Imitation Learning & Behavioral Cloning

Lucas Janson and Sham Kakade

CS/Stat 184: Introduction to Reinforcement Learning

Fall 2023

Today

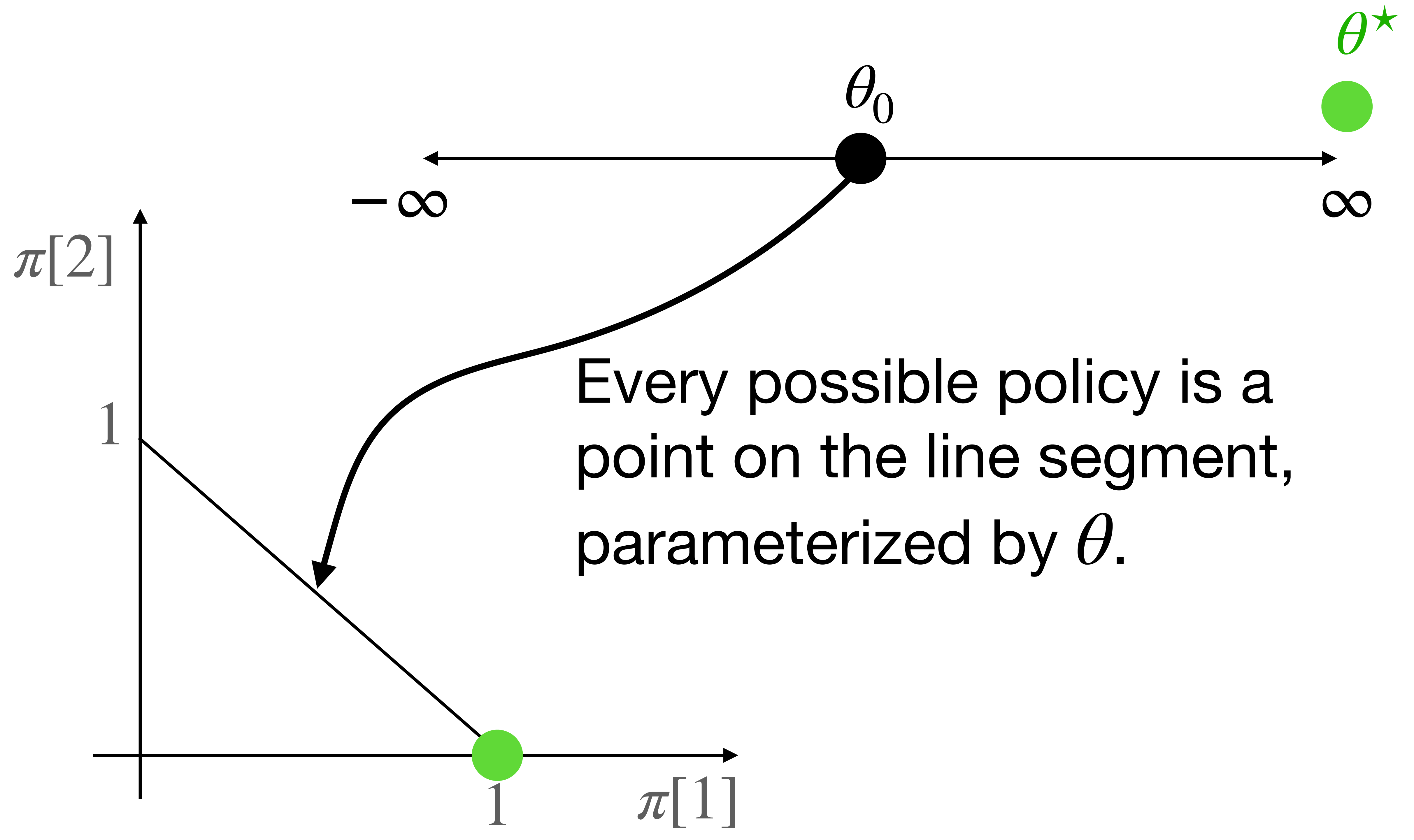
- Recap++
- Imitation Learning:
  - Behavioral Cloning
  - DAgger

Recap

Example of Natural Gradient on 1-d problem: 2 actions, 1 state

$$(\pi_\theta[1], \pi_\theta[2]) := \left(\frac{\exp(\theta)}{1+\exp(\theta)}, \frac{1}{1+\exp(\theta)} \right)$$

$$J(\theta) = 100 \cdot \pi_\theta[1] + 1 \cdot \pi_\theta[2]$$



Gradient: $J'(\theta) = \dfrac{99\exp(\theta)}{(1+\exp(\theta))^2}$

Exact PG: $\theta^{k+1} = \theta^k + \eta \dfrac{99\exp(\theta^k)}{(1+\exp(\theta^k))^2}$

i.e., vanilla GA moves to $\theta = \infty$ with smaller and smaller steps, since $J'(\theta) \to 0$ as $\theta \to \infty$

Fisher information scalar: $F_\theta = \dfrac{\exp(\theta)}{(1+\exp(\theta))^2}$

NPG: $\theta^{k+1} = \theta^k + \eta \dfrac{J'(\theta^k)}{F_{\theta^k}} = \theta_k + \eta \cdot 99$

NPG moves to $\theta = \infty$ much more quickly (for a fixed η)

Meta-Approach: CPI/TRPO/NPG/PPO are all pretty similar.

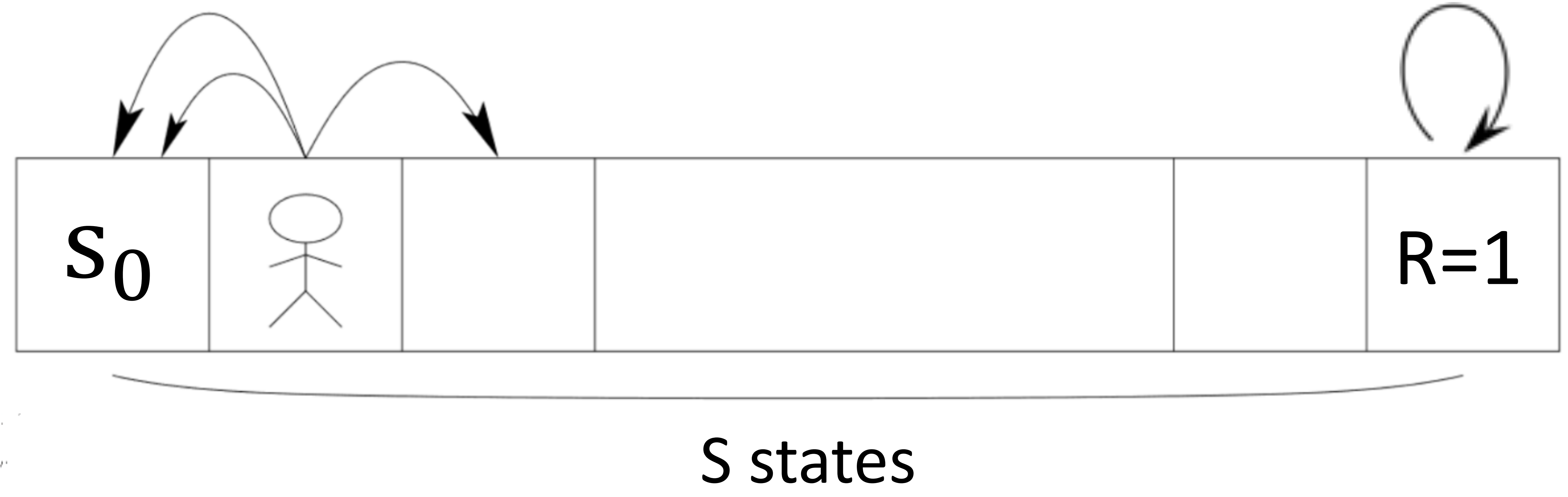
1. Init π_0
2. For $k = 0,\ldots K$:

   $\pi^{k+1} \approx \arg\max_\theta \Delta_k(\pi^\theta)$, where $\Delta_k(\pi) = \mathbb{E}_{s_0,\ldots s_{H-1}\sim\rho_{\pi^k}}\left[\sum_{h=0}^{H-1} \mathbb{E}_{a_h\sim\pi(s_h)} A^{\pi^k}(s_h, a_h)\right]$

   such that ρ_θ is "close" to ρ_{θ^k}

   - CPI: conservative policy iteration
     uses unconstrained optimization: $\tilde{\pi} \approx \arg\max_\theta \Delta_k(\pi^\theta)$,
     enforces closeness with "mixing": $\pi^{k+1} = (1-\alpha)\cdot\pi^k + \alpha\cdot\tilde{\pi}^{k+1}$
   - TRPO: use KL to enforce closeness.
   - NPG: is TRPO up to "leading order" (via Taylor's theorem).
   - PPO: uses a Lagrangian relaxation (i.e. regularization)
3. Return π_K

"Lack of Exploration" leads to Optimization and Statistical Challenges



Thrun '92

- Suppose $H \approx \text{poly}(|S|)$ & $\mu(s_0) = 1$ (i.e. we start at s_0).
- A randomly initialized policy π^0 has prob. $O(1/3^{|S|})$ of hitting the goal state in a trajectory.
- Implications:
  - The following sample based approach, with $\mu(s_0) = 1$, require $O(3^{|S|})$ trajectories.
    - Holds for (sample based) Fitted DP
    - Holds for (sample based) PG/CPI/TRPO/NPG/PPO
- Basically, for these approaches, we are stuck without exploration, if $\mu(s_0) = 1$.

Let's examine the role of μ

Thrun '92

- Suppose that somehow the distribution μ had better coverage.
  - e.g, μ was uniform over the all states in our toy problem, then all approaches we covered would work (with mild assumptions)
  - Theory: CPI/TRPO/NPG/PPO have better guarantees than fitted DP methods (assuming some "coverage")
- Strategies:
  - If we have a simulator, sometimes we can design μ to have better coverage.
    - this is helpful for robustness as well.
  - Imitation learning
    - An expert gives us samples from a "good" μ.
  - Explicit exploration:
    - UCB-VI: we'll merge two good ideas!
    - Encourage exploration in PG methods.
  - Try with reward shaping

Today:

What about guarantees for PG methods? (vs fitted-DP methods)

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) “supervised learning” worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) “supervised learning” worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Issues: let’s consider log-linear policies.

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) “supervised learning” worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Issues: let’s consider log-linear policies.

- Approximation error: For log linear policies, how good does ϕ need to be?
  (comment: hopefully some average case condition for approximating $A^\pi(s,a)$)

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) "supervised learning" worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Issues: let's consider log-linear policies.

- Approximation error: For log linear policies, how good does ϕ need to be?
  (comment: hopefully some average case condition for approximating $A^\pi(s,a)$)
- Sample size: hope to use a # samples that is poly in dim(ϕ) & $1/\epsilon_{accuracy}$.

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) "supervised learning" worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Issues: let's consider log-linear policies.

- Approximation error: For log linear policies, how good does ϕ need to be?
  (comment: hopefully some average case condition for approximating $A^\pi(s,a)$)
- Sample size: hope to use a # samples that is poly in dim(ϕ) & $1/\epsilon_{accuracy}$.
- Coverage: need some coverage condition over the state space.
  (comment: hopefully the coverage conditions are only in "ϕ-space")

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) “supervised learning” worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Issues: let’s consider log-linear policies.

- Approximation error: For log linear policies, how good does ϕ need to be?
  (comment: hopefully some average case condition for approximating $A^\pi(s,a)$)
- Sample size: hope to use a # samples that is poly in dim(ϕ) & $1/\epsilon_{accuracy}$.
- Coverage: need some coverage condition over the state space.
  (comment: hopefully the coverage conditions are only in “ϕ-space”)
- Computation: we want NPG to find something good with poly in $d, 1/\epsilon_{accuracy}, H$ iterations.

What about guarantees for PG methods? (vs fitted-DP methods)

- The hope is that if (average case) “supervised learning” worked, then RL would also work.

$$\pi_\theta(a \mid s) = \frac{\exp(\theta^\top \phi(s,a))}{\sum_{a'} \exp(\theta^\top \phi(s,a'))}$$

Issues: let’s consider log-linear policies.

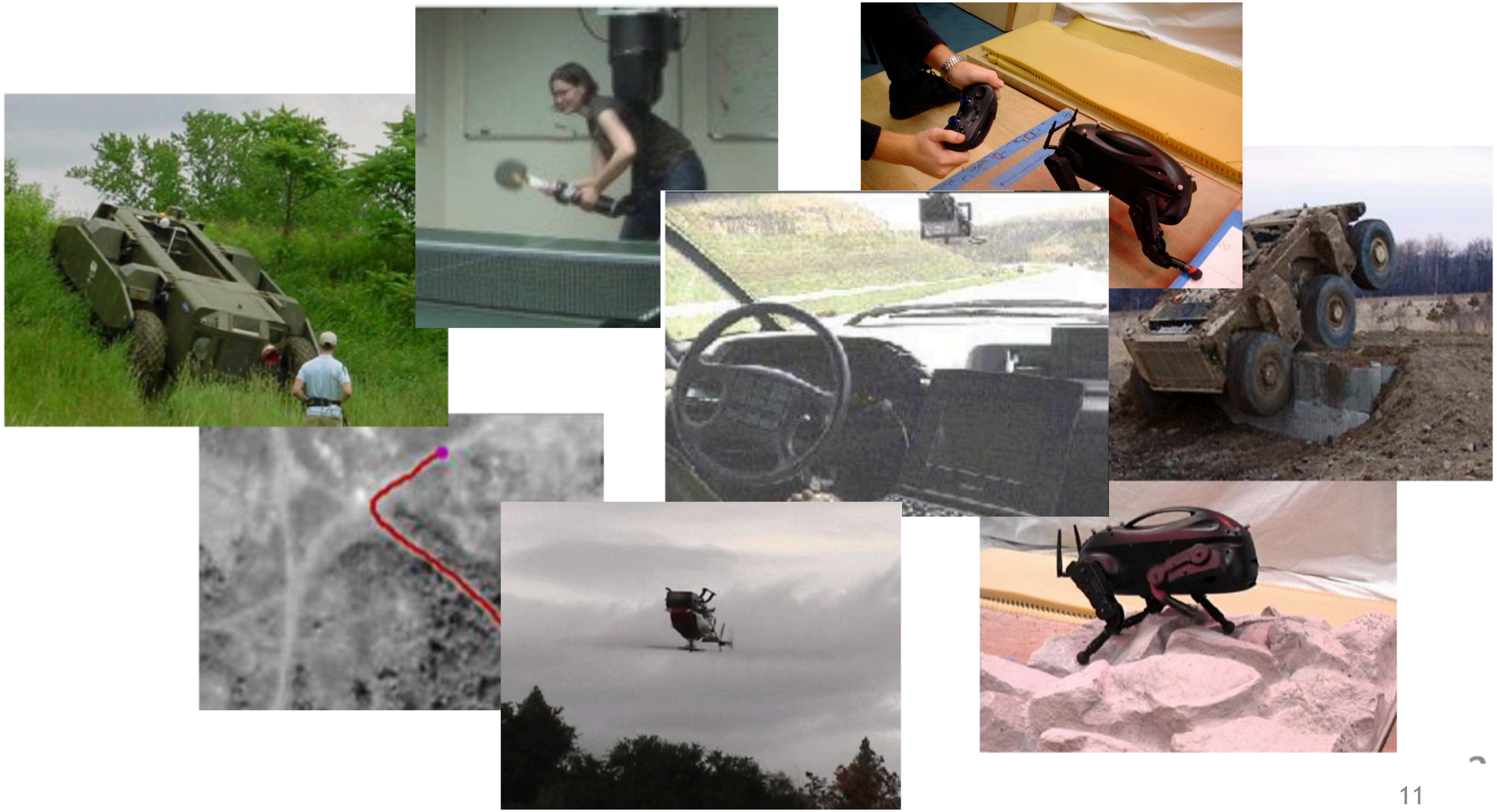
- Approximation error: For log linear policies, how good does ϕ need to be?
  (comment: hopefully some average case condition for approximating $A^\pi(s,a)$)
- Sample size: hope to use a # samples that is poly in dim(ϕ) & $1/\epsilon_{accuracy}$.
- Coverage: need some coverage condition over the state space.
  (comment: hopefully the coverage conditions are only in “ϕ-space”)
- Computation: we want NPG to find something good with poly in $d, 1/\epsilon_{accuracy}, H$ iterations.

- **Theory**: (see AJKS Ch 4+13, for formal log linear policies)
  There are (somewhat subtle) approx/coverage conditions where NPG converges to an $\epsilon_{accuracy}$-opt policy with poly sample, poly computation time.
  (Conditions are weaker than those for fitted-DP methods).

Today

- Recap++
- Imitation Learning:
  - Behavioral Cloning
  - DAgger

Imitation Learning



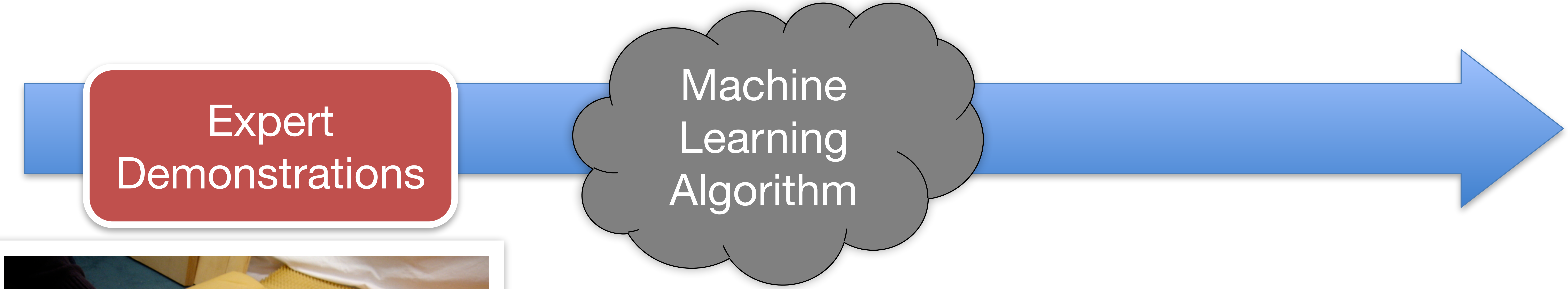
Imitation Learning

Imitation Learning

Imitation Learning

Expert Demonstrations

Imitation Learning



- SVM
- Gaussian Process
- Kernel Estimator
- Deep Networks
- Random Forests
- LWR
- …

Imitation Learning

Expert Demonstrations → Machine Learning Algorithm → Policy π

- SVM
- Gaussian Process
- Kernel Estimator
- Deep Networks
- Random Forests
- LWR
- …

Maps *states* to actions

Learning to Drive by Imitation

[Pomerleau89, Saxena05, Ross11a]

Input:

Camera Image

Policy

Output:

Steering Angle
in [-1, 1]

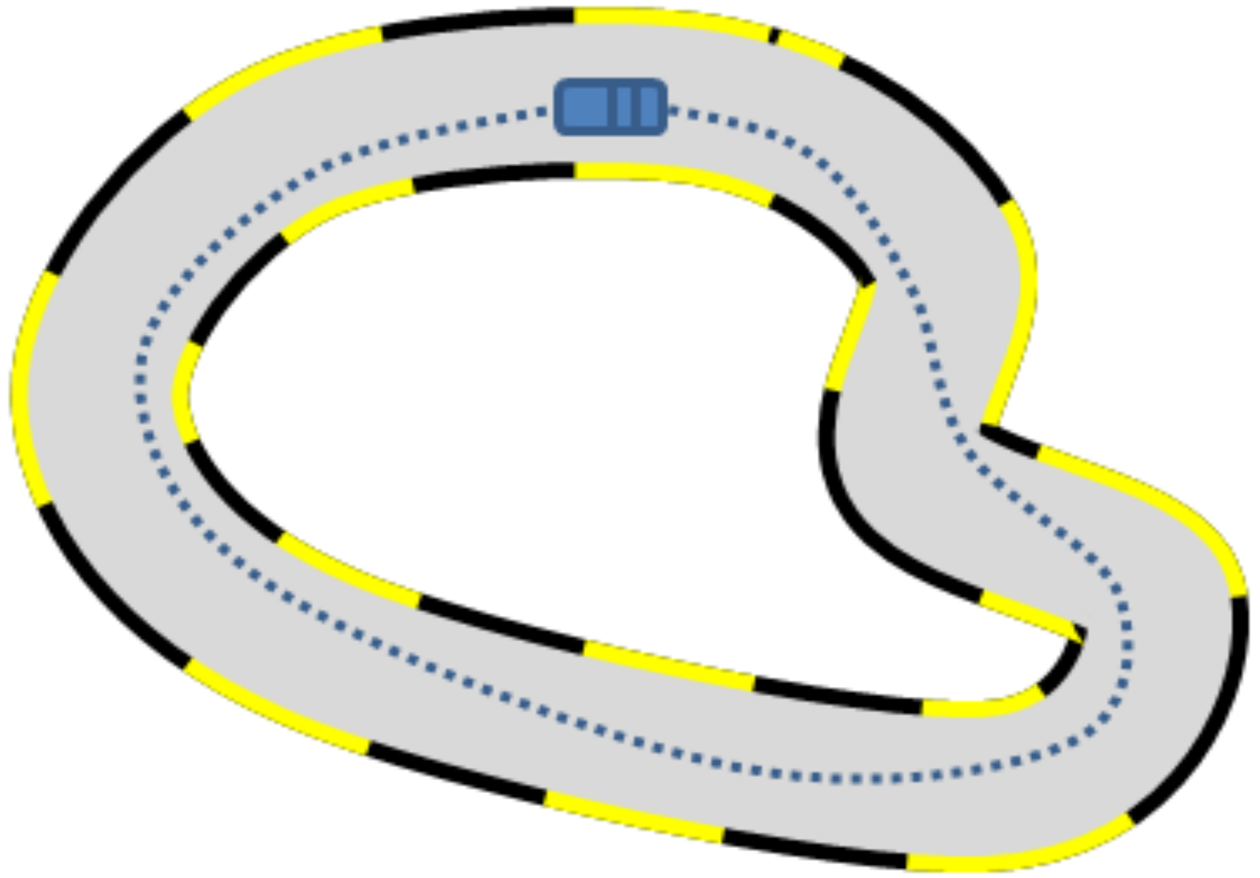
Today

- Recap++
- Imitation Learning:
  - ✓ Behavioral Cloning
  - DAgger

[Widrow64,Pomerleau89]

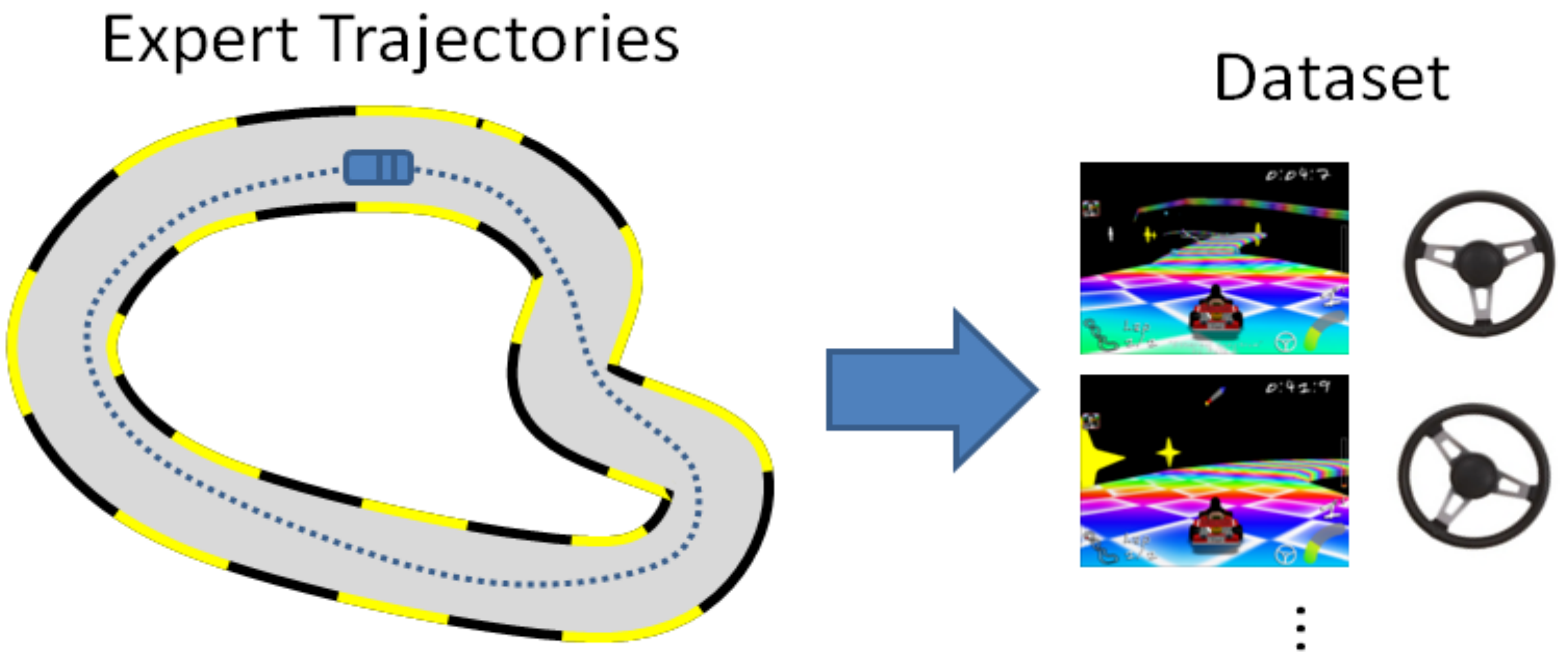
Supervised Learning Approach: Behavior Cloning

Expert Trajectories



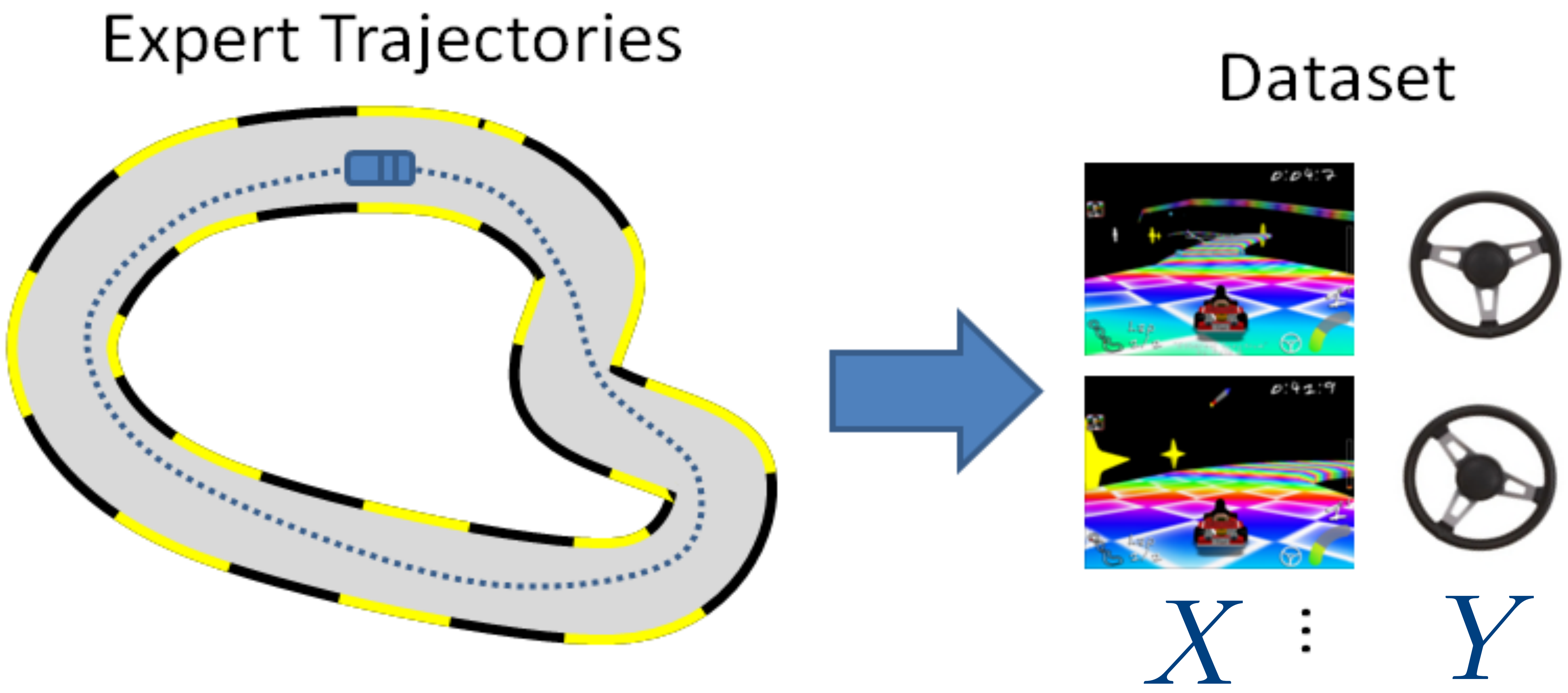
[Widrow64,Pomerleau89]

Supervised Learning Approach: Behavior Cloning



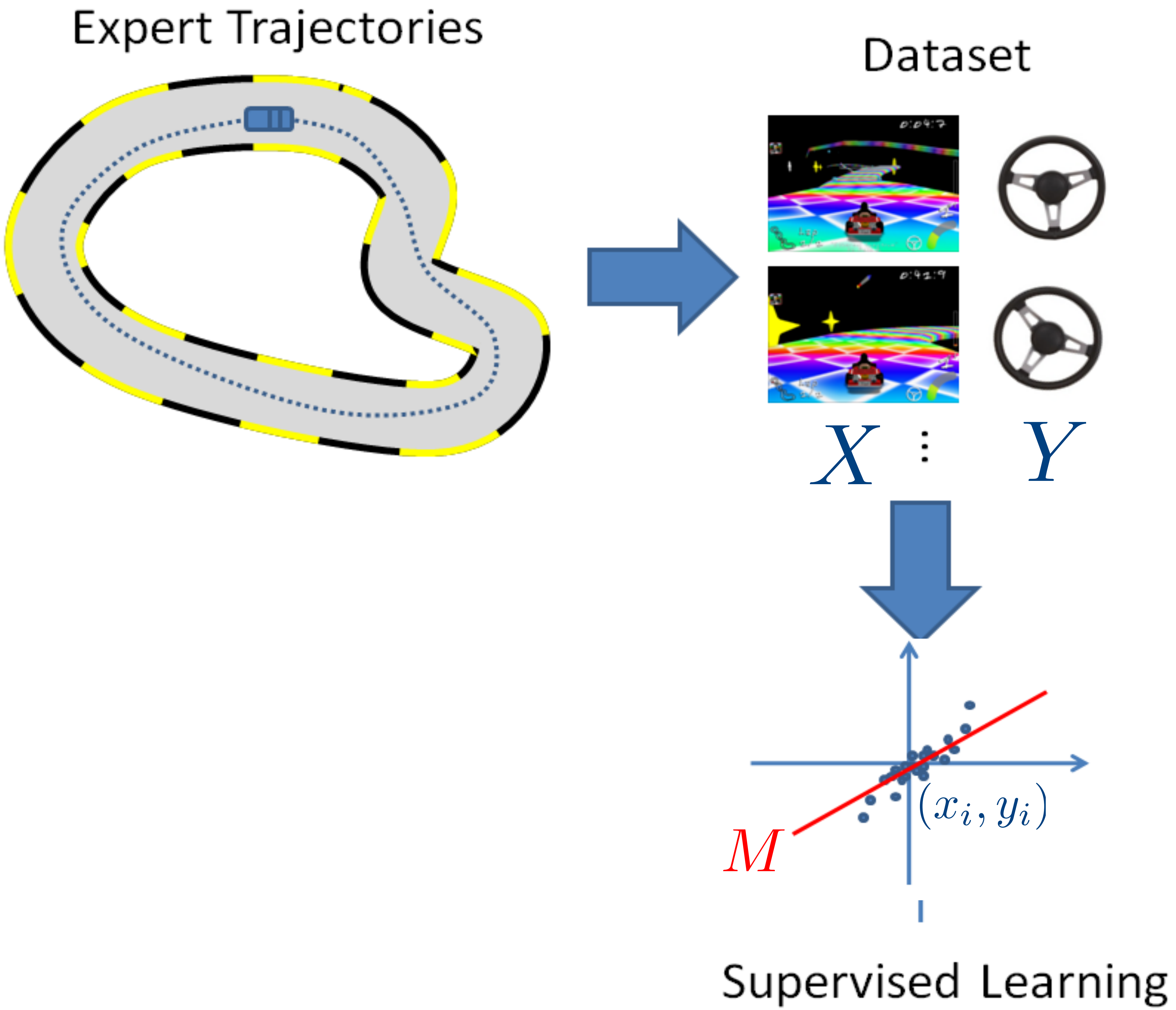
[Widrow64,Pomerleau89]

Supervised Learning Approach: Behavior Cloning



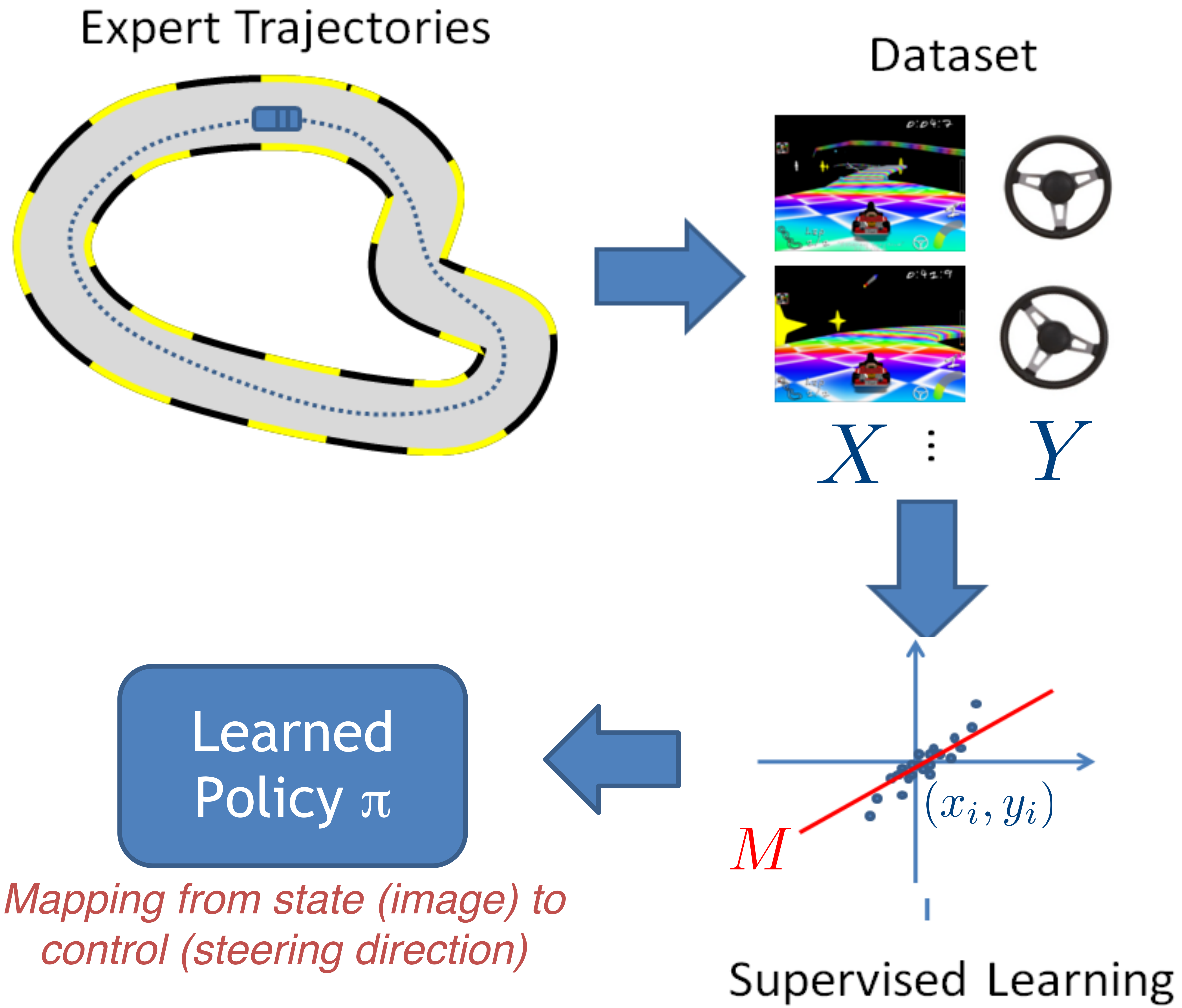
[Widrow64,Pomerleau89]

Supervised Learning Approach: Behavior Cloning



[Widrow64,Pomerleau89]

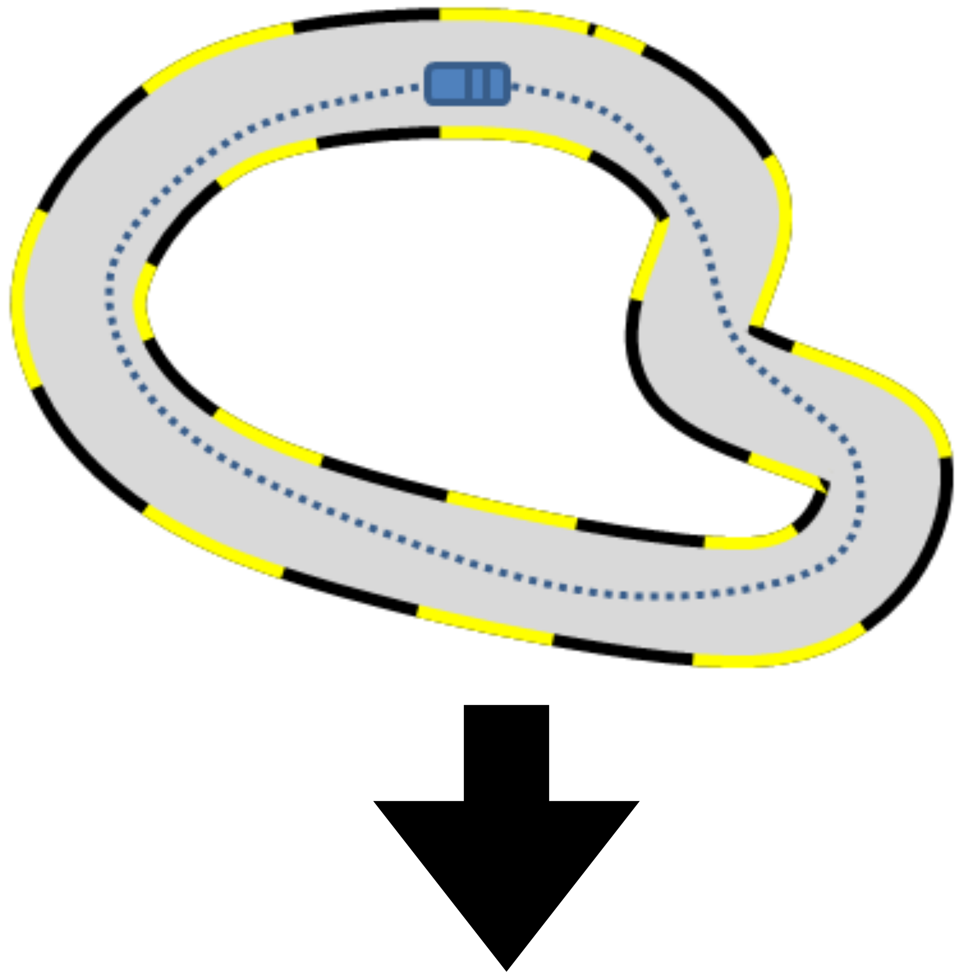
Supervised Learning Approach: Behavior Cloning



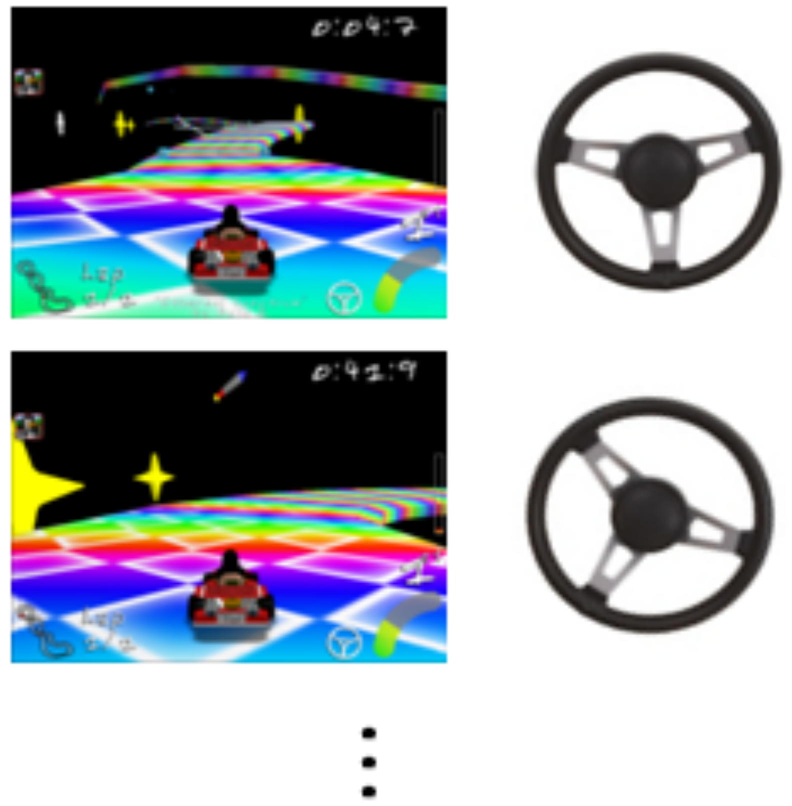
Mapping from state (image) to control (steering direction)

Let's formalize the offline IL Setting and the Behavior Cloning algorithm

Expert Trajectories

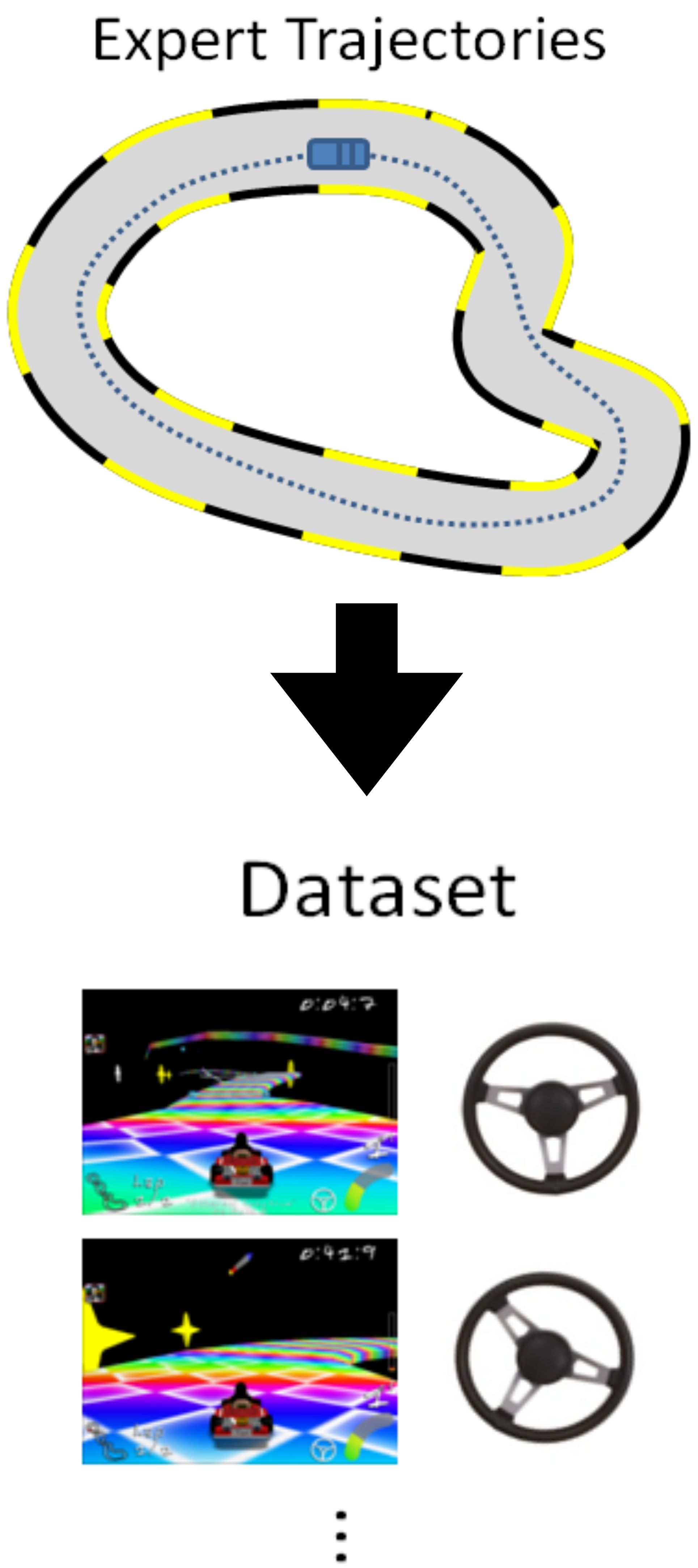


Dataset



Finite horizon MDP $\mathscr{M}$

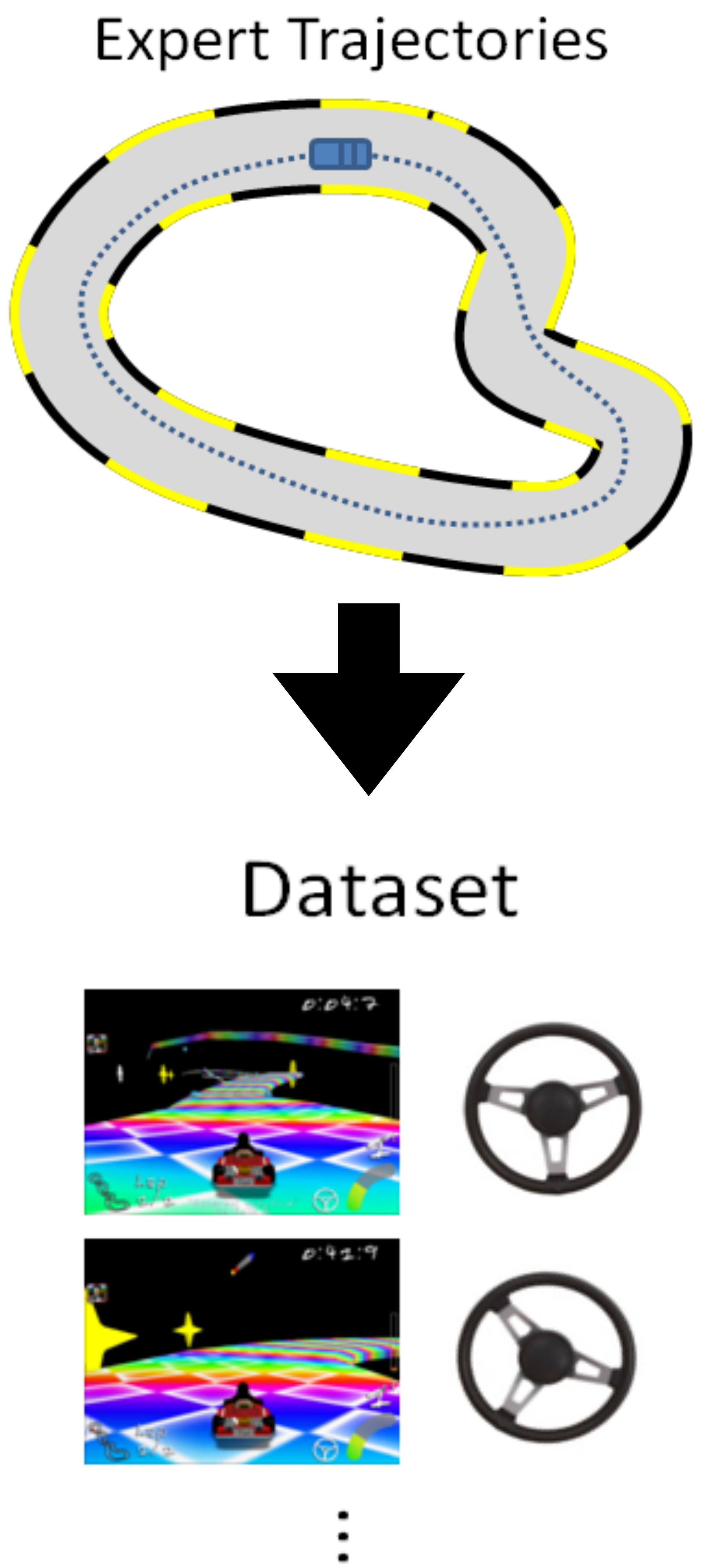
Let's formalize the offline IL Setting and the Behavior Cloning algorithm



Finite horizon MDP $\mathscr{M}$

Ground truth reward $r(s, a) \in [0,1]$ is unknown;
Assume the expert has a good policy $\pi^\star$ (not necessarily opt)

Let’s formalize the offline IL Setting and the Behavior Cloning algorithm

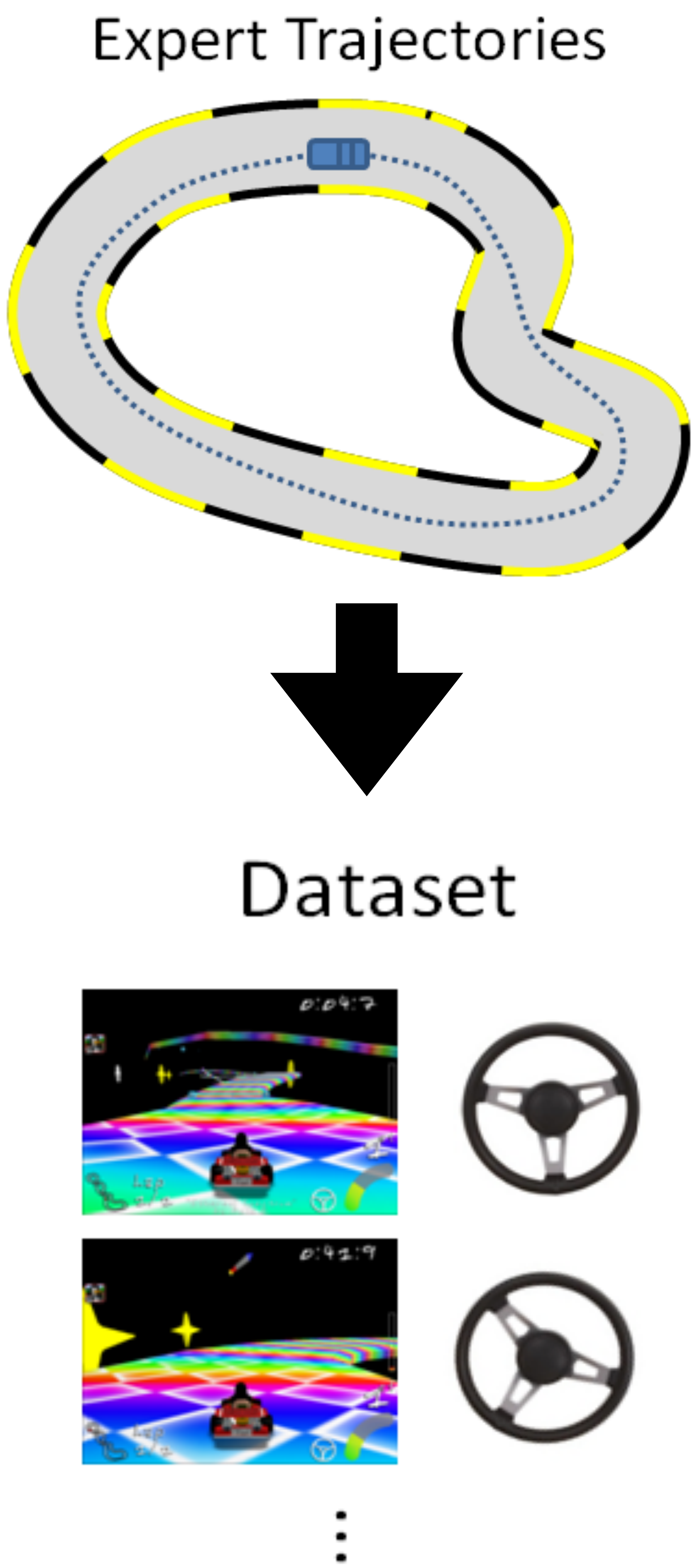


Finite horizon MDP $\mathscr{M}$

Ground truth reward $r(s, a) \in [0,1]$ is unknown;
Assume the expert has a good policy $\pi^\star$ (not necessarily opt)

We have a dataset of M trajectories: $\mathscr{D} = \{\tau_1, \ldots \tau_M\}$,
where $\tau_i = (s_h^i, a_h^i)_{h=0}^{H-1} \sim \rho_{\pi^\star}$

Let's formalize the offline IL Setting and the Behavior Cloning algorithm



Finite horizon MDP $\mathscr{M}$

Ground truth reward $r(s, a) \in [0,1]$ is unknown;
Assume the expert has a good policy $\pi^\star$ (not necessarily opt)

We have a dataset of M trajectories: $\mathscr{D} = \{\tau_1, \ldots \tau_M\}$,
where $\tau_i = (s_h^i, a_h^i)_{h=0}^{H-1} \sim \rho_{\pi^\star}$

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 $\Pi = \{\pi : S \mapsto \Delta(A)\}$

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:

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:

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

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:

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Many choices of loss functions:

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:

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Many choices of loss functions:

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

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:

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Many choices of loss functions:

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

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

Theorem: IL is (almost) as easy as SL

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Note a training and testing “mismatch”

Theorem: IL is (almost) as easy as SL

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Note a training and testing "mismatch"

Theorem [BC Performance]:

suppose we assume supervised learning succeeds, with ϵ classification error:

$$\mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\frac{1}{H} \sum_{h=0}^{H-1} \mathbf{1}\left[\widehat{\pi}(s_h) \neq \pi^\star(s_h)\right] \right] \leq \epsilon,$$

Theorem: IL is (almost) as easy as SL

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Note a training and testing "mismatch"

Theorem [BC Performance]:

suppose we assume supervised learning succeeds, with ϵ classification error:

$$\mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\frac{1}{H} \sum_{h=0}^{H-1} \mathbf{1}\left[\hat{\pi}(s_h) \neq \pi^\star(s_h) \right] \right] \leq \epsilon,$$

(where $\pi^\star$ is the experts policy, which need not be optimal)

Theorem: IL is (almost) as easy as SL

$$\hat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Note a training and testing "mismatch"

Theorem [BC Performance]:

suppose we assume supervised learning succeeds, with ϵ classification error:

$$\mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\frac{1}{H} \sum_{h=0}^{H-1} \mathbf{1}\left[\hat{\pi}(s_h) \neq \pi^\star(s_h)\right] \right] \leq \epsilon,$$

(where $\pi^\star$ is the experts policy, which need not be optimal)

then, under μ, we have:

$$|V^{\pi^\star} - V^{\hat{\pi}}| \leq H^2 \epsilon$$

Theorem: IL is (almost) as easy as SL

$$\widehat{\pi} = \arg\min_{\pi \in \Pi} \sum_{i=1}^{M} \sum_{h=0}^{H-1} \ell\left(\pi, s_h^i, a_h^i\right)$$

Note a training and testing "mismatch"

Theorem [BC Performance]:

suppose we assume supervised learning succeeds, with ϵ classification error:

$$\mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\frac{1}{H} \sum_{h=0}^{H-1} \mathbf{1} \left[\widehat{\pi}(s_h) \neq \pi^\star(s_h) \right] \right] \leq \epsilon,$$

(where $\pi^\star$ is the experts policy, which need not be optimal)

then, under μ, we have:

$$| V^{\pi^\star} - V^{\widehat{\pi}} | \leq H^2 \epsilon$$

The quadratic amplification is annoying

Proof:

Proof:

By the PDL

$$|V^{\pi^\star}(s) - V^{\widehat{\pi}}(s)| = \left| \mathbb{E}_{\tau \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\widehat{\pi}}(s_h, a_h) \right] \right|$$

Proof:

By the PDL

$$|V^{\pi^\star}(s) - V^{\widehat{\pi}}(s)| = \left| \mathbb{E}_{\tau \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\widehat{\pi}}(s_h, a_h) \right] \right|$$

$$= \left| \mathbb{E}_{s_1, \ldots s_h \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\widehat{\pi}}(s_h, \pi^\star(s_h)) \right] \right|$$

Proof:

By the PDL

$$|V^{\pi^\star}(s) - V^{\hat{\pi}}(s)| = \left| \mathbb{E}_{\tau \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\hat{\pi}}(s_h, a_h) \right] \right|$$

$$= \left| \mathbb{E}_{s_1, \ldots s_h \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\hat{\pi}}(s_h, \pi^\star(s_h)) \right] \right|$$

$$\leq H \left| \mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\sum_{h=0}^{H-1} \mathbf{1} \left[\hat{\pi}(s_h) \neq \pi^\star(s_h) \right] \right| \right]$$

Proof:

By the PDL

$$|V^{\pi^\star}(s) - V^{\widehat{\pi}}(s)| = \left| \mathbb{E}_{\tau \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\widehat{\pi}}(s_h, a_h) \right] \right|$$

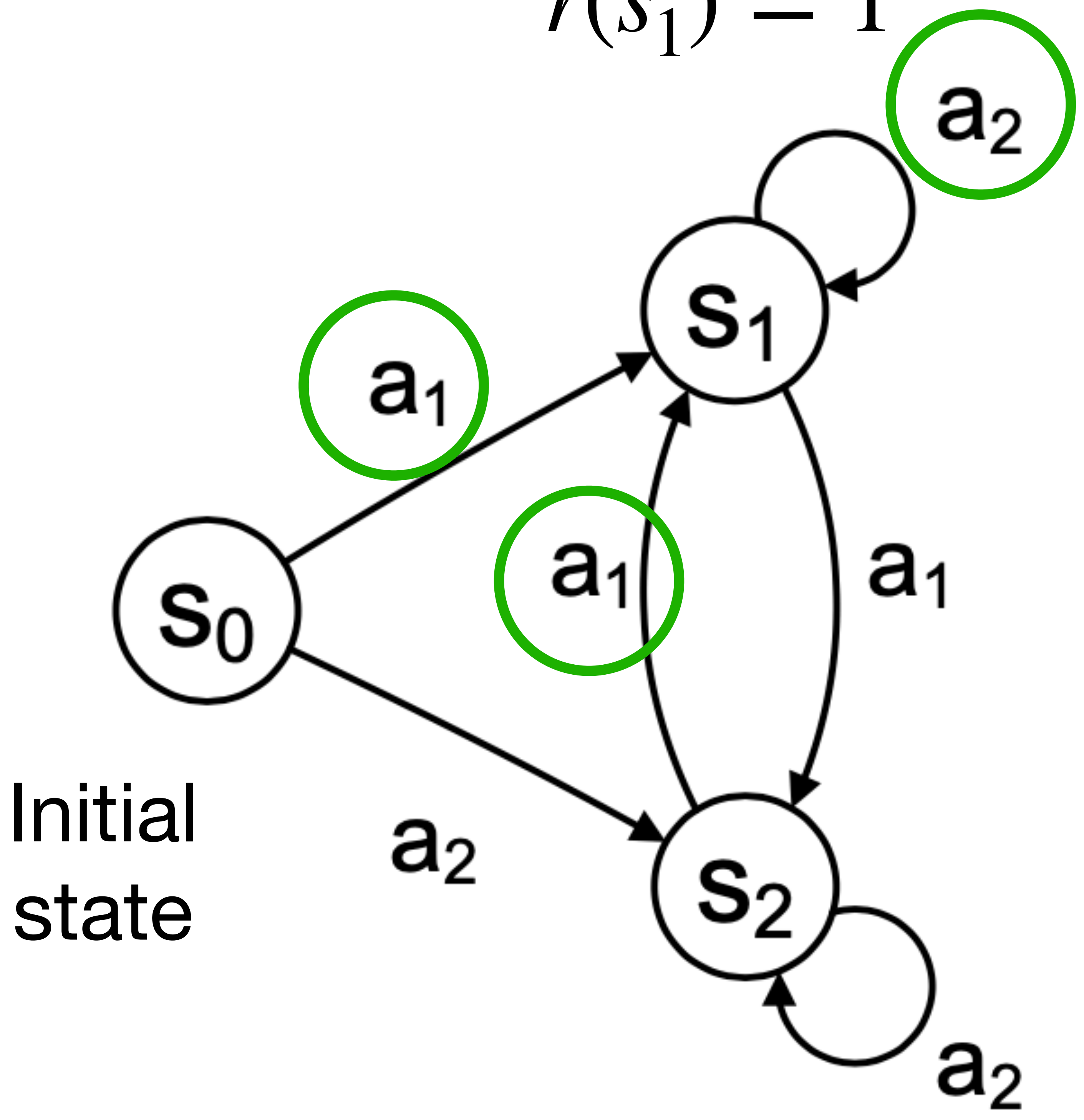
$$= \left| \mathbb{E}_{s_1, \ldots s_h \sim \rho_{\pi^\star}} \left[\sum_{h=0}^{H-1} A_h^{\widehat{\pi}}(s_h, \pi^\star(s_h)) \right] \right|$$

$$\leq H \left| \mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\sum_{h=0}^{H-1} \mathbf{1} \left[\widehat{\pi}(s_h) \neq \pi^\star(s_h) \right] \right] \right|$$

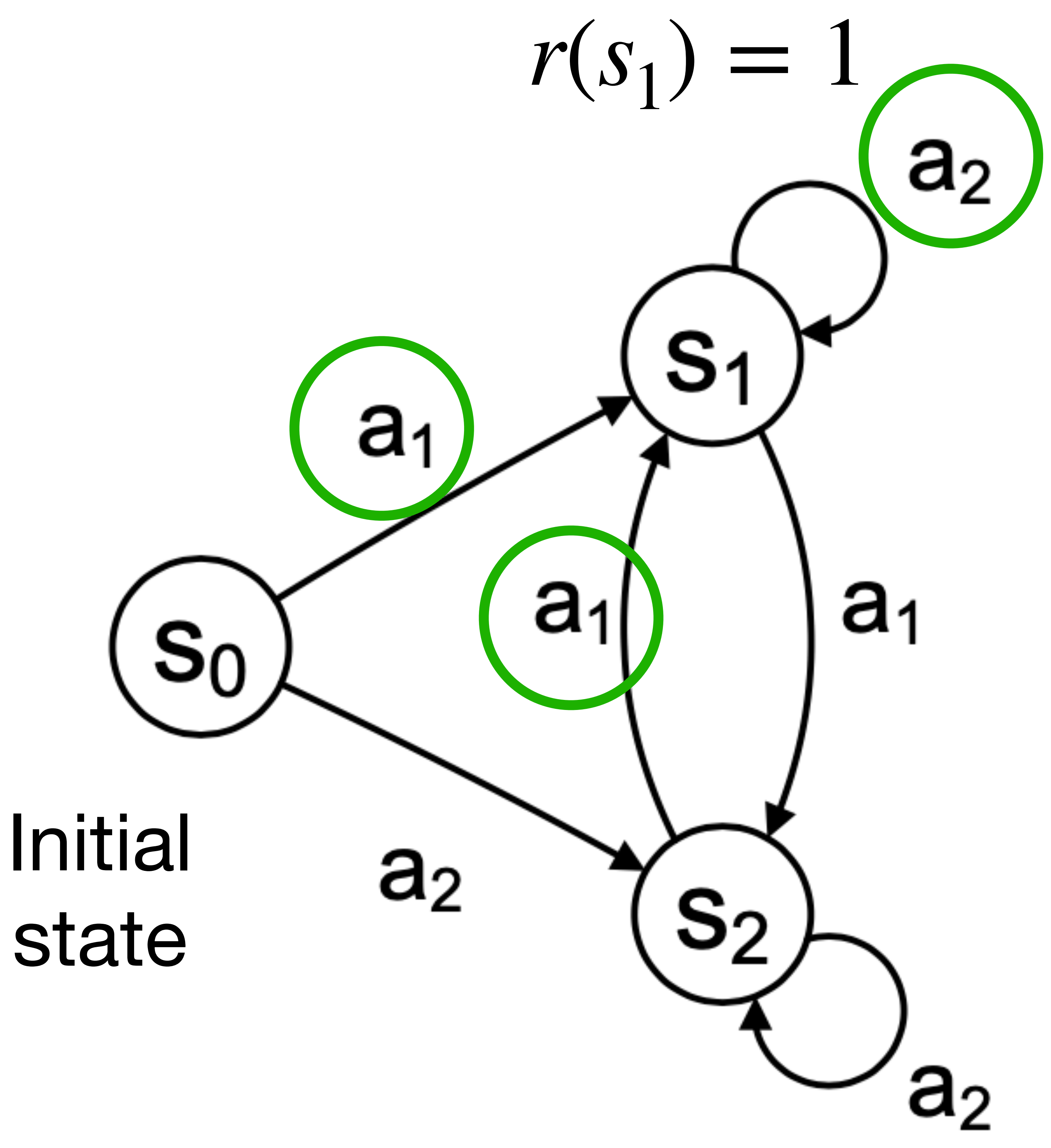
$$\leq H^2 \epsilon$$

Distribution Shift Example (H^2 factor is tight)

$r(s_1) = 1$

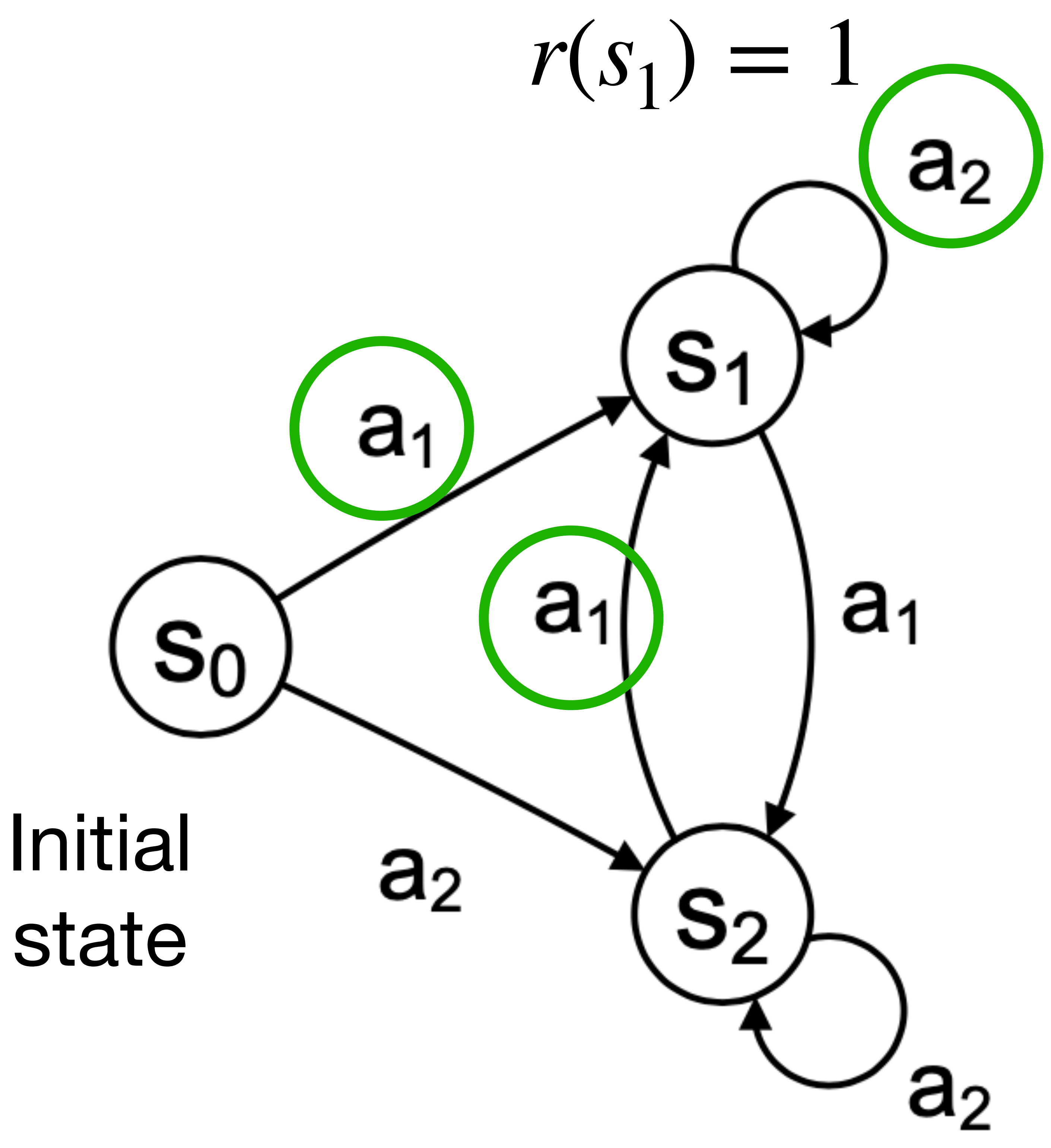


Distribution Shift Example (H^2 factor is tight)



Opt policy:

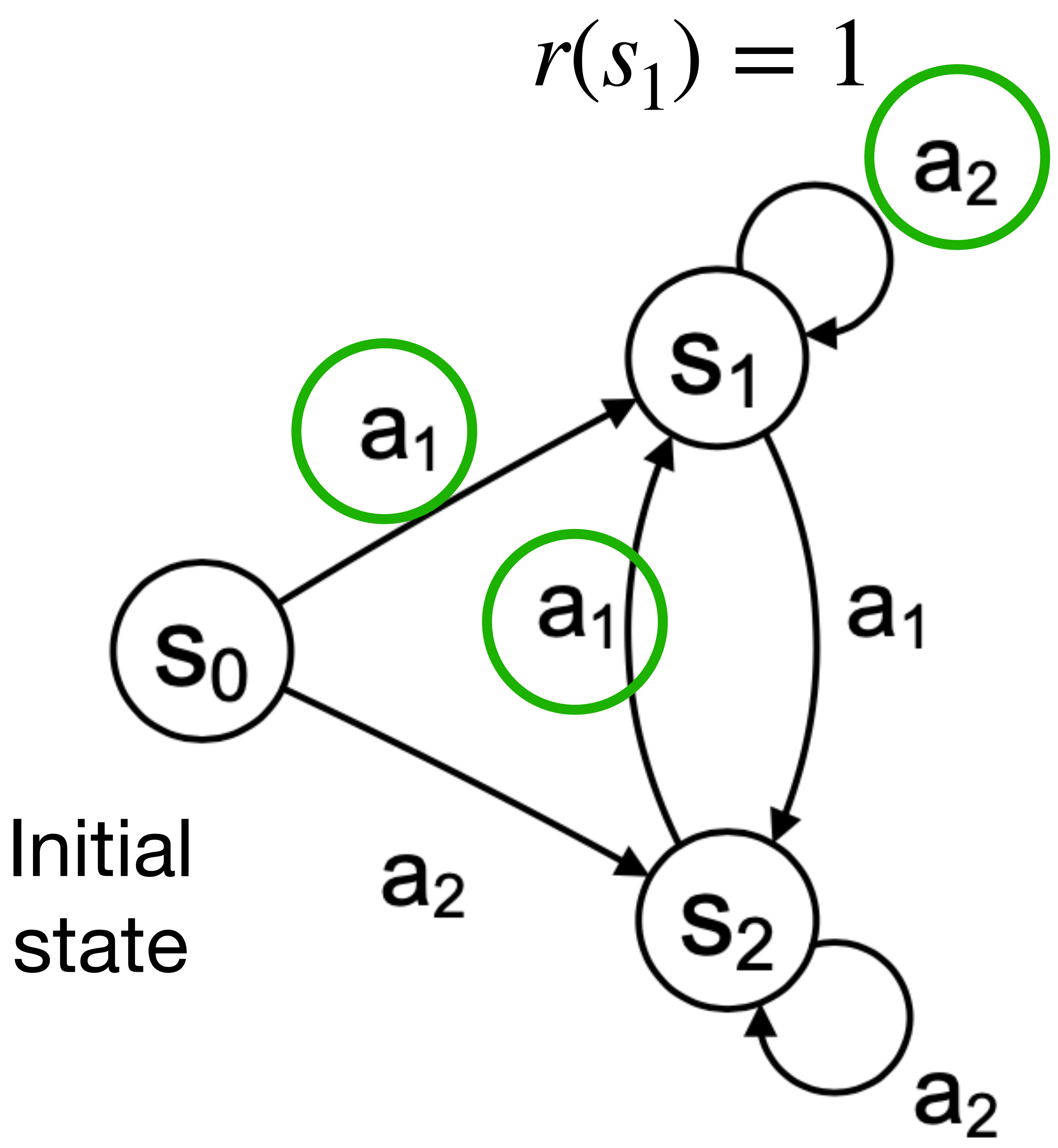
Distribution Shift Example (H^2 factor is tight)



Opt policy:

Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

Distribution Shift Example (H^2 factor is tight)

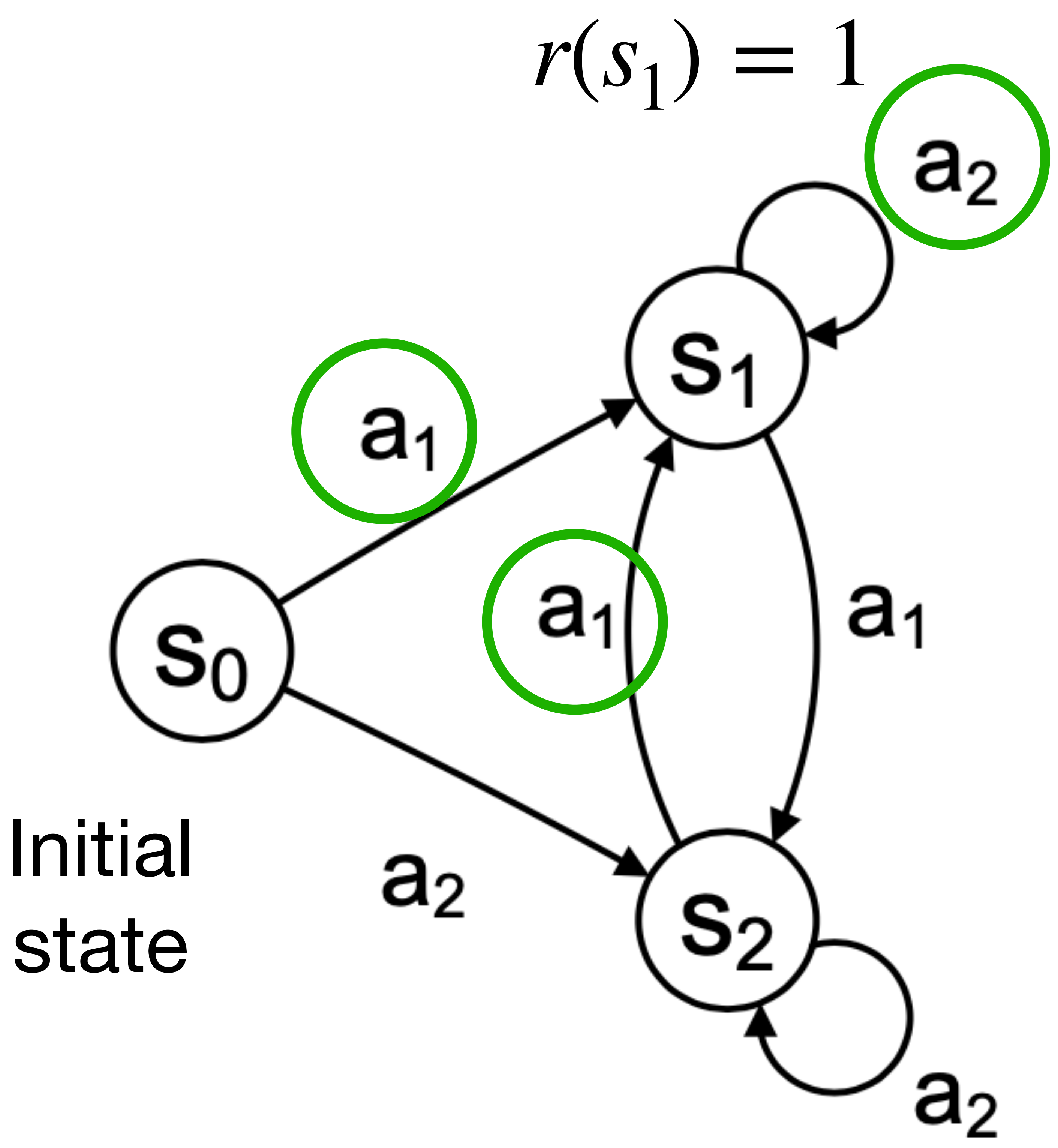


Opt policy:

Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

$\rho_{\pi^\star}(s_h = s_2) = 0$

Distribution Shift Example (H^2 factor is tight)



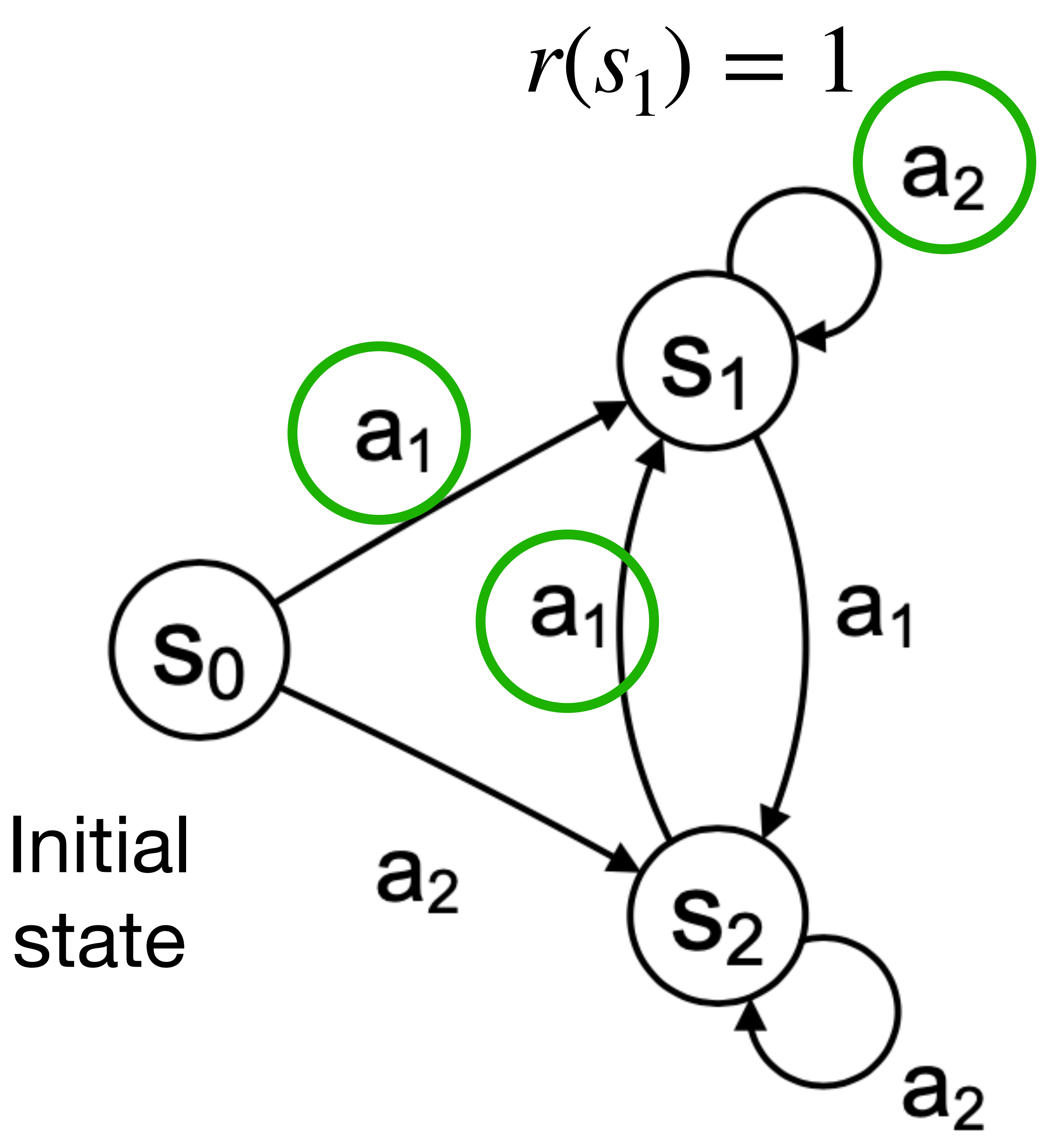
Opt policy:

Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

$\rho_{\pi^\star}(s_h = s_2) = 0$

$V_H^{\pi^\star}(s_0) = H - 1$

Distribution Shift Example (H^2 factor is tight)



Assume SL returns the policy $\widehat{\pi}$:

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

Opt policy:

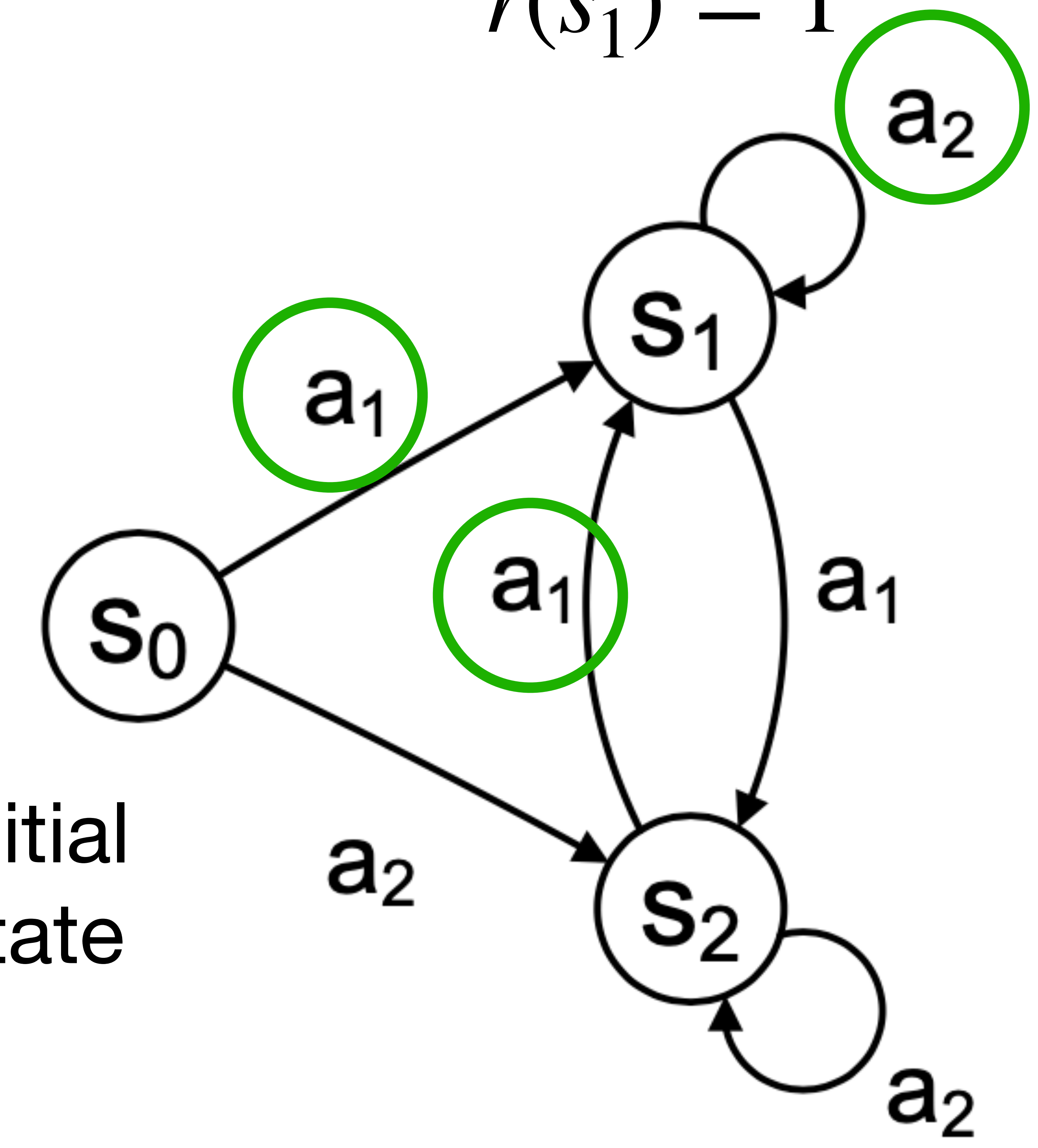
Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

$\rho_{\pi^\star}(s_h = s_2) = 0$

$V_H^{\pi^\star}(s_0) = H - 1$

Distribution Shift Example (H^2 factor is tight)

$r(s_1) = 1$



Assume SL returns the 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:

$$\mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}}\left[\frac{1}{H}\sum_{h=0}^{H-1} \mathbf{1}\left[\widehat{\pi}(s_h) \neq \pi^\star(s_h)\right]\right] = \epsilon$$

note: while $\widehat{\pi}(s_2) \neq \widehat{\pi}^\star(s_2)$, state s_2 is never visited under $\pi^\star$

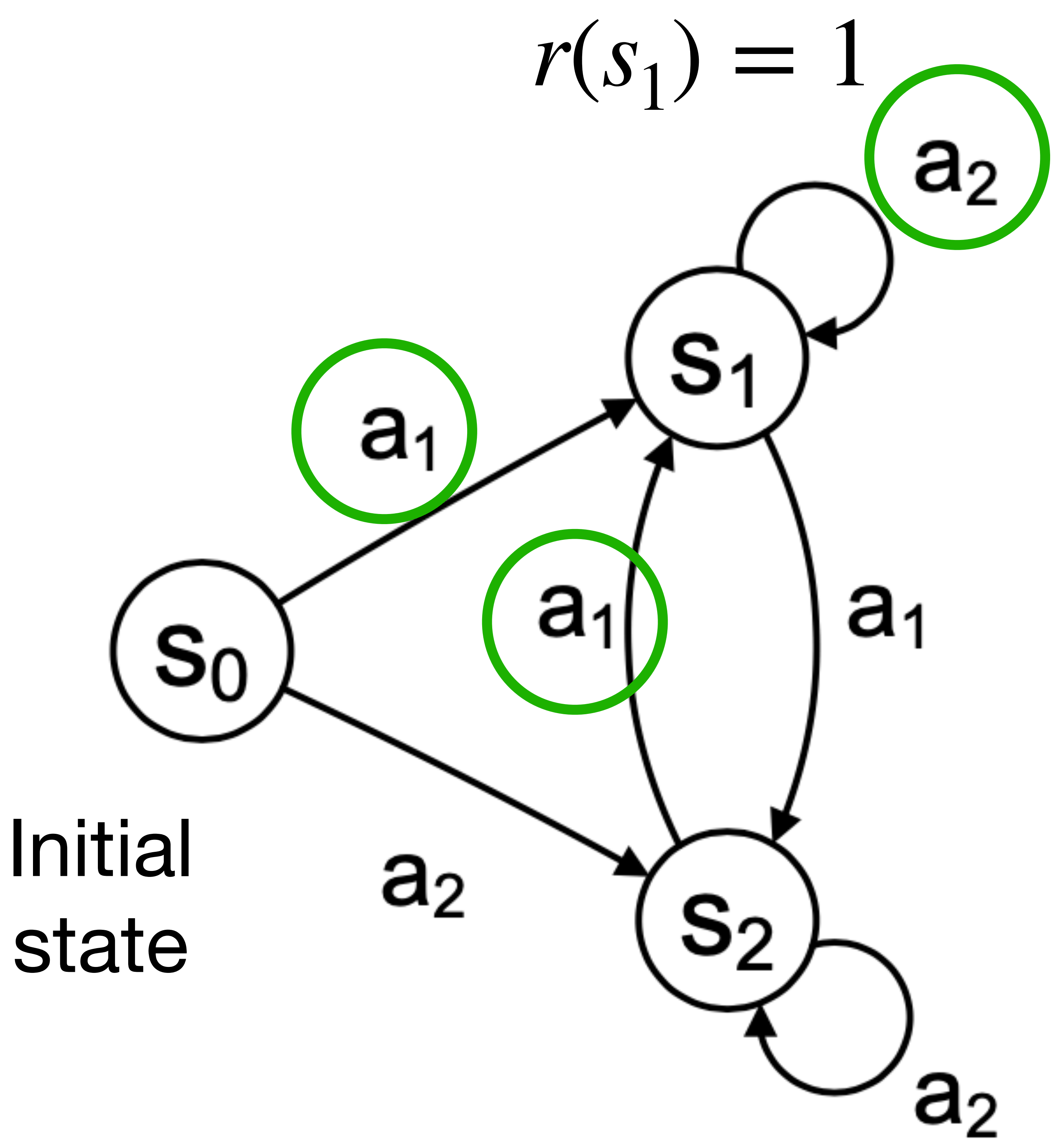
Opt policy:

Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

$\rho_{\pi^\star}(s_h = s_2) = 0$

$V_H^{\pi^\star}(s_0) = H - 1$

Distribution Shift Example (H^2 factor is tight)



Opt policy:

Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

$\rho_{\pi^\star}(s_h = s_2) = 0$

$V_H^{\pi^\star}(s_0) = H - 1$

Assume SL returns the 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:

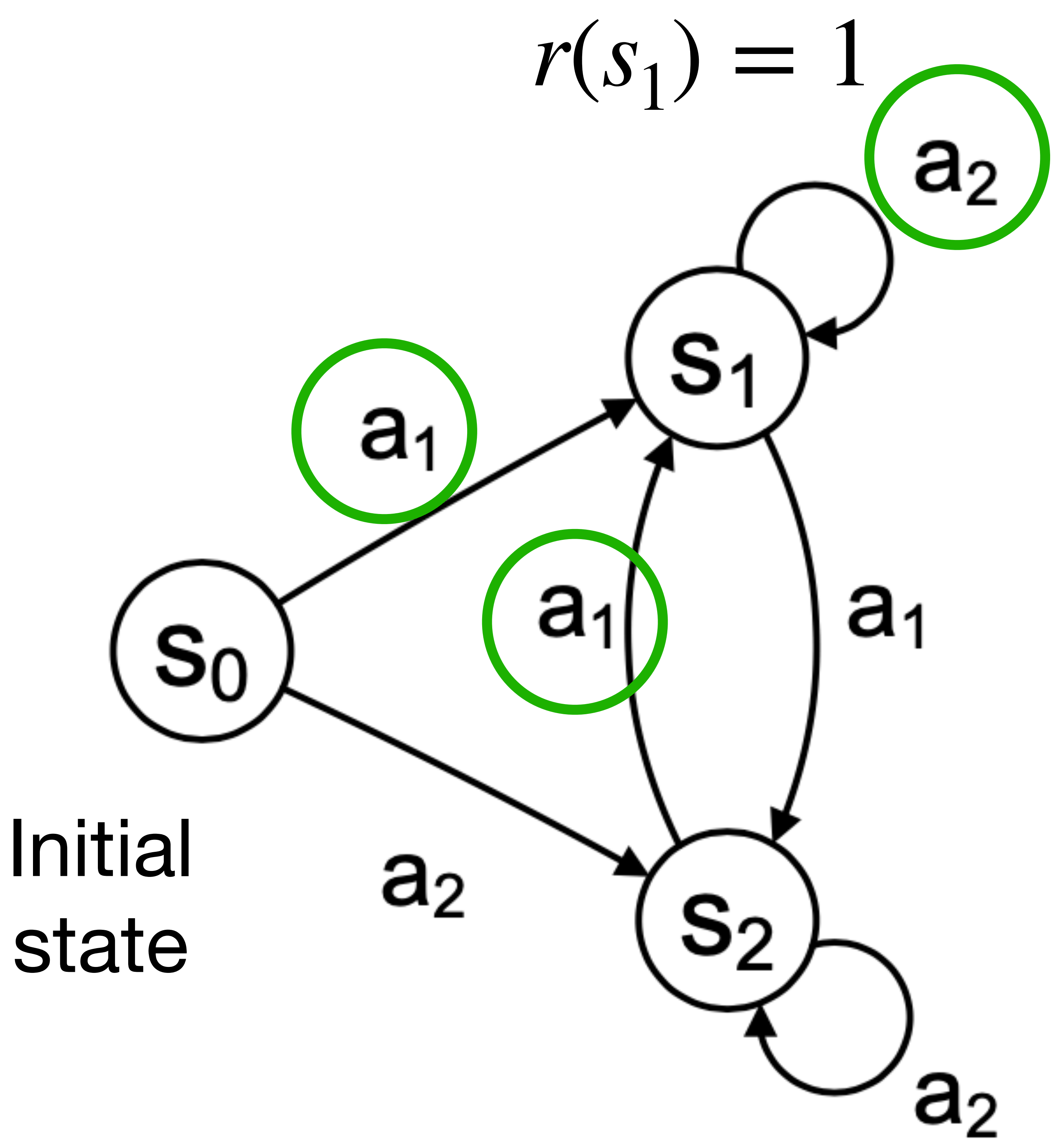
$$\mathbb{E}_{\tau \sim \rho_{\pi_{\pi^\star}}} \left[\frac{1}{H} \sum_{h=0}^{H-1} \mathbf{1}\left[\widehat{\pi}(s_h) \neq \pi^\star(s_h)\right] \right] = \epsilon$$

note: while $\widehat{\pi}(s_2) \neq \widehat{\pi}^\star(s_2)$, state s_2 is never visited under $\pi^\star$

We have quadratic degradation (in H):

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

Distribution Shift Example (H^2 factor is tight)



Opt policy:

Under $\rho_{\pi^\star}$, trajectory is $s_0, s_1, s_1, \ldots$

$\rho_{\pi^\star}(s_h = s_2) = 0$

$V_H^{\pi^\star}(s_0) = H - 1$

Assume SL returns the 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:

$$\mathbb{E}_{\tau\sim\rho_{\pi_{\pi^\star}}}\left[\frac{1}{H}\sum_{h=0}^{H-1}\mathbf{1}\left[\widehat{\pi}(s_h) \neq \pi^\star(s_h)\right]\right] = \epsilon$$

note: while $\widehat{\pi}(s_2) \neq \widehat{\pi}^\star(s_2)$, state s_2 is never visited under $\pi^\star$

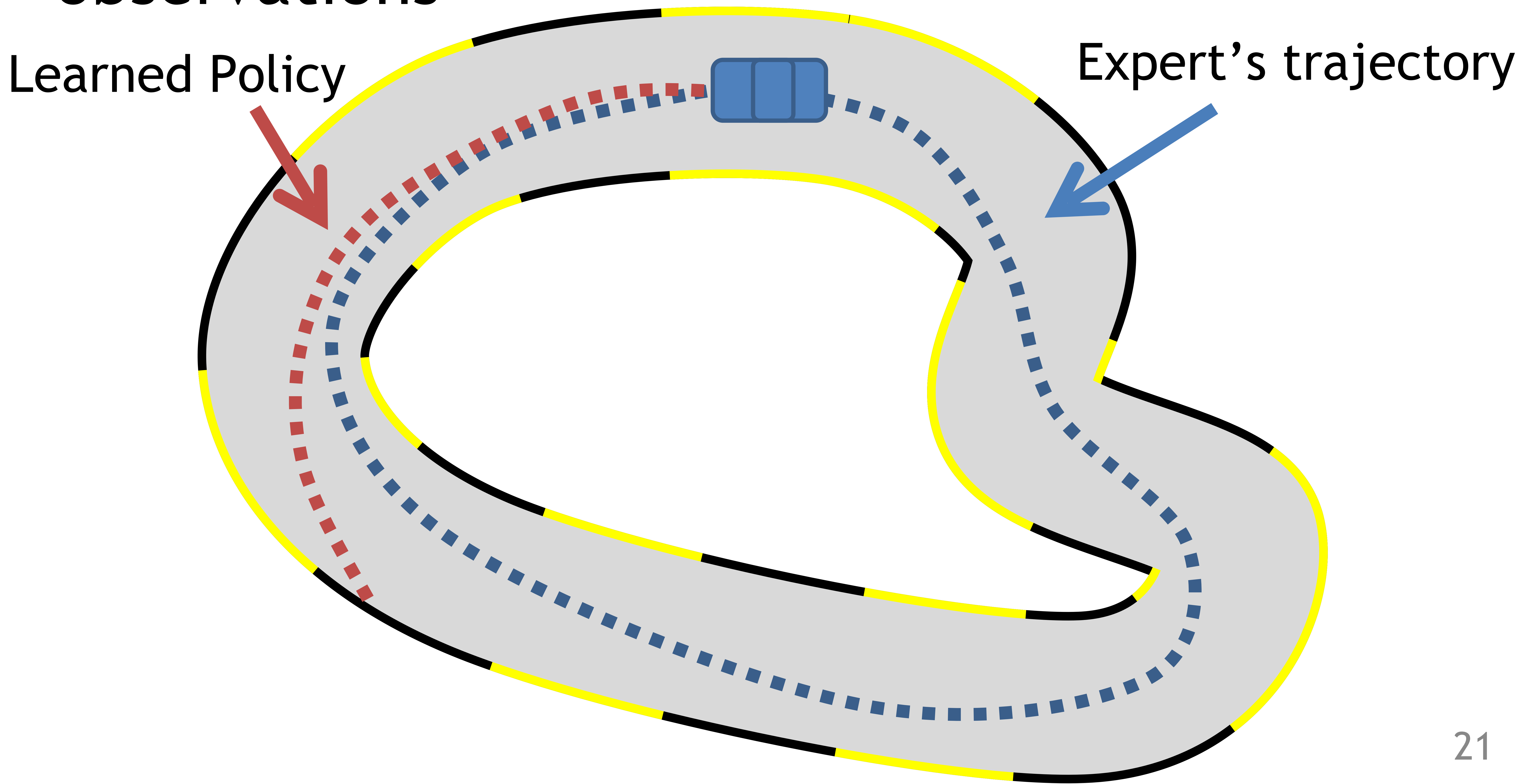
We have quadratic degradation (in H):

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

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

What could go wrong?

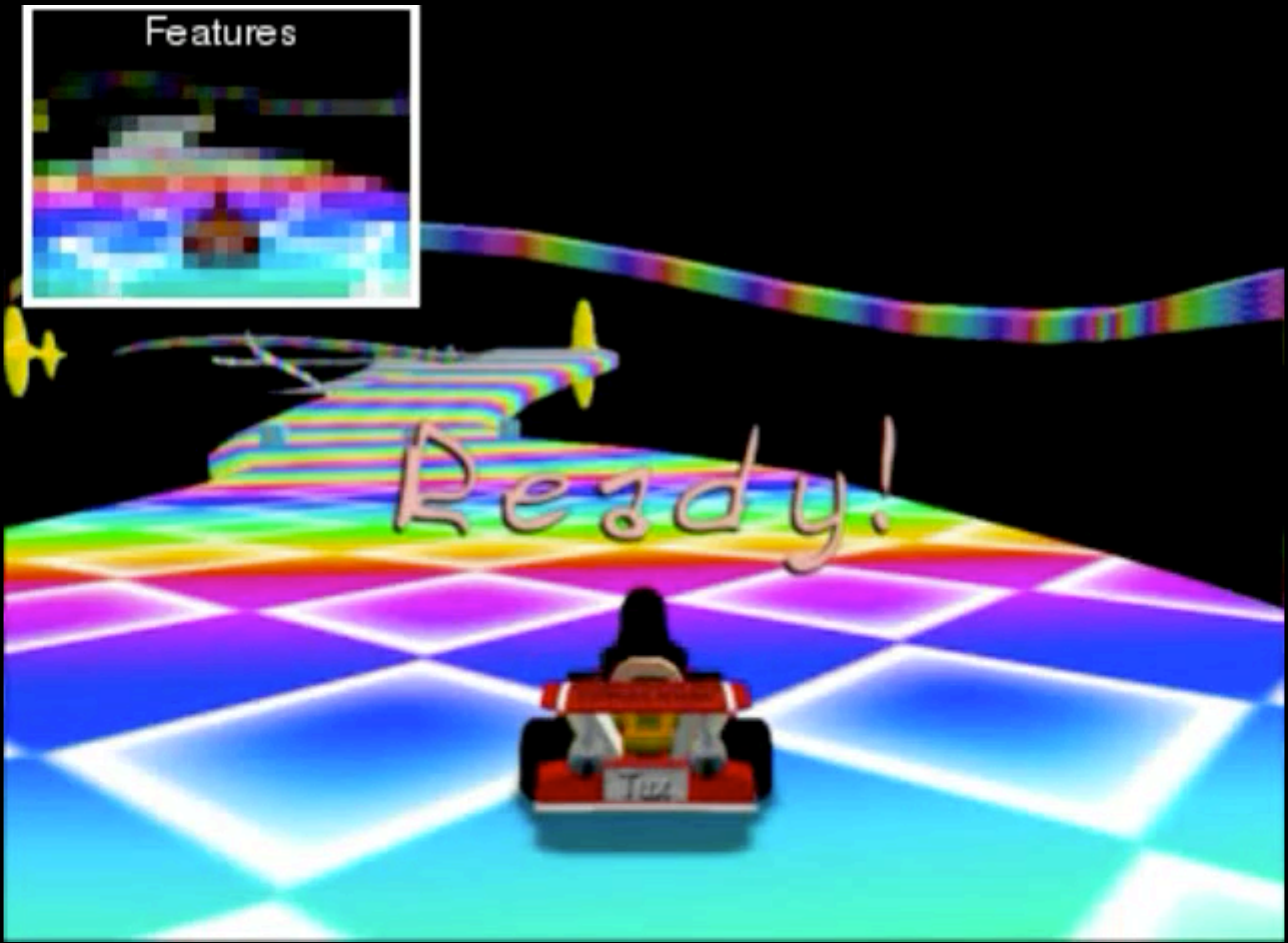
- Predictions affect future inputs/ observations

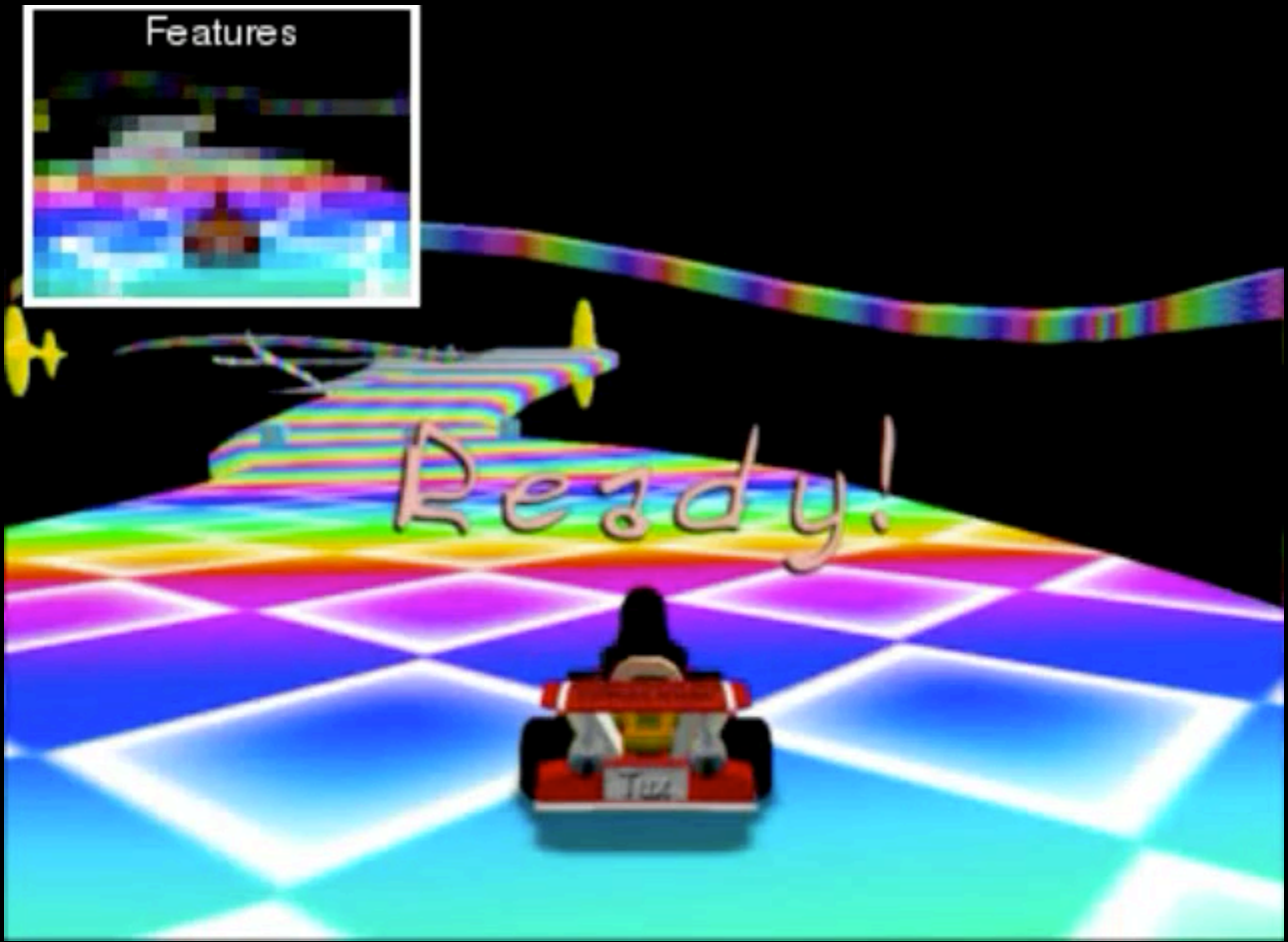


Expert Demos


Ready!


Ready!


Features
Ready!

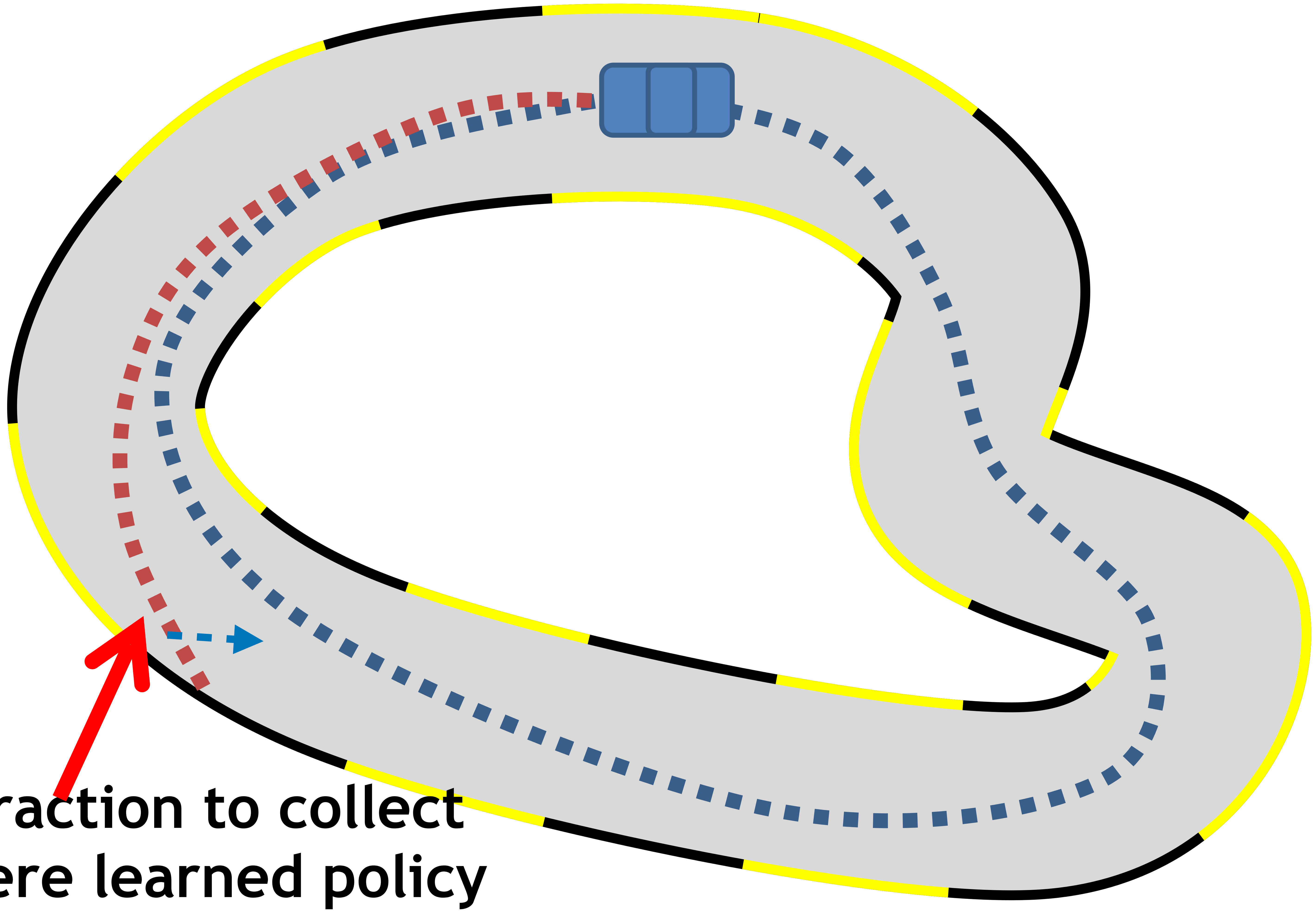

Features
Ready!

BC Policy

Today

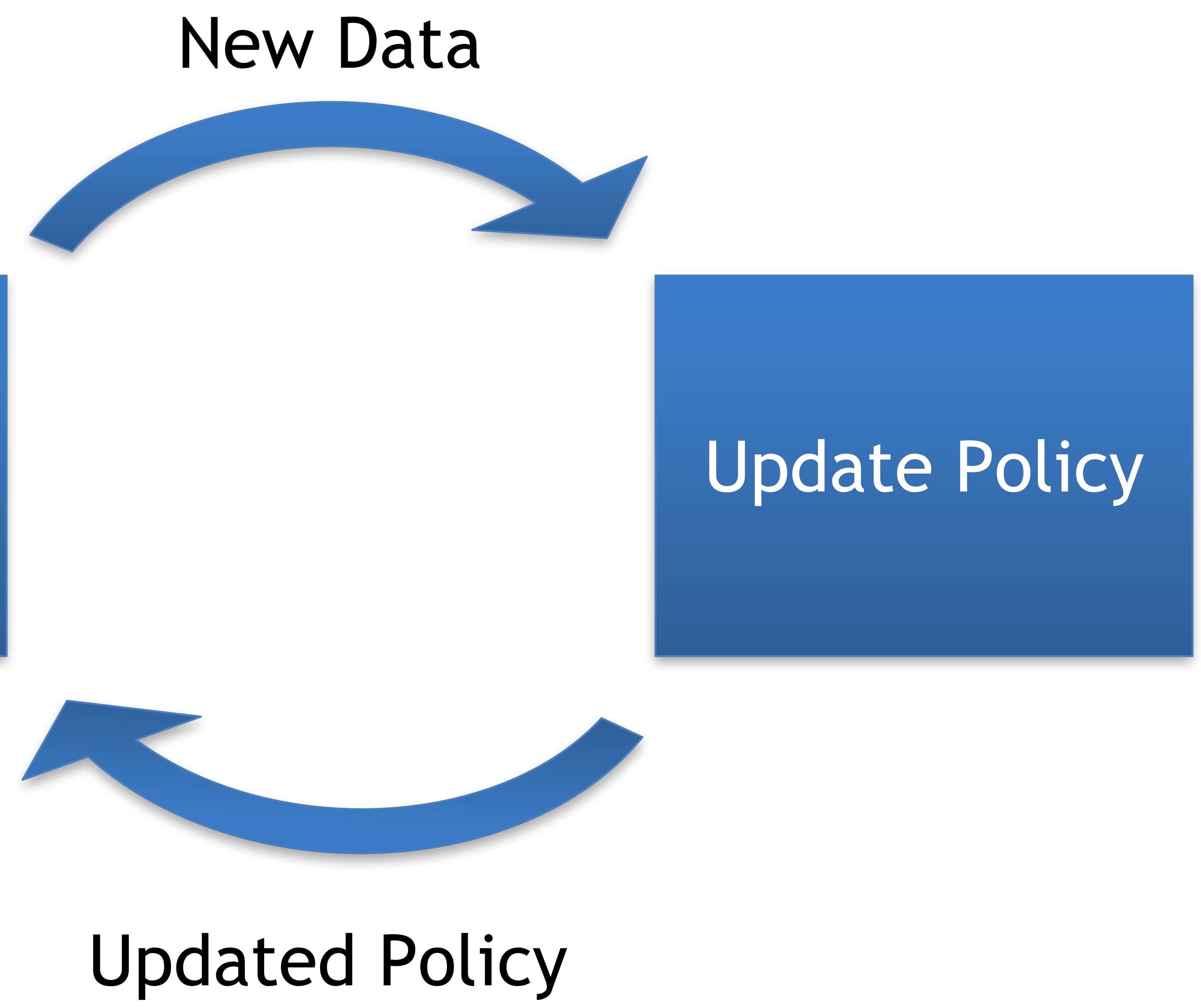
- Recap++
- Imitation Learning:
  - Behavioral Cloning
  - ✓ DAgger

Intuitive solution: Interaction



Use interaction to collect data where learned policy goes

General Idea: Iterative Interactive Approach

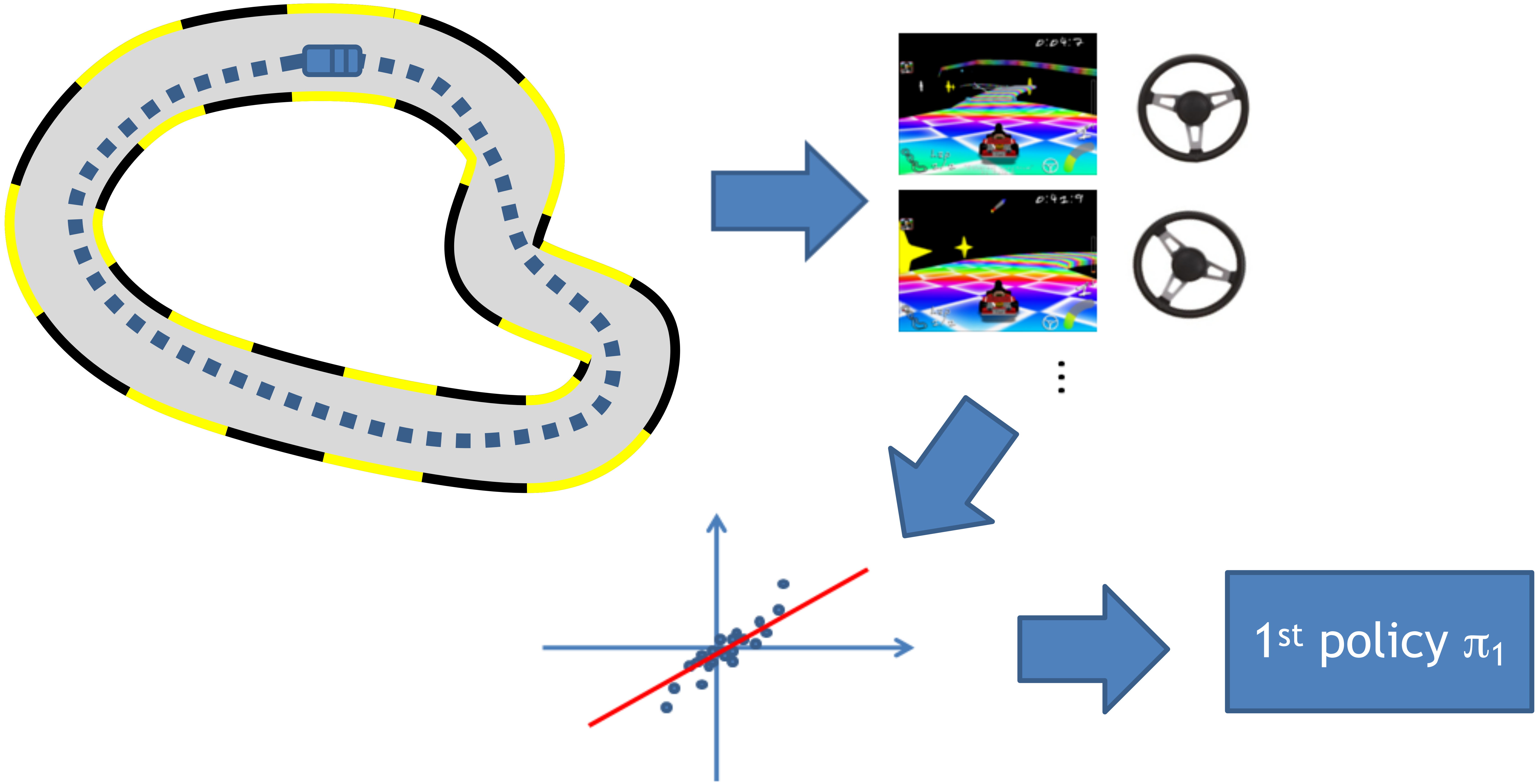


DAgger: Dataset Aggregation

[Ross11a]

0th iteration

Expert Demonstrates Task **Dataset**

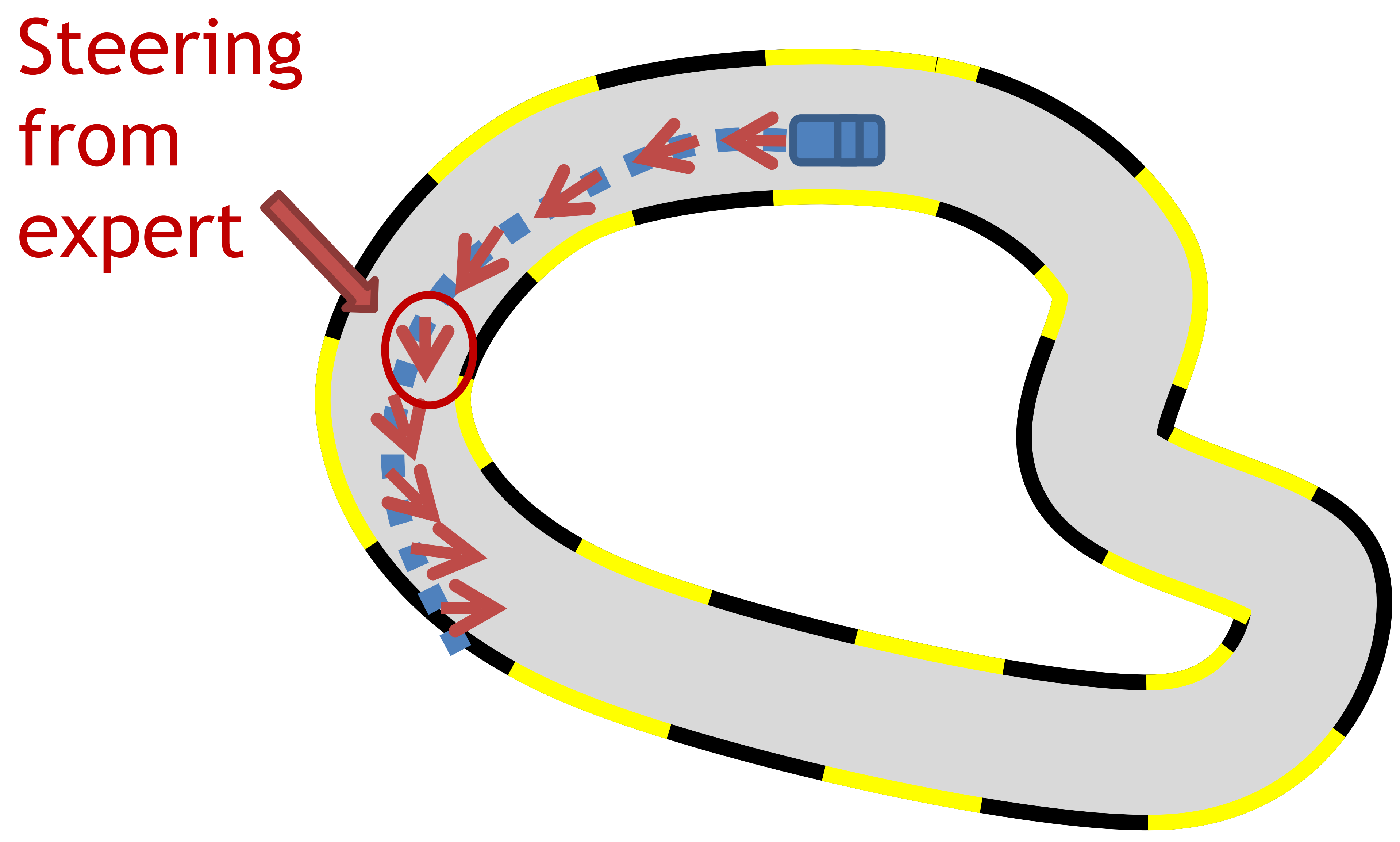


Supervised Learning

DAgger: Dataset Aggregation [Ross11a]

1st iteration

Execute π_1 and Query Expert

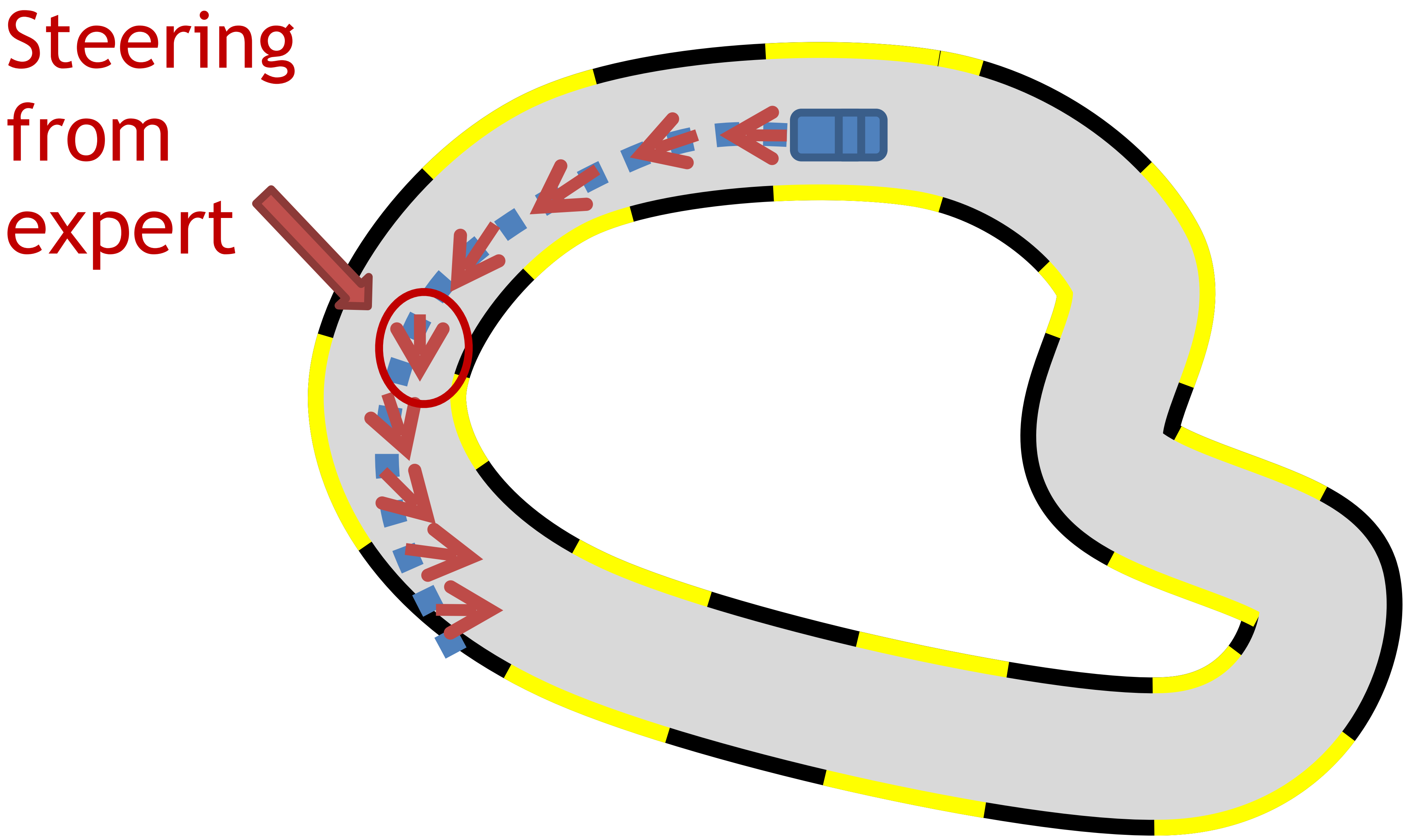


DAgger: Dataset Aggregation

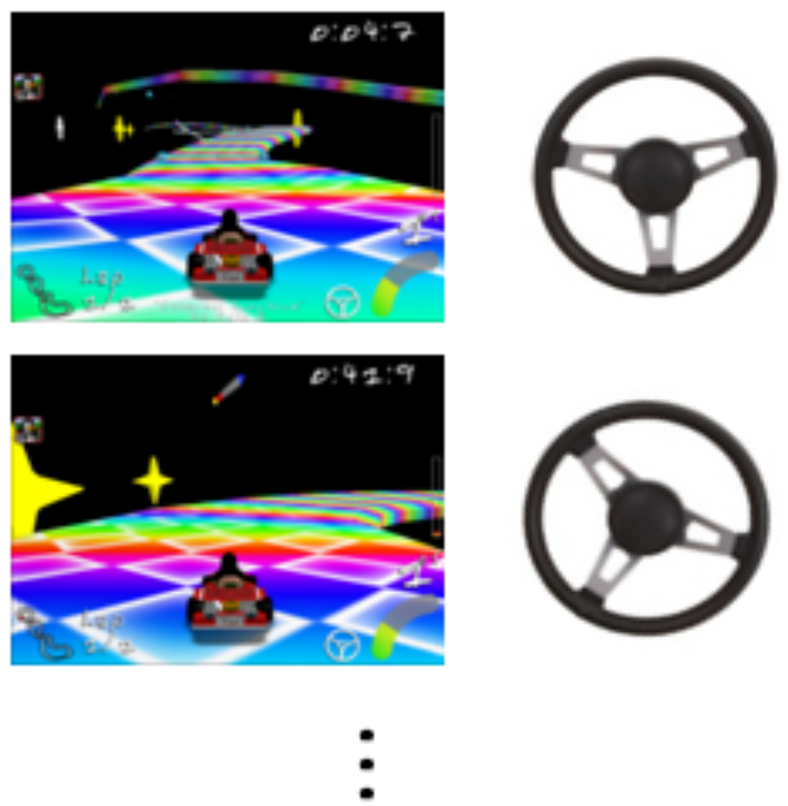
[Ross11a]

1st iteration

Execute π_1 and Query Expert



New Data

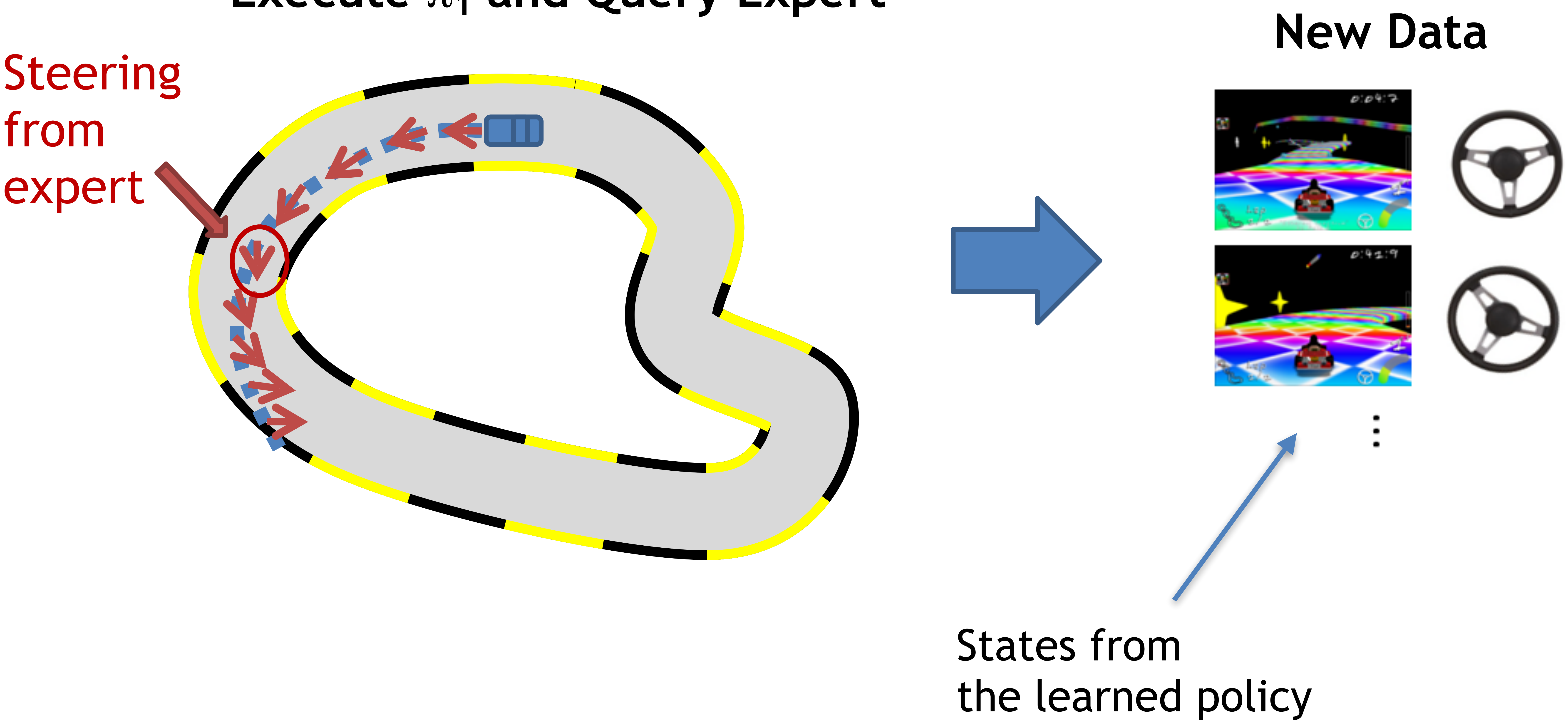


DAgger: Dataset Aggregation

[Ross11a]

1st iteration

Execute π_1 and Query Expert

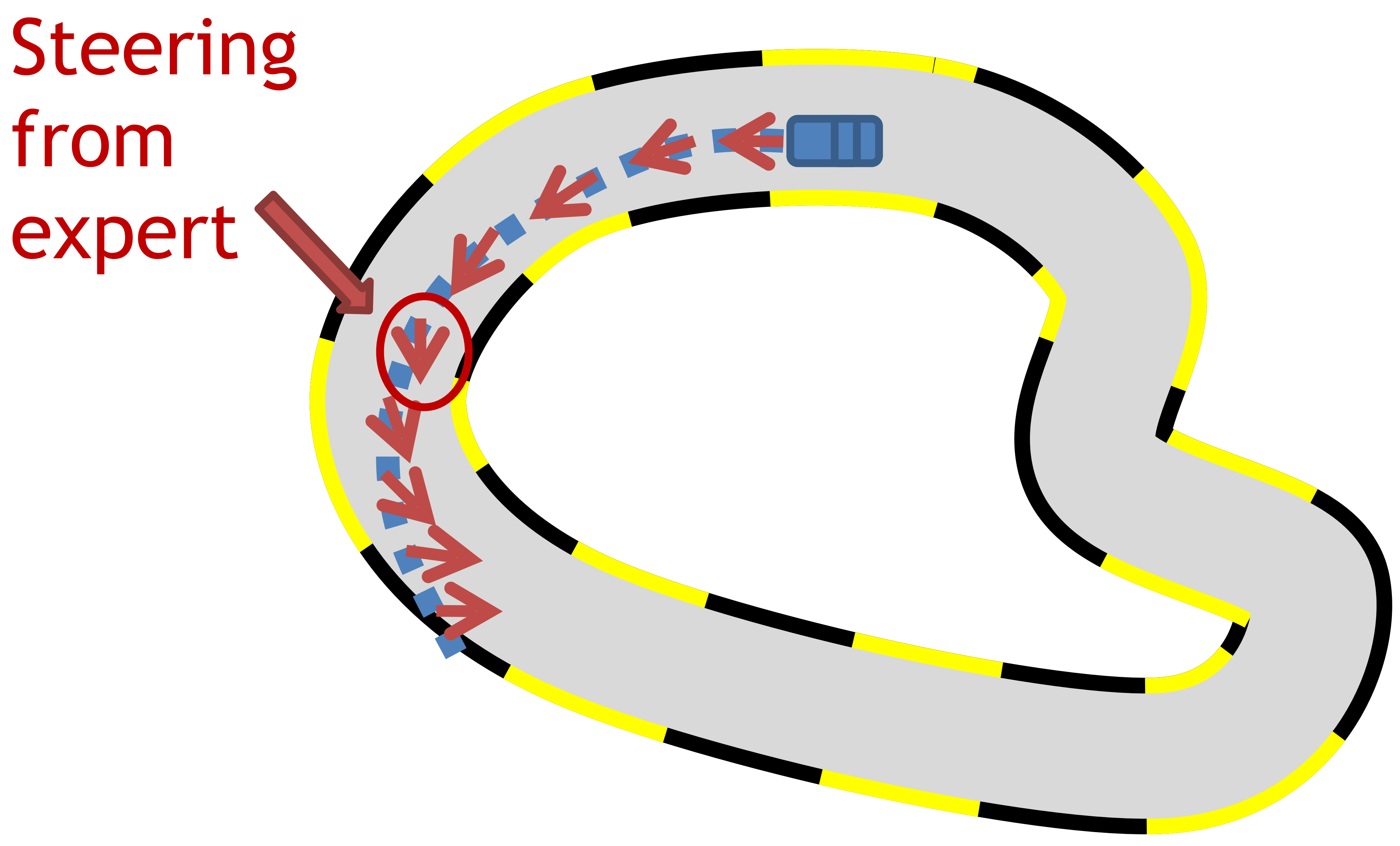


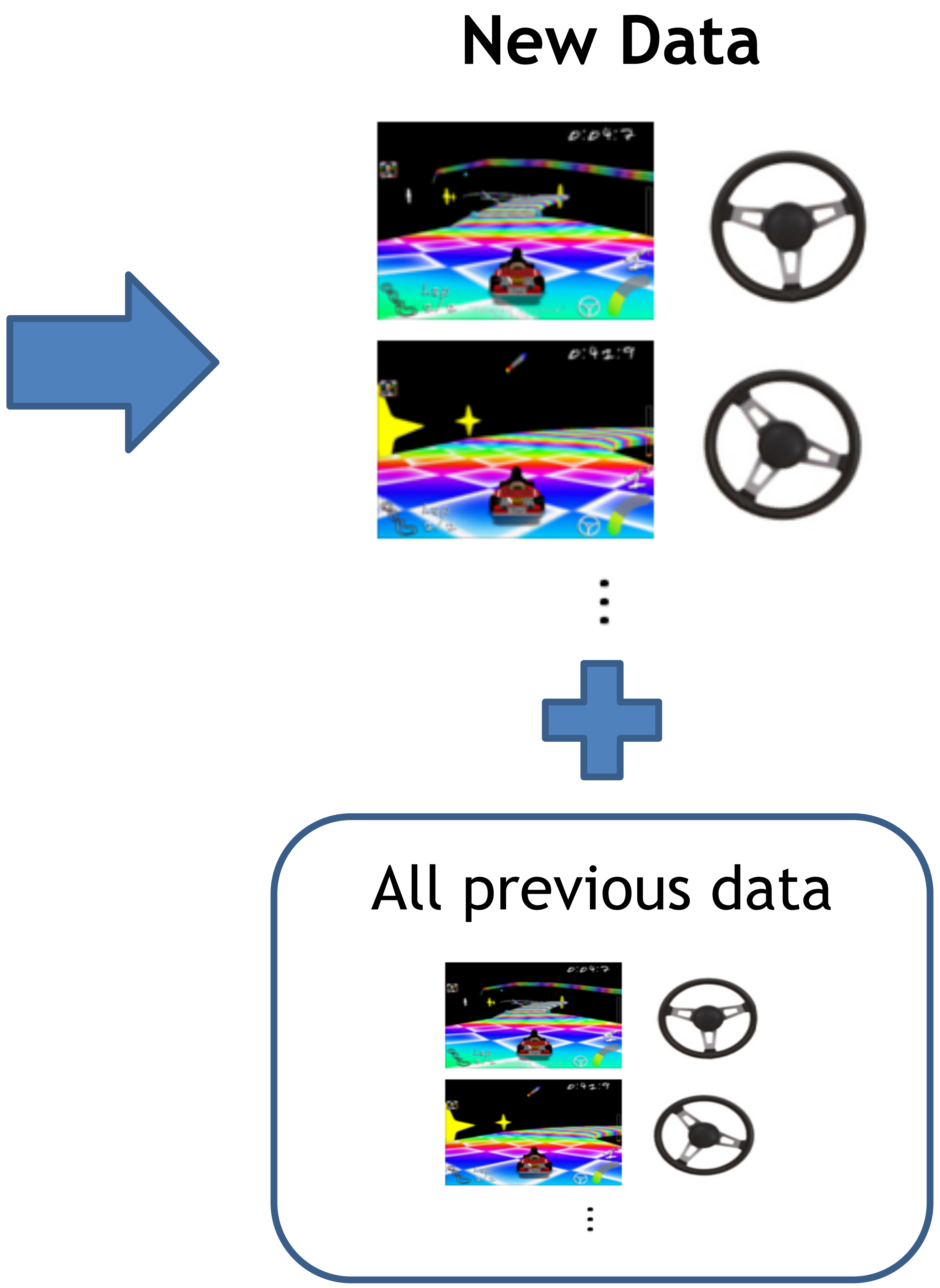
[Ross11a]

DAgger: Dataset Aggregation

1st iteration

Execute π_1 and Query Expert

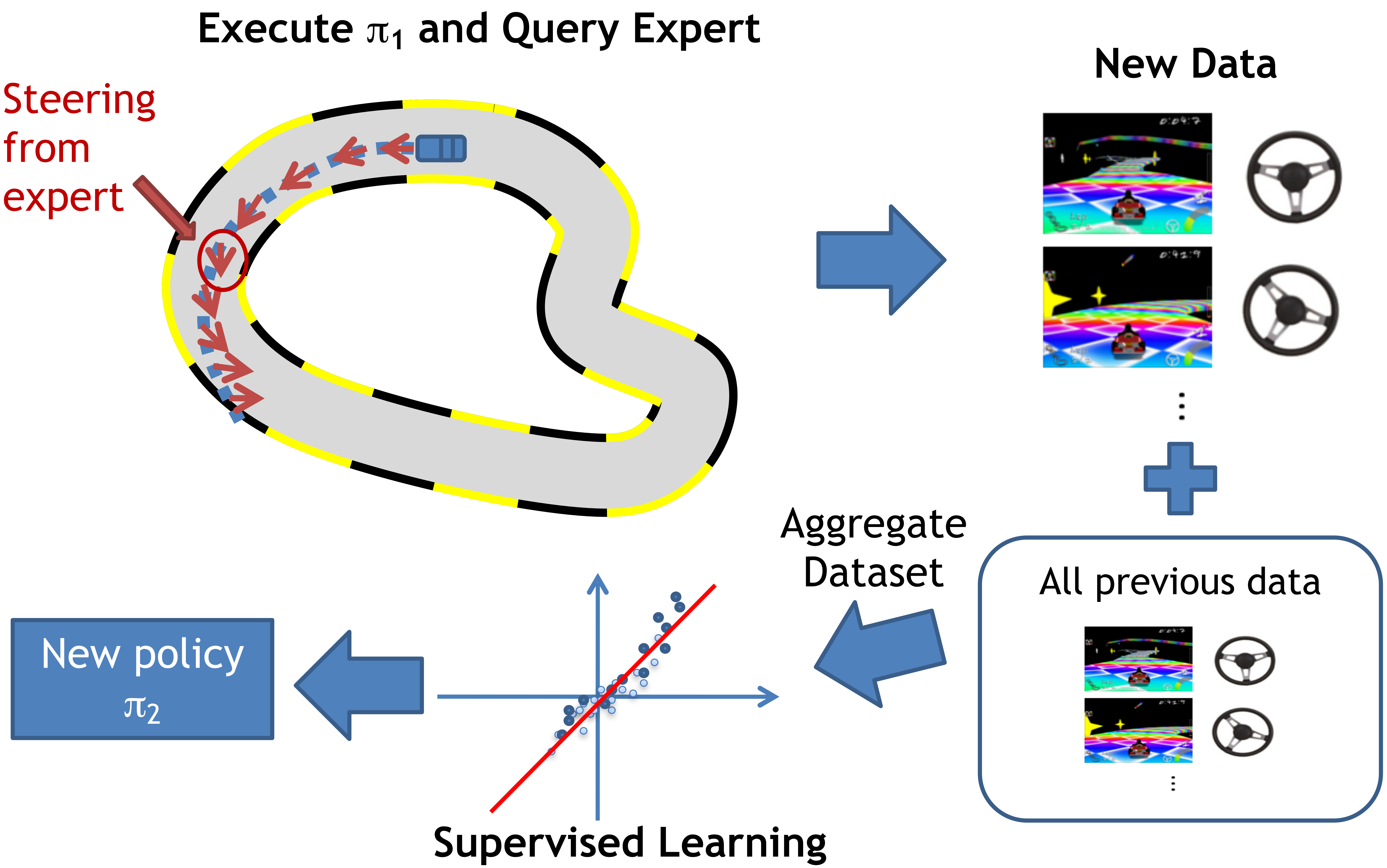




DAgger: Dataset Aggregation

[Ross11a]

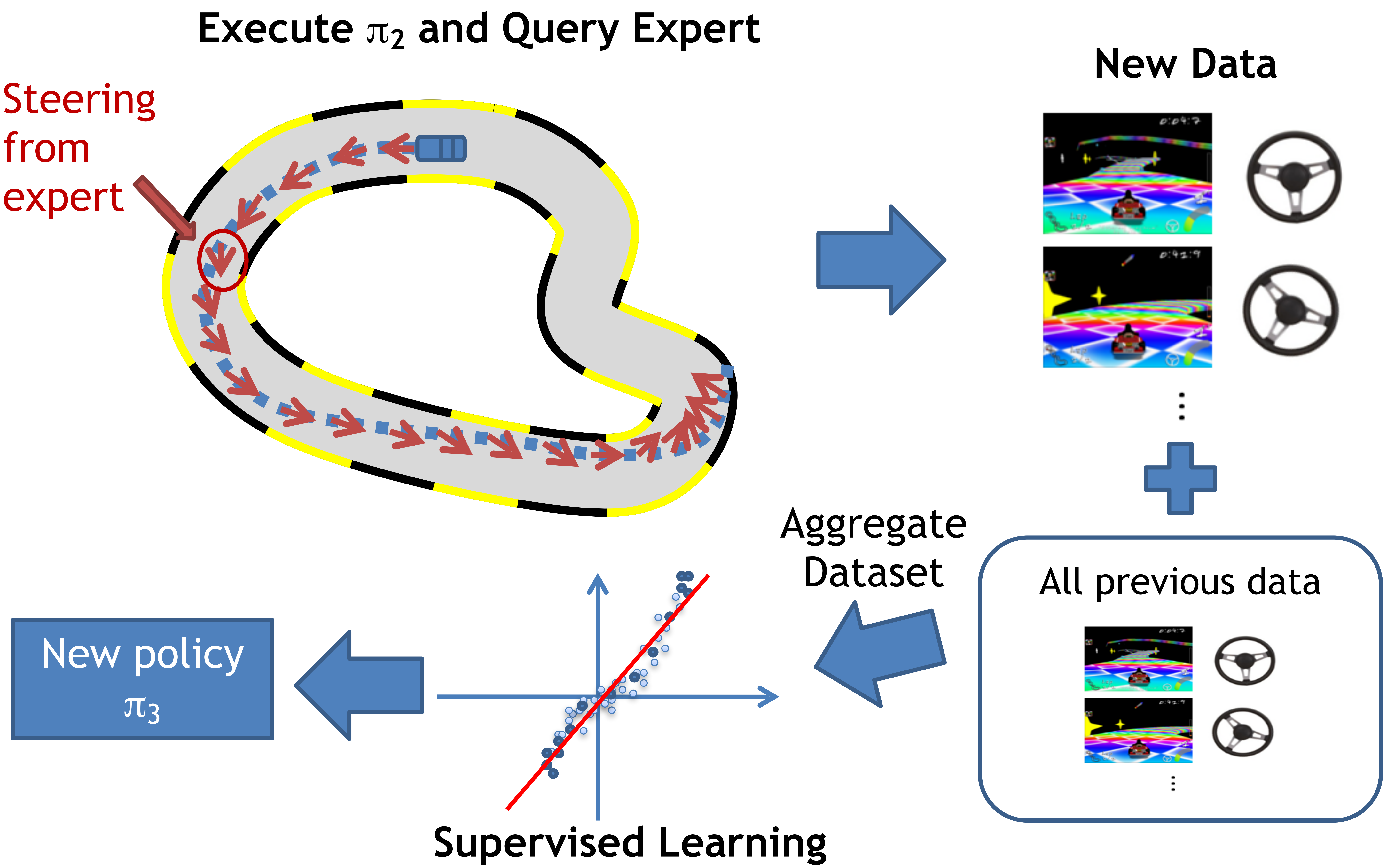
1st iteration



[Ross11a]

DAgger: Dataset Aggregation

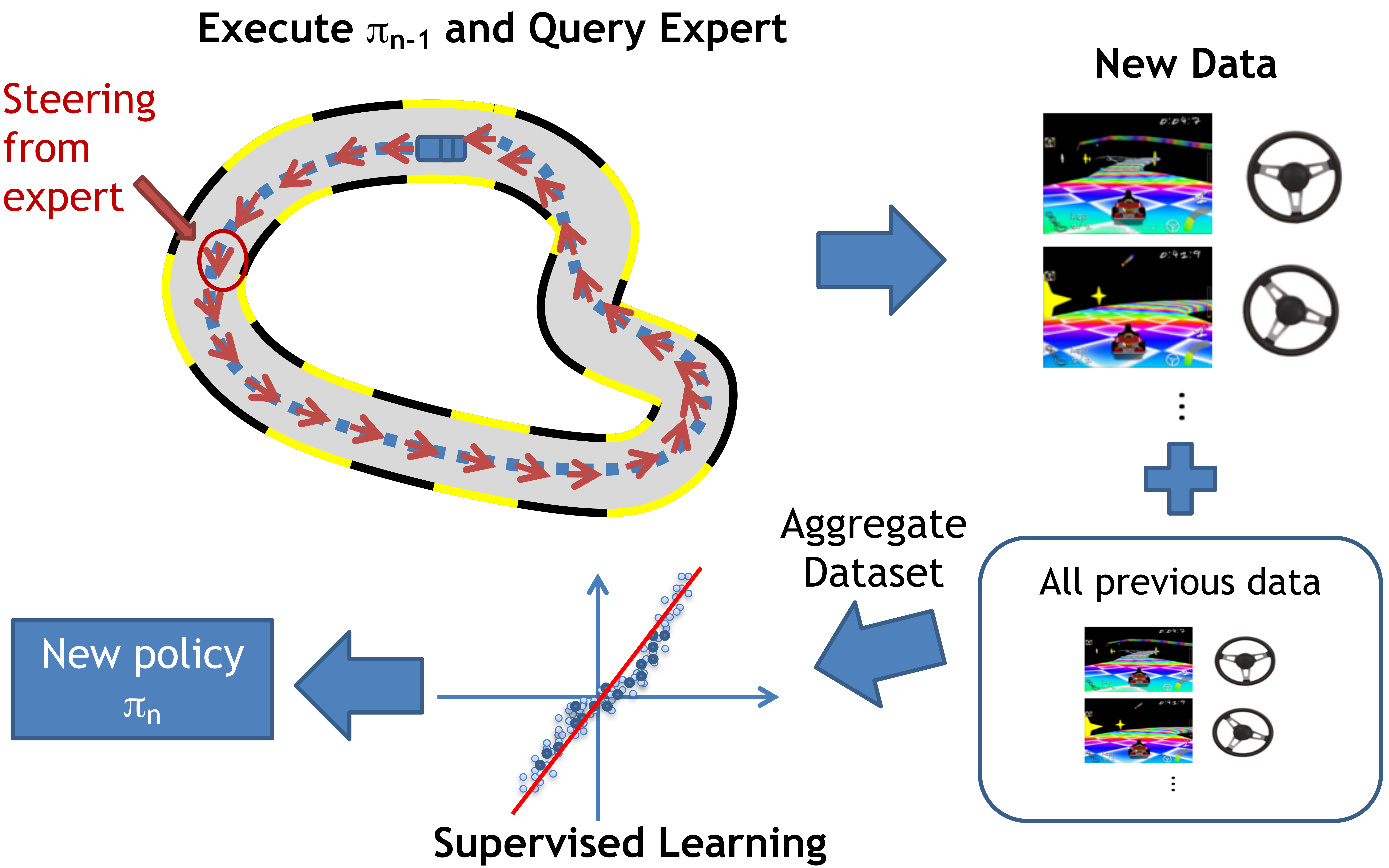
2nd iteration



DAgger: Dataset Aggregation

[Ross11a]

n^{th} iteration



The DAgger algorithm

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

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

The DAgger algorithm

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

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

1. W/ π^t, generate dataset of trajectories $\mathscr{D}^t = \{\tau_1, \tau_2, \ldots\}$
where for all trajectories $s_h \sim \rho_{\pi^t}$, $a_h = \pi^\star(s_h)$

The DAgger algorithm

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

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

1. W/ π^t, generate dataset of trajectories $\mathscr{D}^t = \{\tau_1, \tau_2, \ldots\}$
   where for all trajectories $s_h \sim \rho_{\pi^t}$, $a_h = \pi^\star(s_h)$
2. **Data aggregation:** $\mathscr{D} = \mathscr{D} \cup \mathscr{D}^t$

The DAgger algorithm

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

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

1. W/ π^t, generate dataset of trajectories $\mathscr{D}^t = \{\tau_1, \tau_2, \ldots\}$
where for all trajectories $s_h \sim \rho_{\pi^t}, a_h = \pi^\star(s_h)$
2. **Data aggregation:** $\mathscr{D} = \mathscr{D} \cup \mathscr{D}^t$
3. **Update policy via Supervised-Learning**: $\pi^{t+1} = \text{SL}\left(\mathscr{D}\right)$

The DAgger algorithm

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

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

1. W/ π^t, generate dataset of trajectories $\mathscr{D}^t = \{\tau_1, \tau_2, \ldots\}$ where for all trajectories $s_h \sim \rho_{\pi^t}$, $a_h = \pi^\star(s_h)$
2. **Data aggregation:** $\mathscr{D} = \mathscr{D} \cup \mathscr{D}^t$
3. **Update policy via Supervised-Learning**: $\pi^{t+1} = \text{SL}\left(\mathscr{D}\right)$

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

[Informal Theorem] Under more assumptions + assuming ϵ SL error is achievable, the DAgger algorithm has error: $|V^{\pi^\star} - V^{\widehat{\pi}}| \leq H\epsilon$

Success!

[Ross AISTATS 2011]

Success!

[Ross AISTATS 2011]

Summary:

1. NPG: a simpler way to do TRPO, a “pre-conditioned” gradient method.
2. PPO: “first order” approx to TRPO

Attendance:
bit.ly/3RcTC9T

Feedback:
bit.ly/3RHtlxy