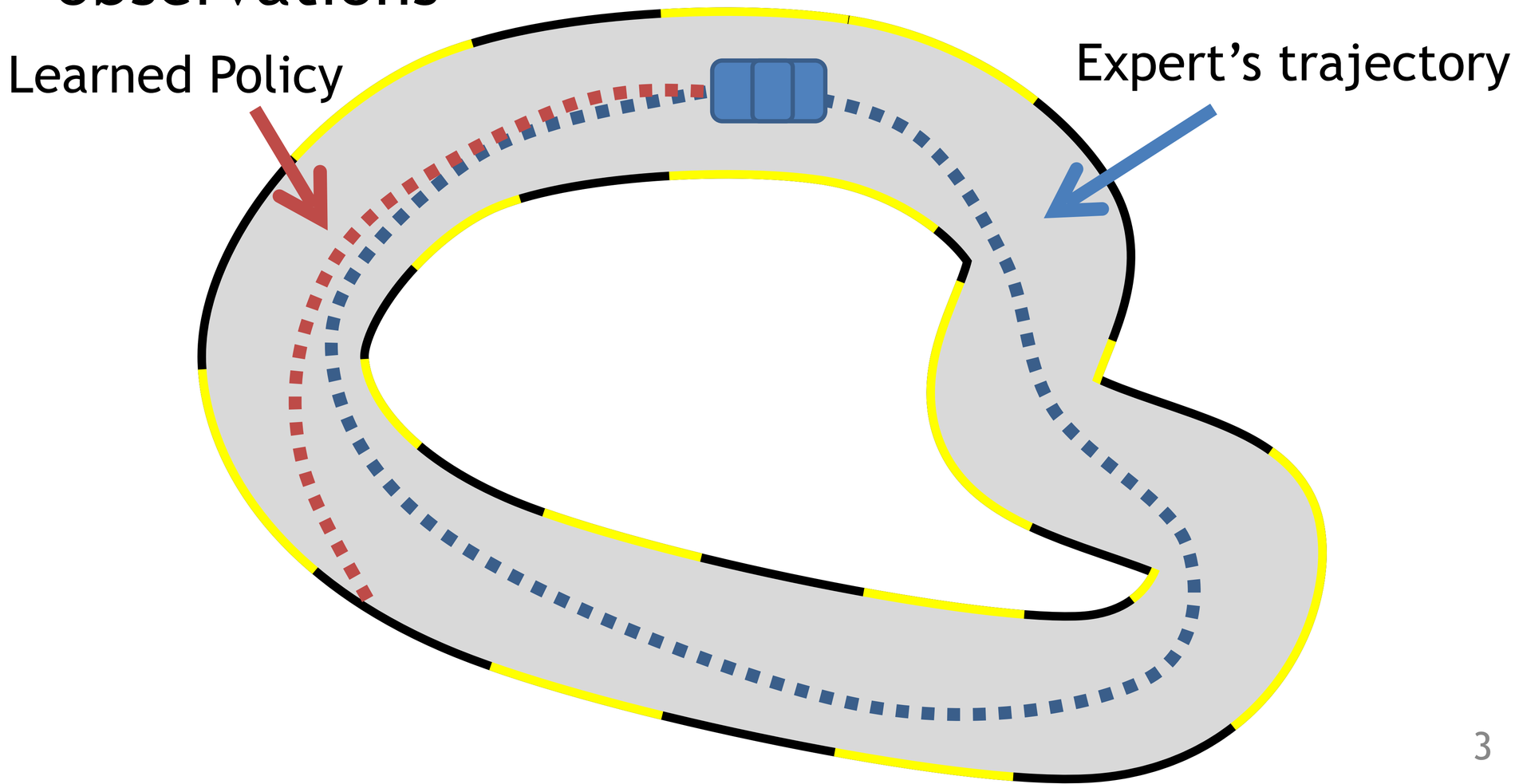
Choose regression (for continuous action) or classification loss $\ell(\pi(s), a)$, and perform SL:

$$\widehat{\pi} = \min_{\pi \in \Pi} \sum_{i=1}^{M} \ell(\pi(s_i^\star), a_i^\star)$$

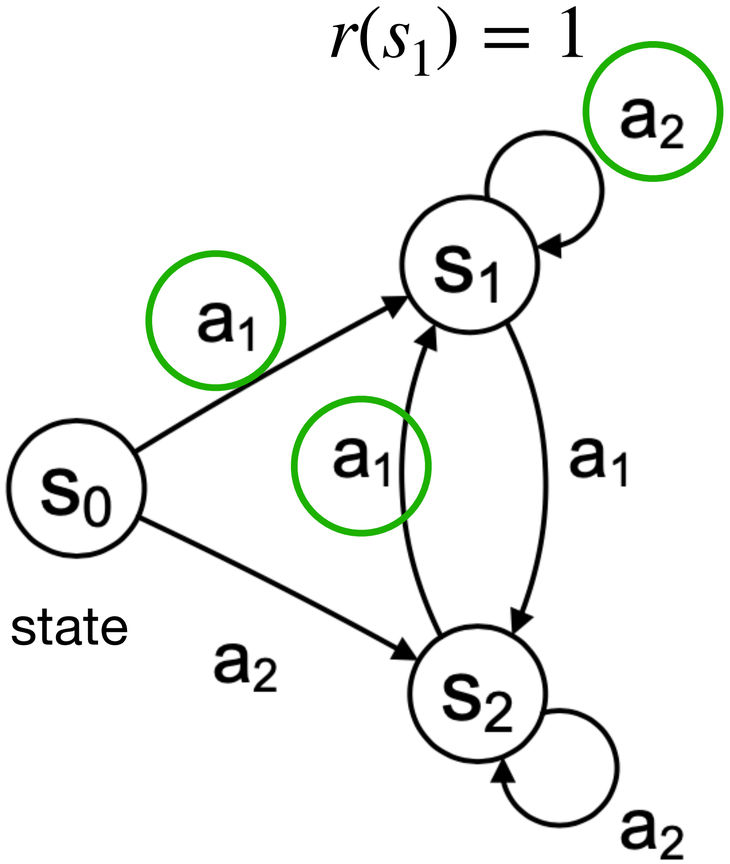
What could go wrong?

[Pomerleau89,Daume09]

- Predictions affect future inputs/ observations



Distribution Shift: Example



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

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

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!

An Autonomous Land Vehicle In A Neural Network *[Pomerleau, NIPS '88]*

"If the network is not presented with sufficient variability in its training exemplars to cover the conditions it is likely to encounter…[it] will perform poorly"

Question for today:

How to mitigate the distribution shift issue?

Solution:

Interactive Imitation Learning Setting

Key assumption:
we can query expert $\pi^\star$ at any time and any state during training

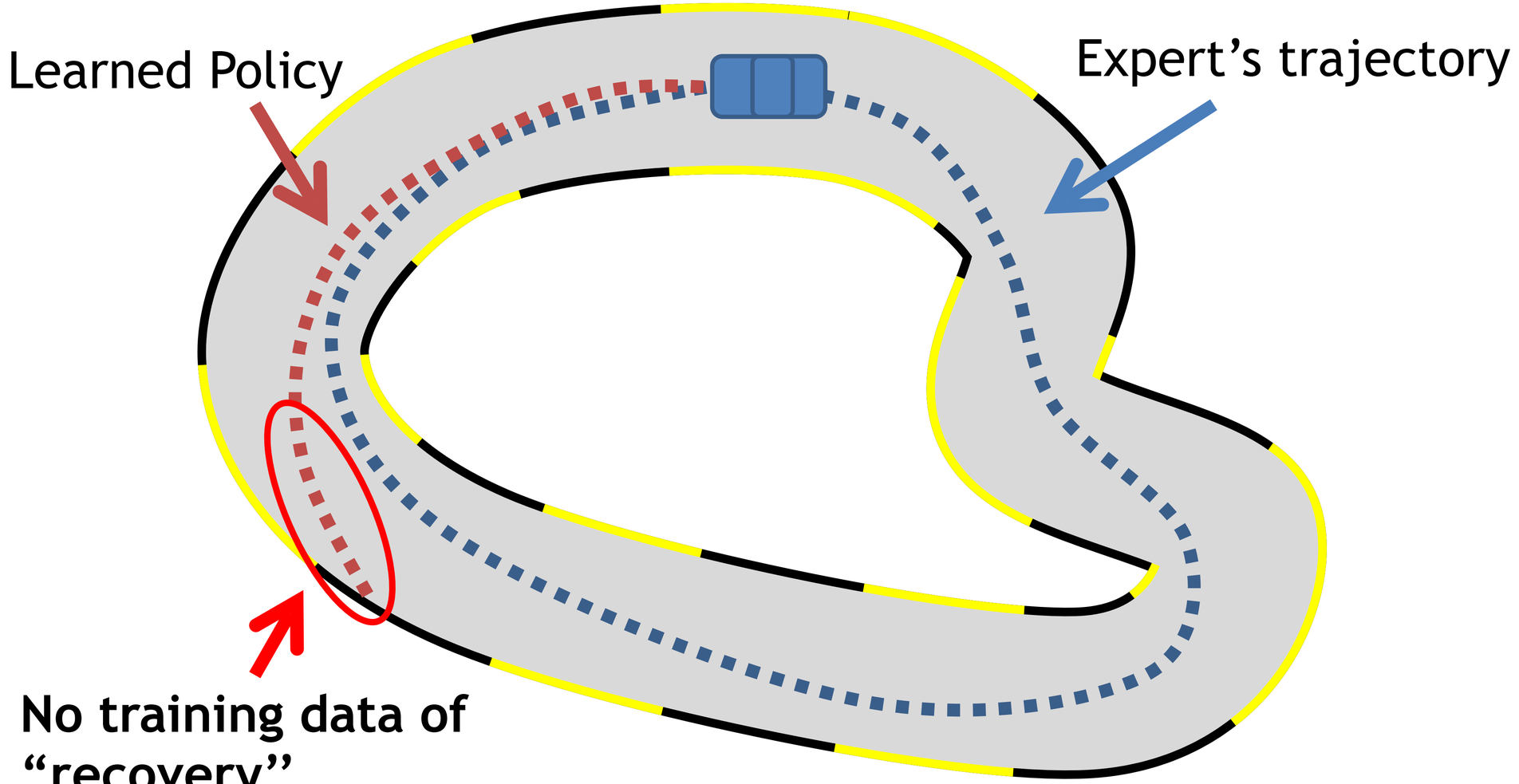
(Recall that previously we only had an offline dataset $\mathscr{D} = (s_i^\star, a_i^\star)_{i=1}^M \sim d_\mu^{\pi^\star}$)

Outline for today:

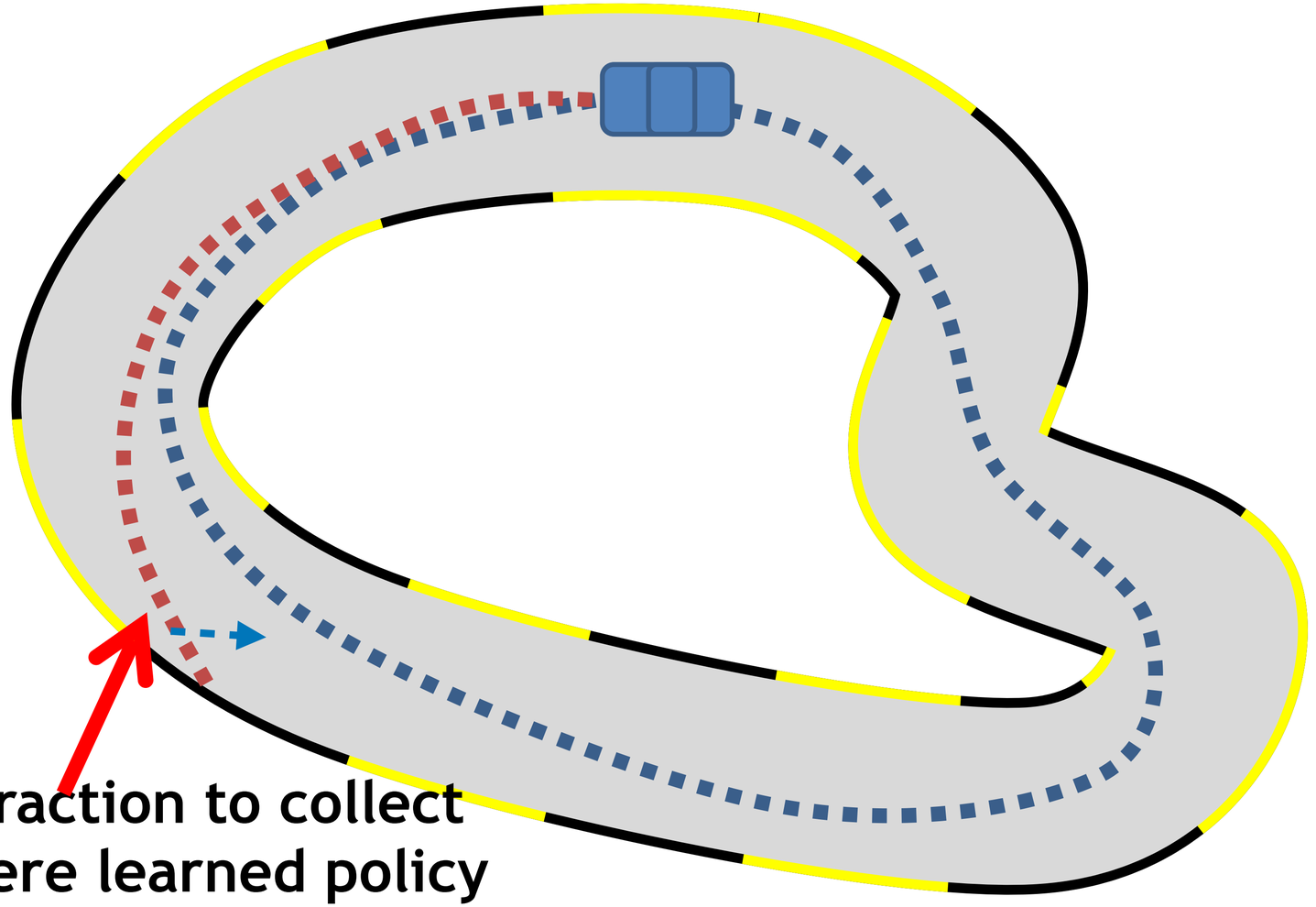
1. The DAgger (Data Aggregation) Algorithm

2. Analysis of DAgger: DAgger as online learning

Recall the Main Problem from Behavior Cloning:

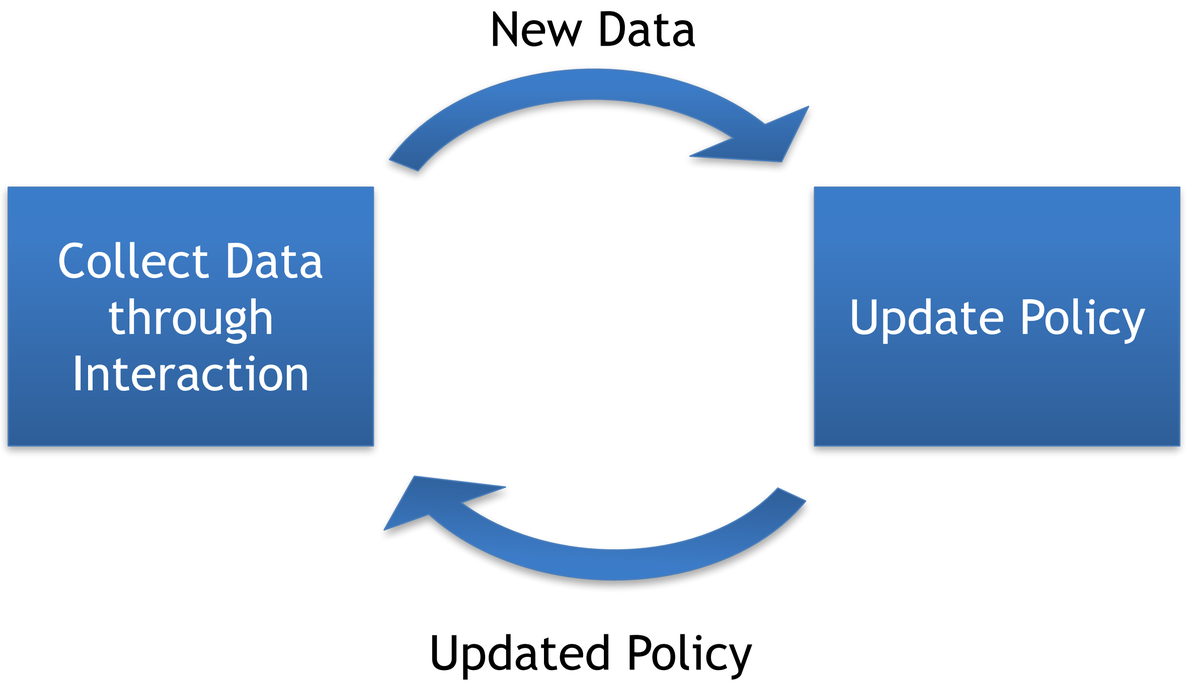


Intuitive solution: **Interaction**



Use interaction to collect data where learned policy goes

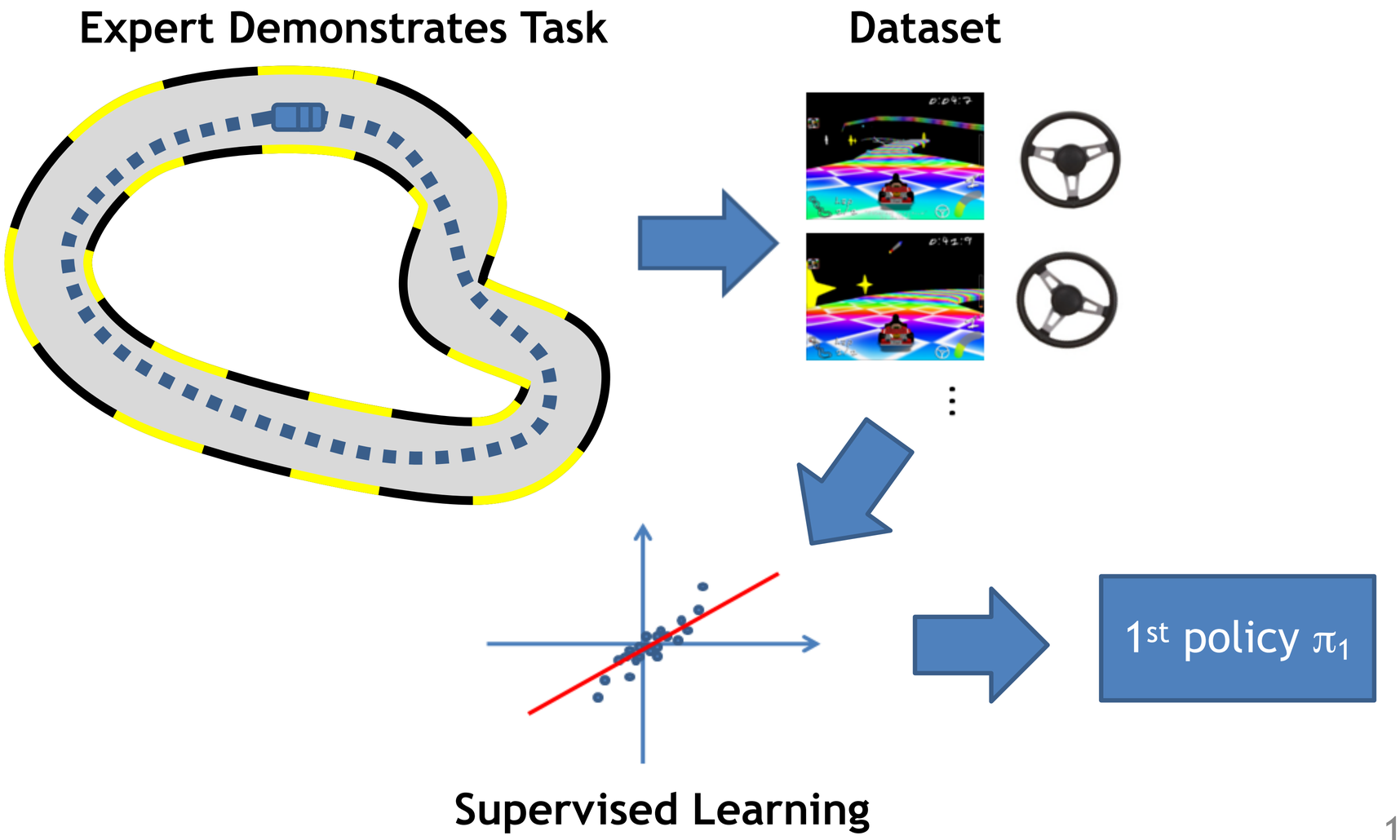
General Idea: Iterative Interactive Approach



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

DAgger: Dataset Aggregation [Ross11a]

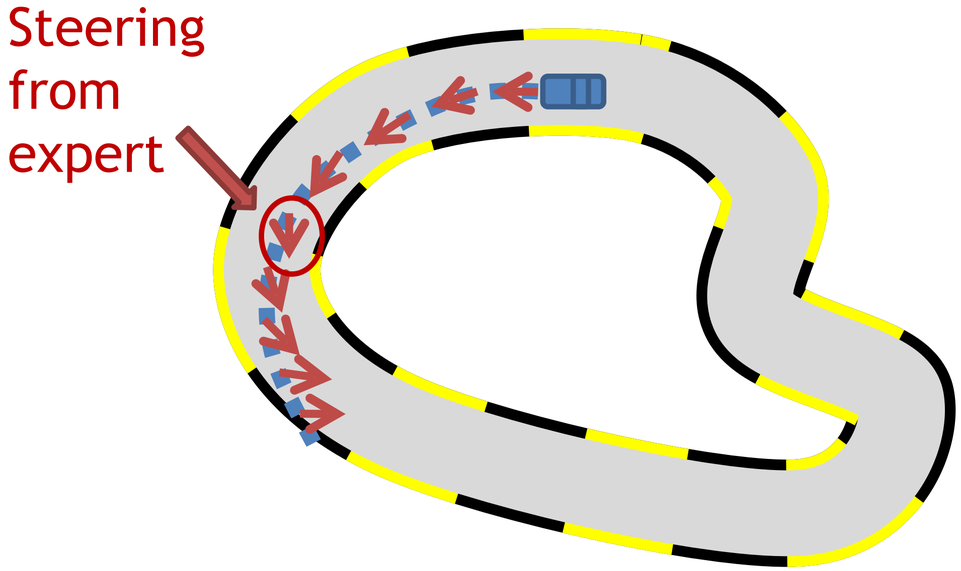
0th iteration



DAgger: Dataset Aggregation [Ross11a]

1st iteration

Execute π_1 and Query Expert

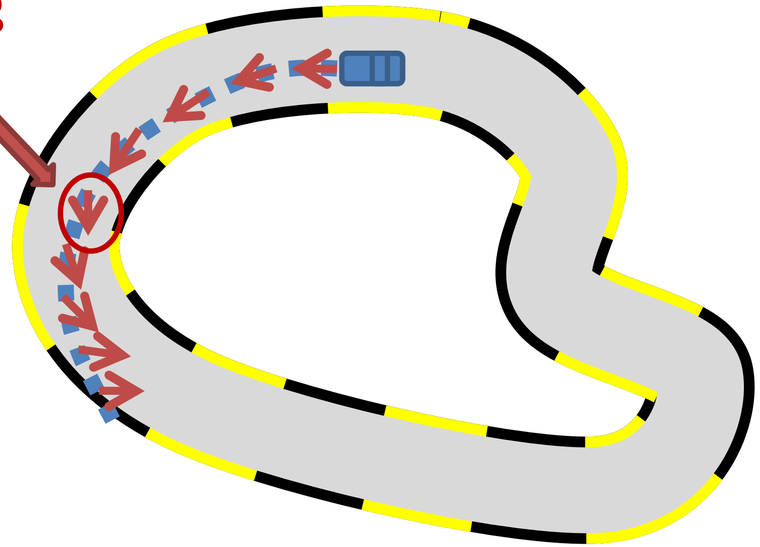


DAgger: Dataset Aggregation [Ross11a]

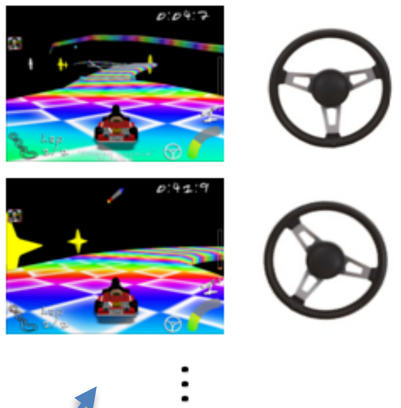
1st iteration

Execute π_1 and Query Expert

Steering from expert



New Data



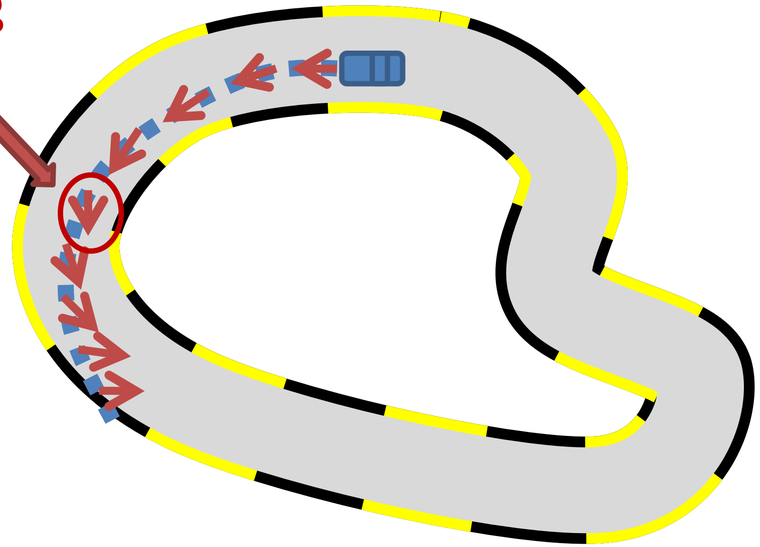
States from the learned policy

DAgger: Dataset Aggregation [Ross11a]

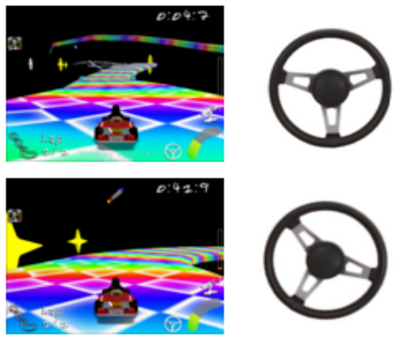
1st iteration

Execute π_1 and Query Expert

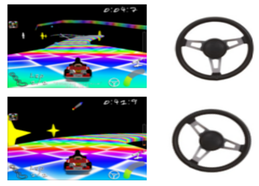
Steering from expert



New Data

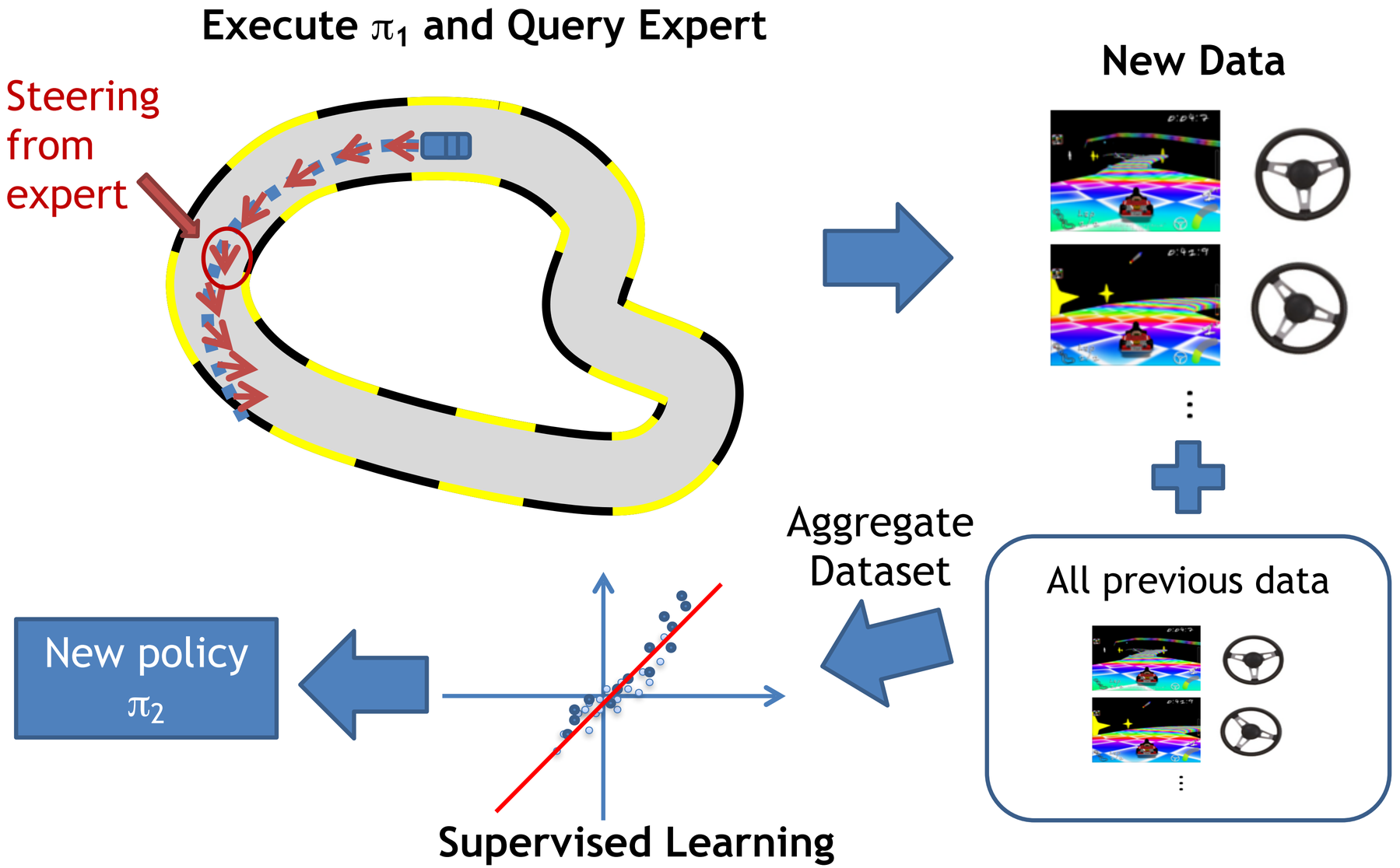


All previous data



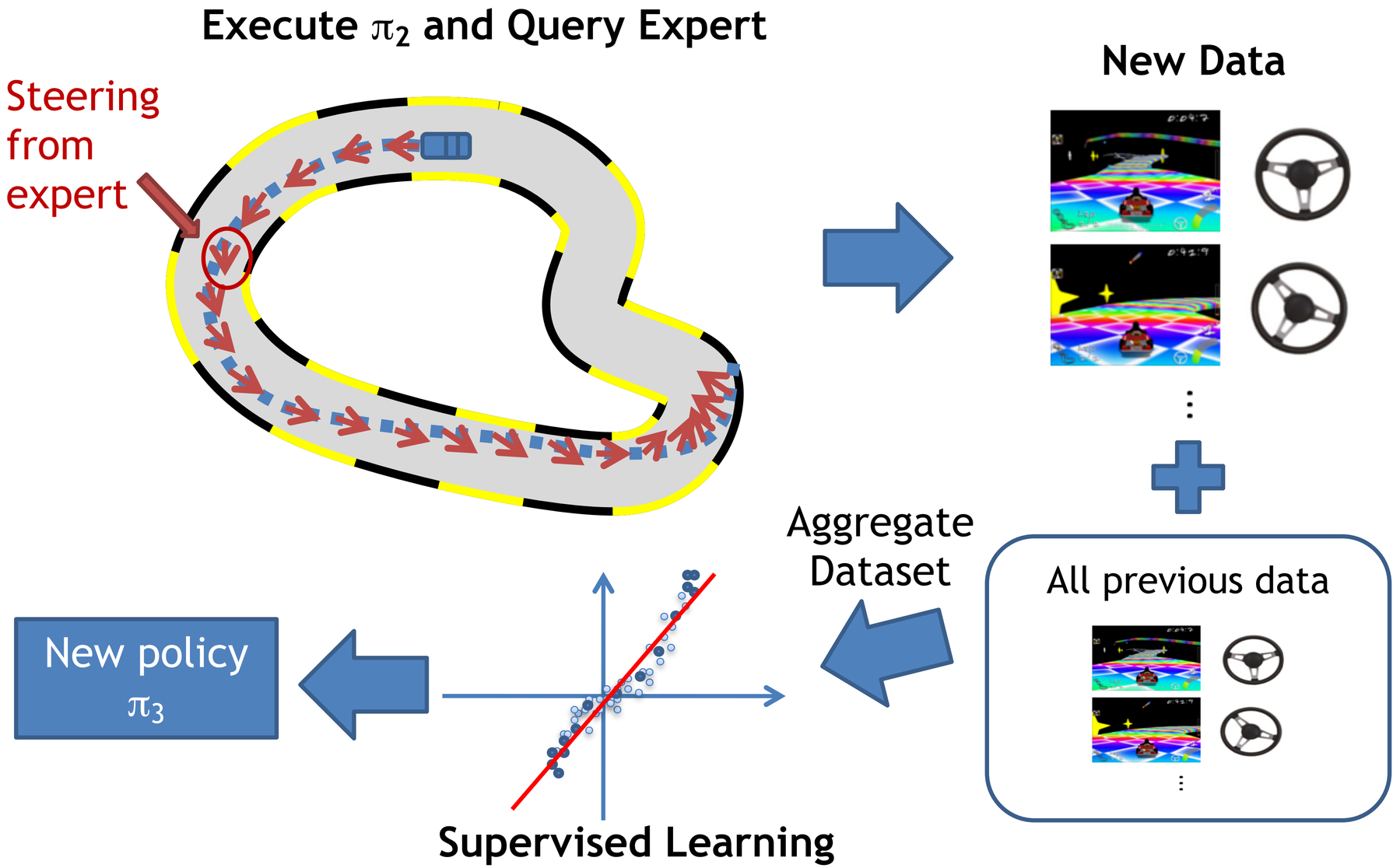
DAgger: Dataset Aggregation [Ross11a]

1st iteration



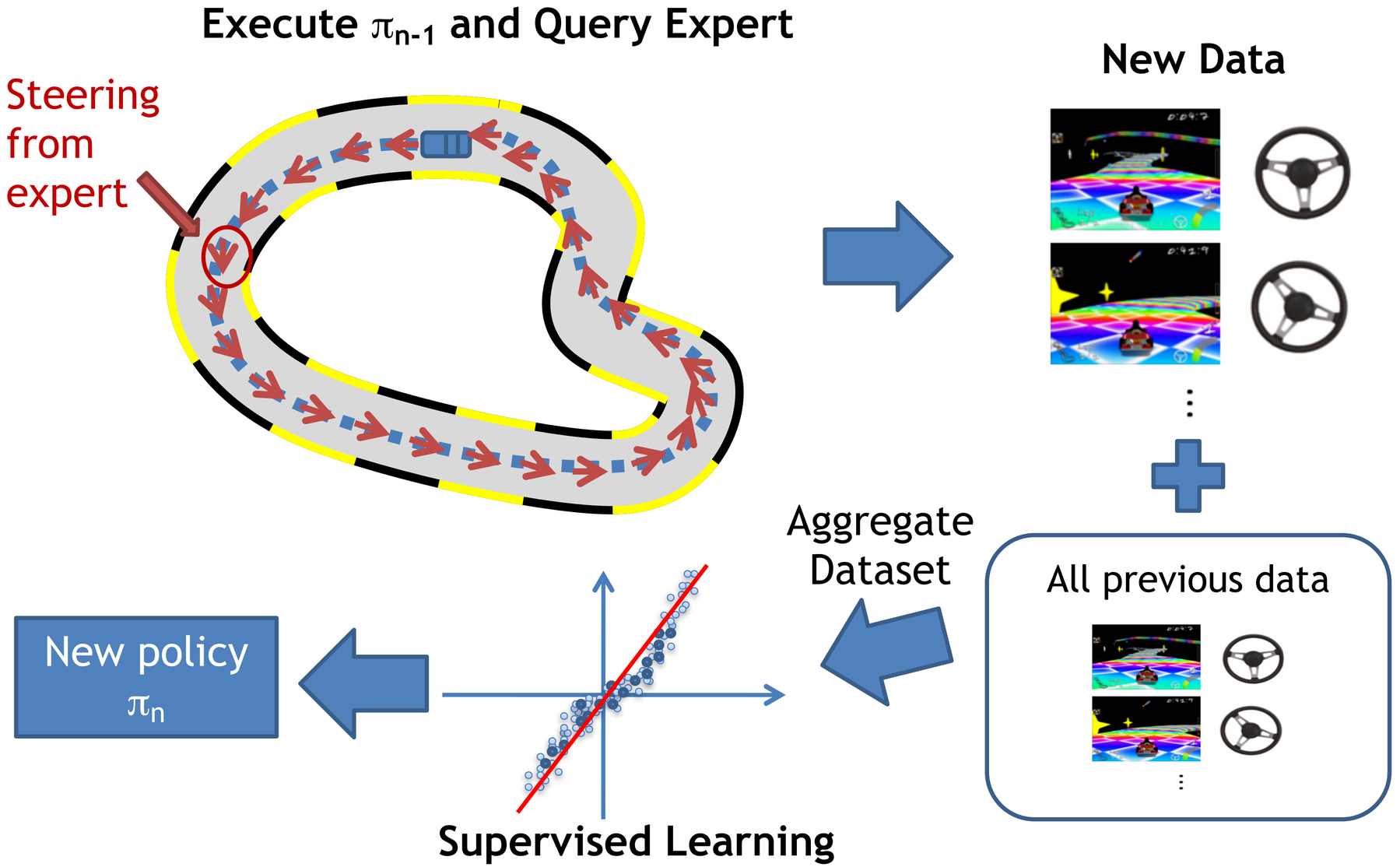
DAgger: Dataset Aggregation [Ross11a]

2nd iteration



DAgger: Dataset Aggregation [Ross11a]

n^{th} iteration



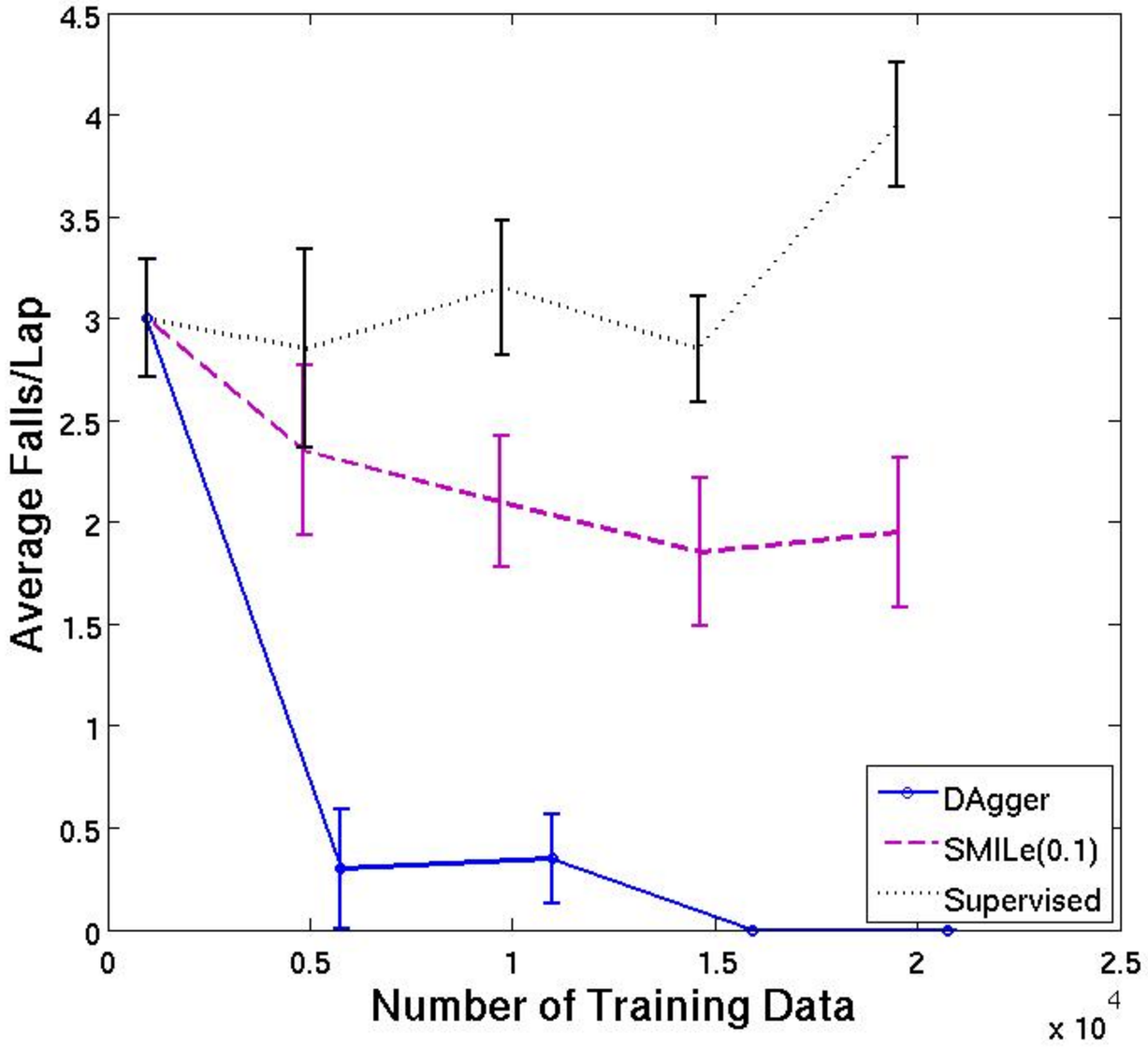
Success!

[Ross AISTATS 2011]

[Ross AISTATS 2011]

Average Falls/Lap

Better




FPS: 24
Attempt: 1 of 1
AgentLinear
Selected Actions:
RIGHT
SPEED

More fun than Video Games…

[Ross ICRA 2013]

Forms of the Interactive Experts

Interactive Expert is expensive, especially when the expert is human…

But expert does not have to be human…

Example: high-speed off-road driving
[Pan et al, RSS 18, Best System Paper]

Fig. 4: The AutoRally car and the test track.

Goal: learn a racing control policy that maps from data on cheap on-board sensors (raw-pixel imagine) to low-level control (steer and throttle)

→ Steering + throttle

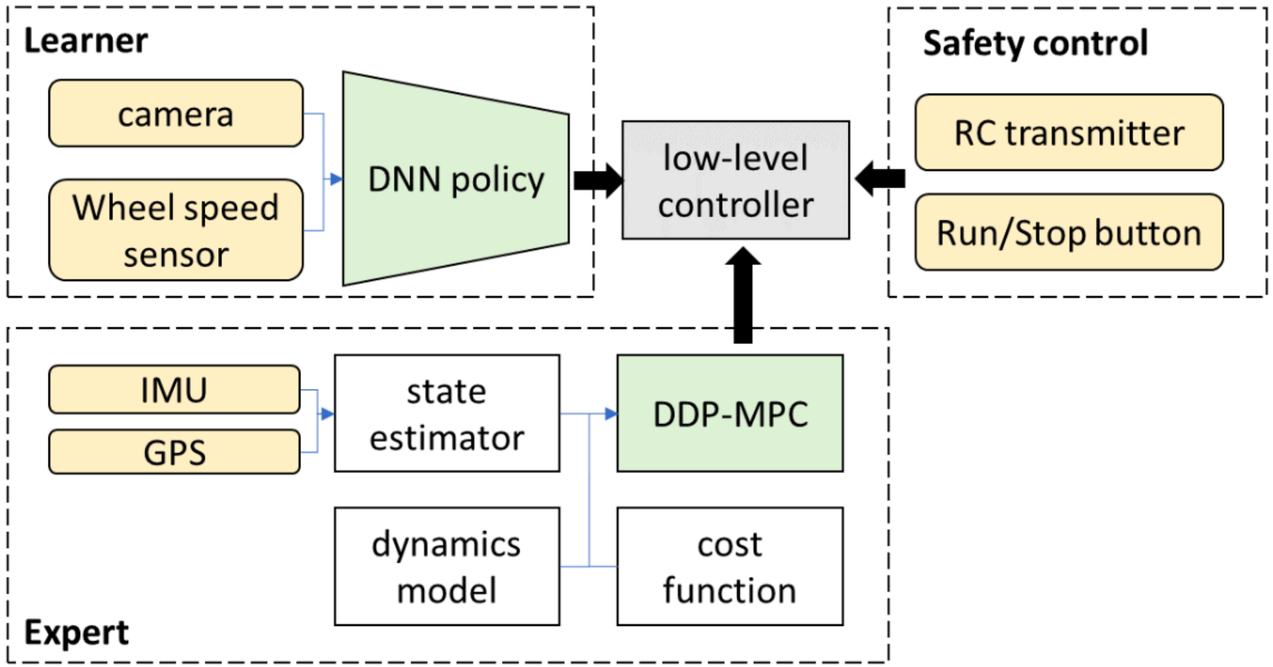
(a) raw image

Forms of the Interactive Experts

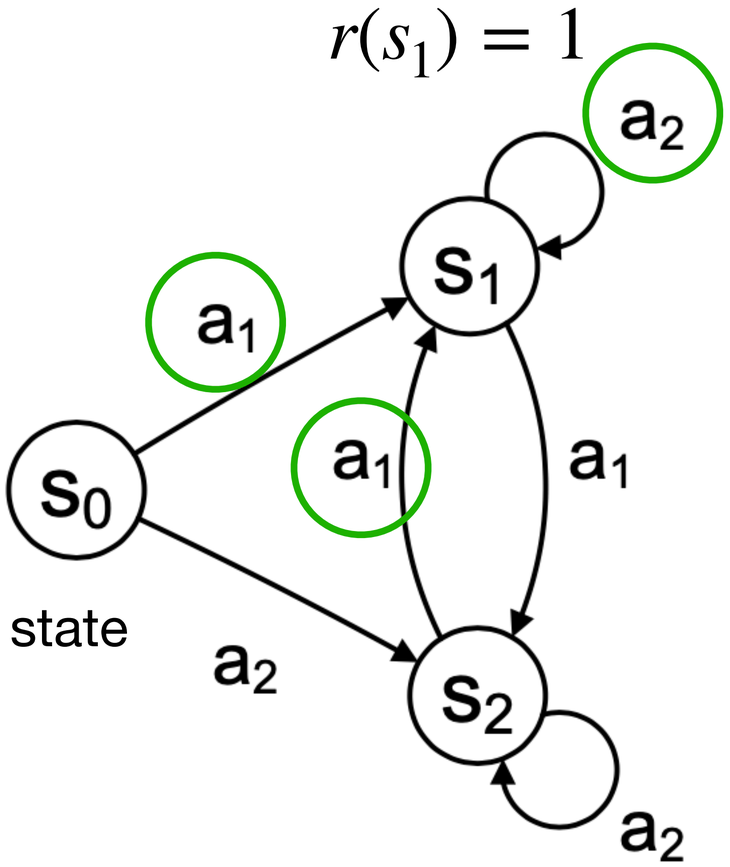
Example: high-speed off-road driving
[Pan et al, RSS 18, Best System Paper]

Their Setup:
At Training, we have expensive sensors for accurate state estimation
and we have computation resources for **MPC** (i.e., high-frequency replanning)

The MPC is the expert in this case!



Distribution Shift: Example



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

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

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

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

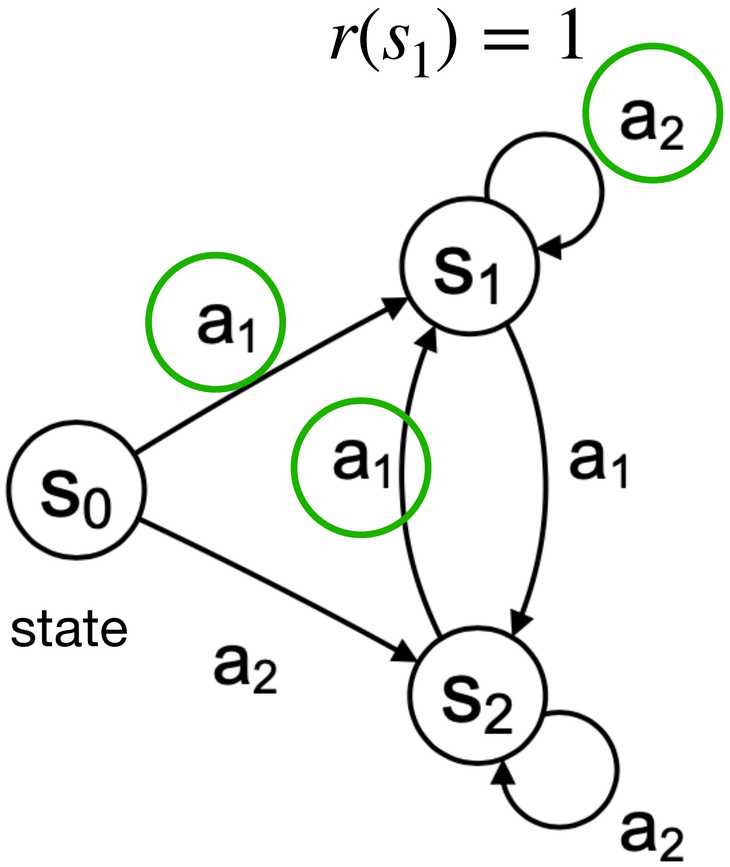
We will have good supervised learning error:

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

But we have quadratic error in performance:

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

Distribution Shift: Example



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

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

Why DAgger can fix this problem?

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

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

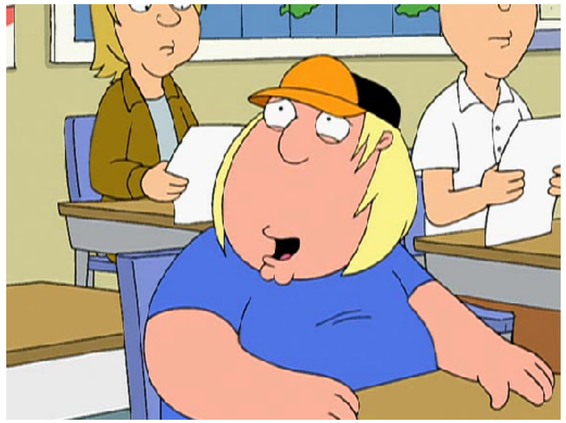
Outline for today:

1. The DAgger (Data Aggregation) Algorithm

2. Analysis of DAgger: DAgger as online learning

Online Learning

Learner



convex Decision set Θ

Learner picks a decision θ_0

Adversary picks a loss $\ell_0 : \Theta \to \mathbb{R}$

Learner picks a new decision θ_1

Adversary picks a loss $\ell_1 : \Theta \to \mathbb{R}$

...

Adversary

$$\text{Regret} = \sum_{t=0}^{T-1} \ell_t(\theta_t) - \min_{\theta \in \Theta} \sum_{t=0}^{T-1} \ell_t(\theta)$$

[Vovk92,Warmuth94,Freund97,Zinkevich03,Kalai05,Hazan06,Kakade08]

Example: online linear regression

Can we perform linear regression in online fashion with non i.i.d (or even adversary) data?

Every iteration t :

1. Learner first picks $\theta_t \in$ Ball $\subset \mathbb{R}^d$

2. Adversary **then** picks $x_t \in \mathcal{X} \subset \mathbb{R}^d, y_t \in [a, b]$

3. Learner suffers loss $\ell_t(\theta_t) = (\theta_t^\top x_t - y_t)^2$

Learner has to make decision θ_t based on history up to $t-1$,
while adversary could pick (x_t, y_t) even after seeing θ_t

Adversary seems too powerful…

Example: online linear regression

BUT, a very intuitive algorithm actually achieves no-regret property:

Every iteration t :

1. Learner first picks θ_t that minimizes the aggregated loss

$$\theta_t = \arg \min_{\theta \in \text{Ball}} \sum_{i=0}^{t-1} \left(\theta^\top x_i - y_i\right)^2 + \lambda \|\theta\|_2^2$$

This is called Follow-the-Regularized-Leader (FTRL), and it achieves no-regret property:

$$\sum_{i=0}^{T-1} \ell_i(\theta_i) - \min_{\theta \in \text{Ball}} \sum_{i=0}^{T-1} \ell_i(\theta) = O\left(\sqrt{T}\right)$$

Generally, Follow-the-Regularized-Leader is no-regret

At time step t, learner has seen $\ell_0, \ldots \ell_{t-1}$, which new decision she could pick?

$$\textbf{FTRL: } \theta_t = \min_{\theta \in \Theta} \sum_{i=0}^{t-1} \ell_i(\theta) + \lambda R(\theta)$$

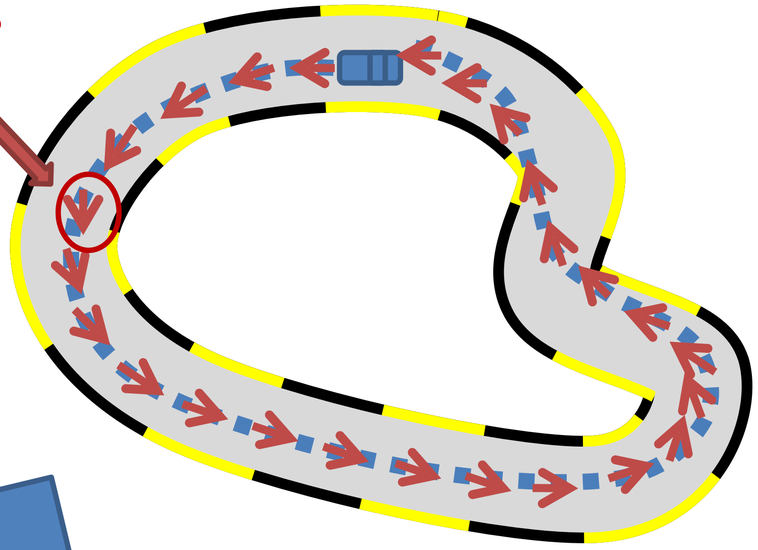
Informal Theorem (FTRL): when things are convex, FTRL is no-regret, i.e.,

$$\frac{1}{T}\left[\sum_{t=0}^{T-1} \ell_t(\theta_t) - \min_{\theta \in \Theta} \sum_{t=0}^{T-1} \ell_t(\theta)\right] = O\left(1/\sqrt{T}\right)$$

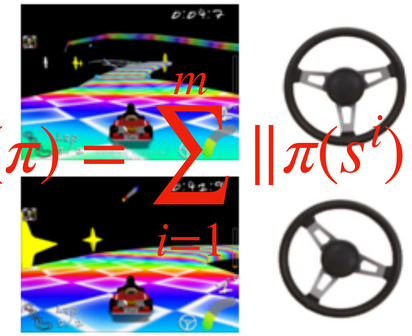
DAgger Revisit

At iteration t:

Steering from expert



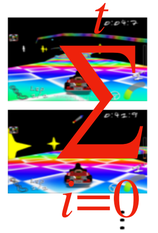
New Data



$$\ell_t(\pi) = \sum_{i=1}^{m} \|\pi(s^i) - \pi^\star(s^i)\|_2^2$$

Aggregate Dataset

All previous data



$$\sum_{i=0}^{t} \ell_i(\pi)$$

$$\pi_t = \arg\min_{\pi} \sum_{i=0}^{t} \ell_i(\pi) + \lambda R(\pi)$$

New policy π_n

Supervised Learning

Data Aggregation = Follow-the-Regularized-Leader Online Learner

Summary for Today

1. The DAgger algorithm

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

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

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**: $\mathscr{D} = \mathscr{D} + \mathscr{D}^t$
3. **Update policy via Supervised-Learning**: $\pi^{t+1} = \mathrm{SL}\left(\mathscr{D}\right)$

2. We can see that DAgger is essentially an online-learning algorithm (FTRL)