Model-Based Reinforcement Learning

CMPT 729 G100

Jason Peng

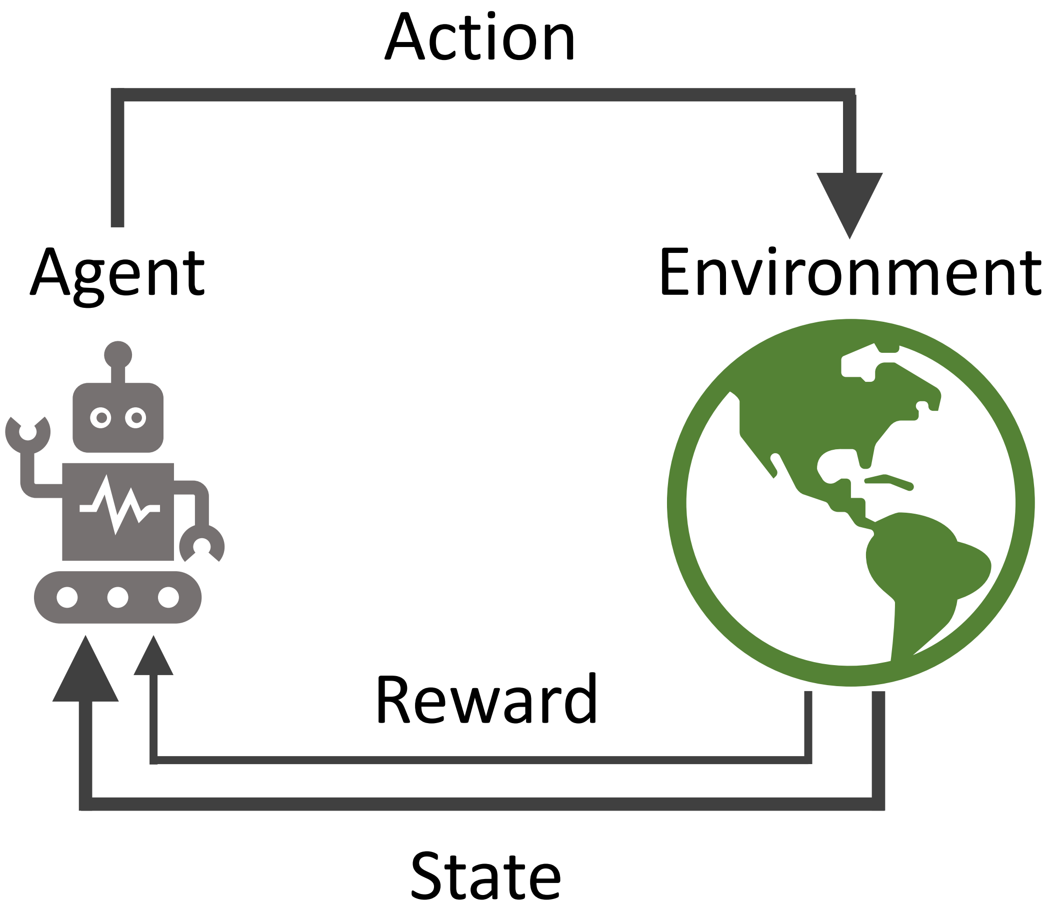
Overview

- Model-Based RL
- DYNA
- Model Representations
- Uncertainty Estimation
- MPC

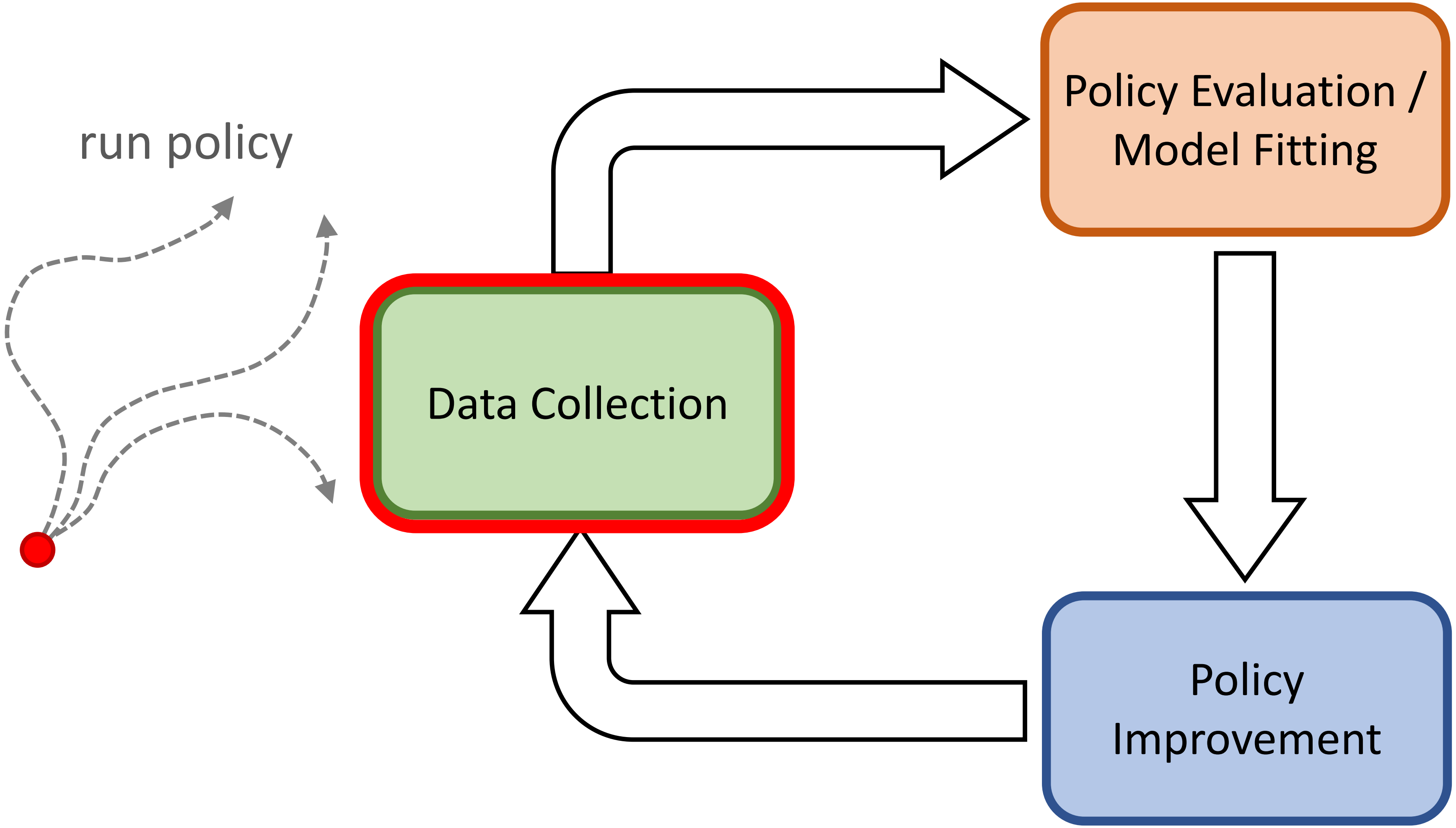
Taxonomy of RL Algorithms

- Policy-Based Methods
- Value-Based Methods
- Actor-Critic Methods
- Model-Based Methods

Reinforcement Learning



RL Algorithms



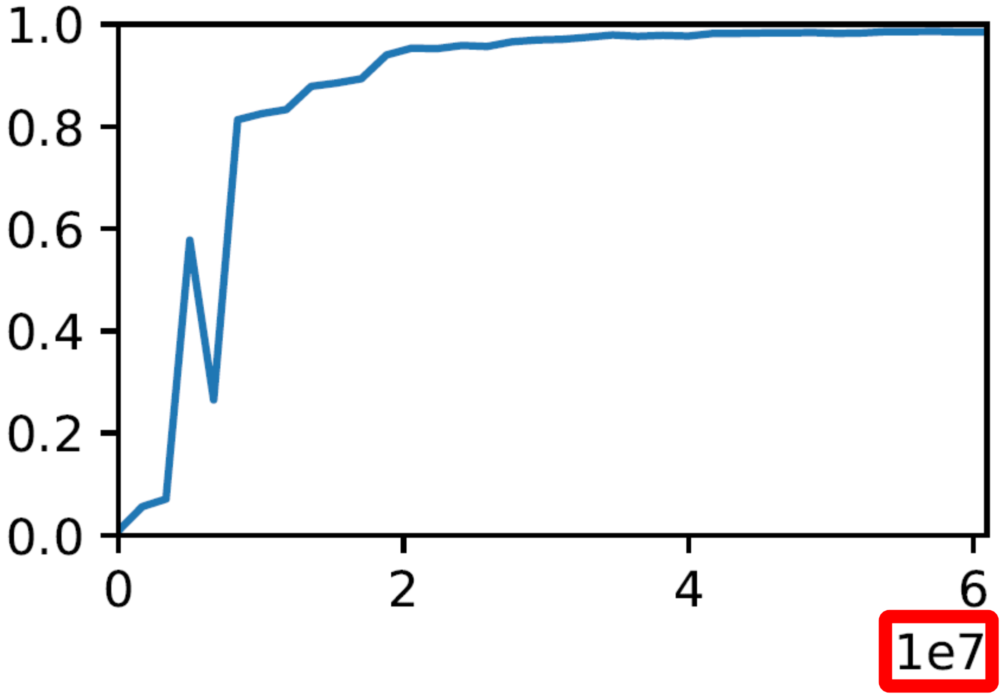
Sample Complexity

Simulation

Learning Agile Robotic Locomotion Skills by Imitating Animals
[Peng et al. 2020]

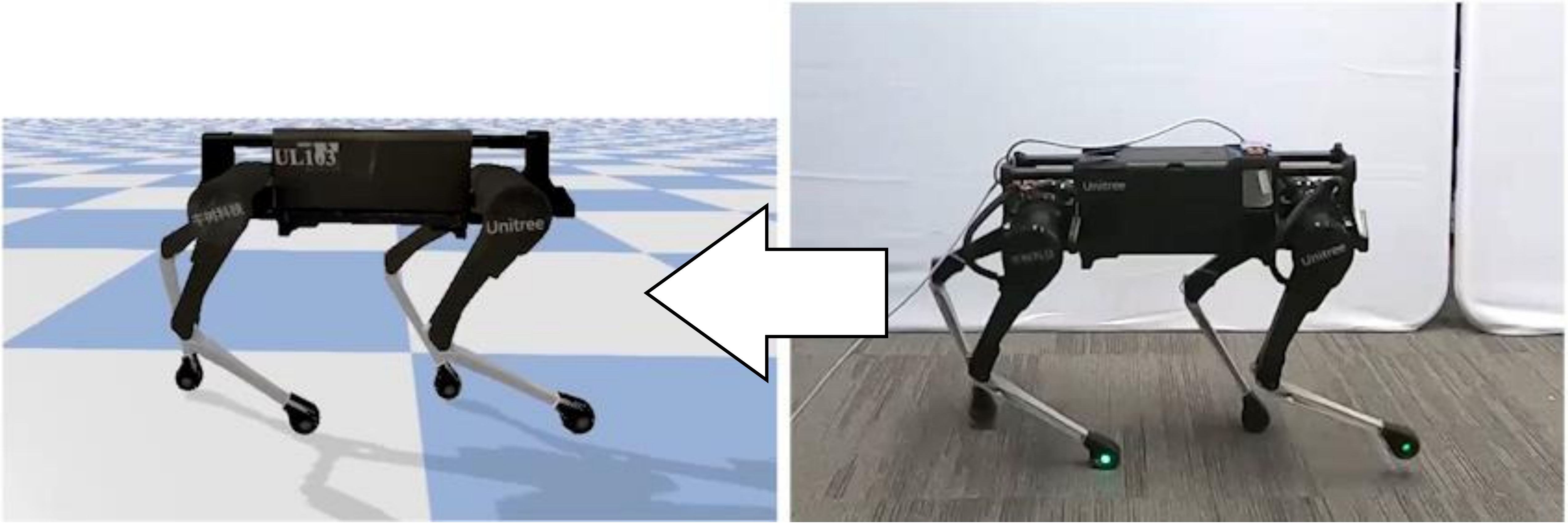
Sample Complexity

Simulation



Learning Agile Robotic Locomotion Skills by Imitating Animals
[Peng et al. 2020]

Sample Complexity



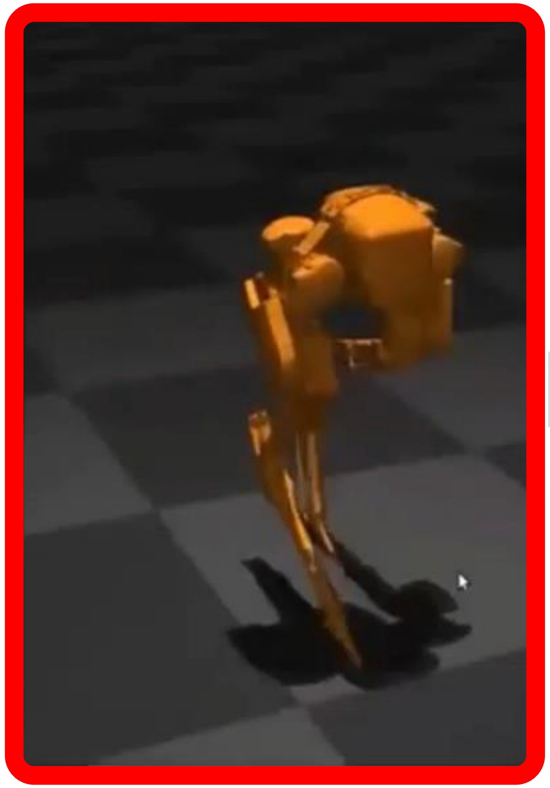
Simulation Real World

Learning Agile Robotic Locomotion Skills by Imitating Animals
[Peng et al. 2020]

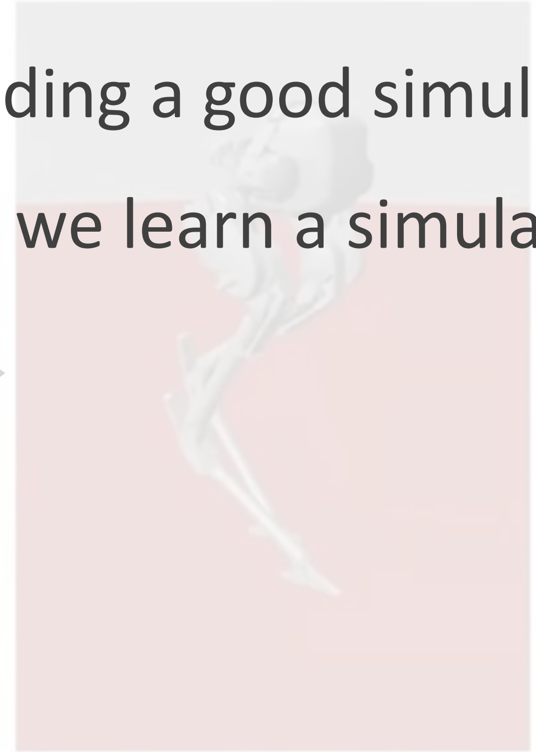
Sim-to-Real

Building a good simulator is hard

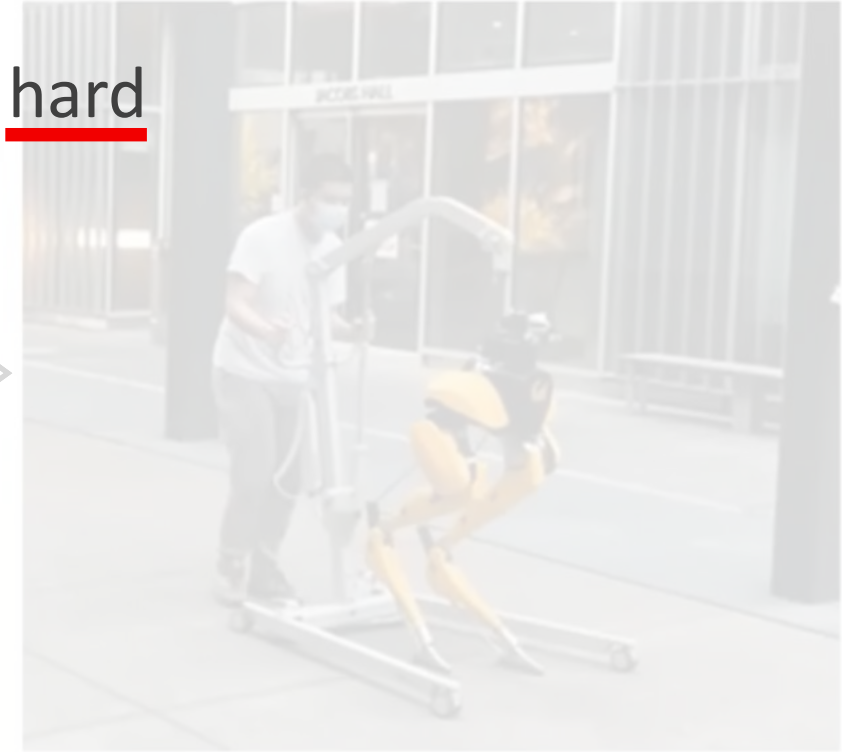
Can we learn a simulator?



Simulation
(Low-Fidelity)



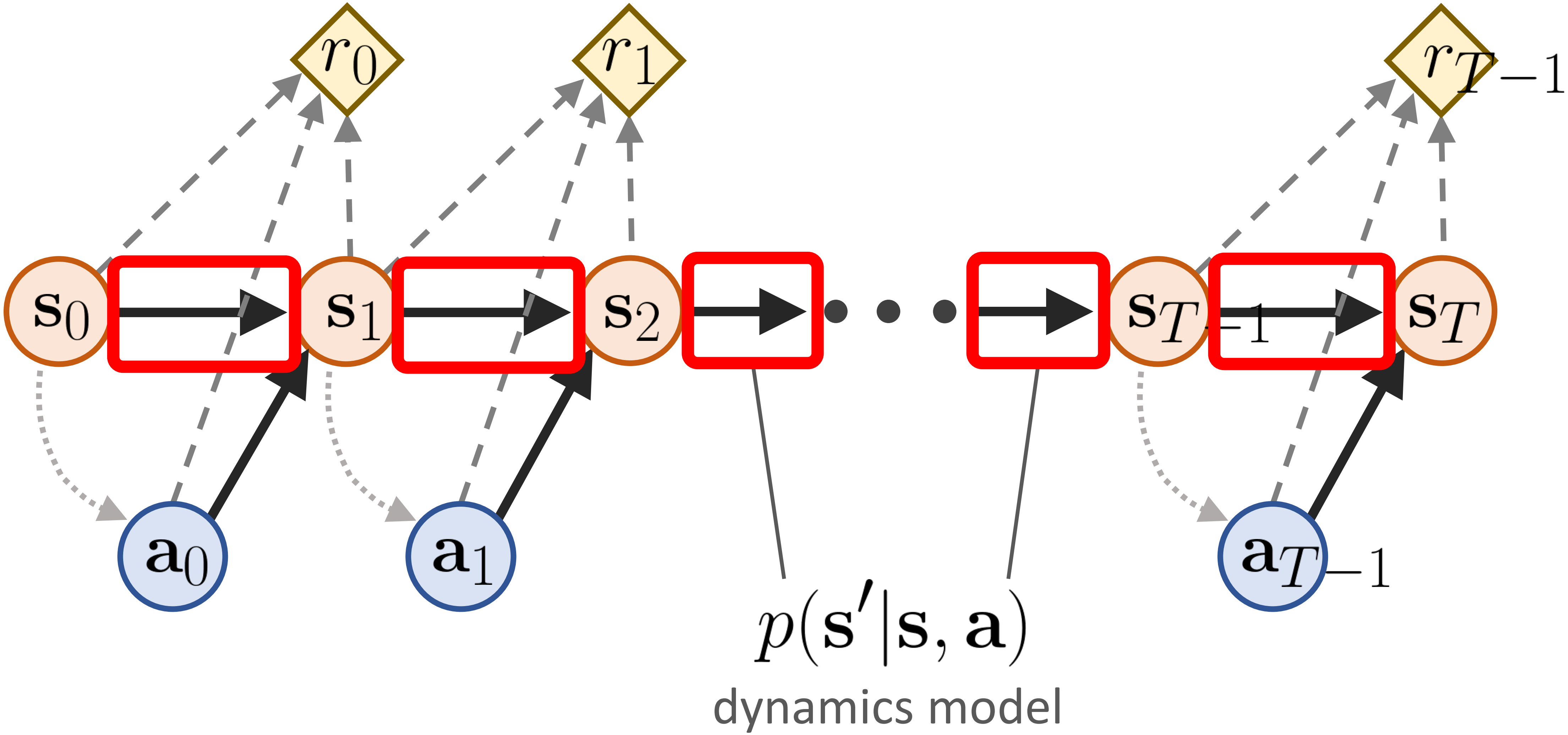
Simulation
(High-Fidelity)



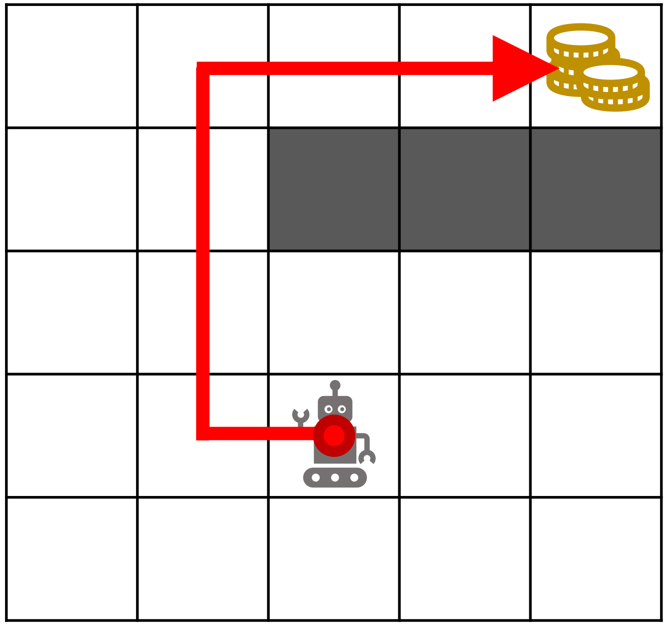
Real World

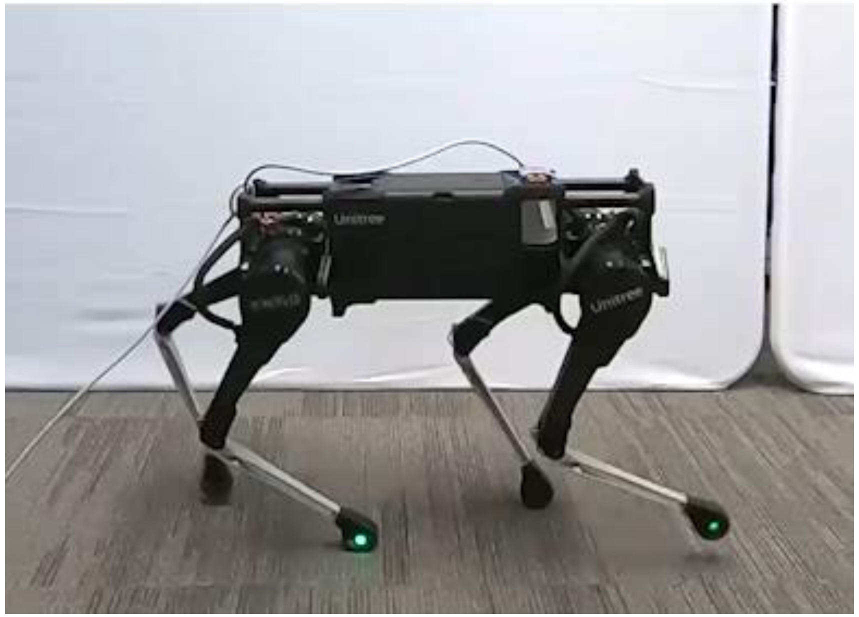
Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots
[Li et al. 2021]

Dynamics Model



Why Learn a Dynamics Model?





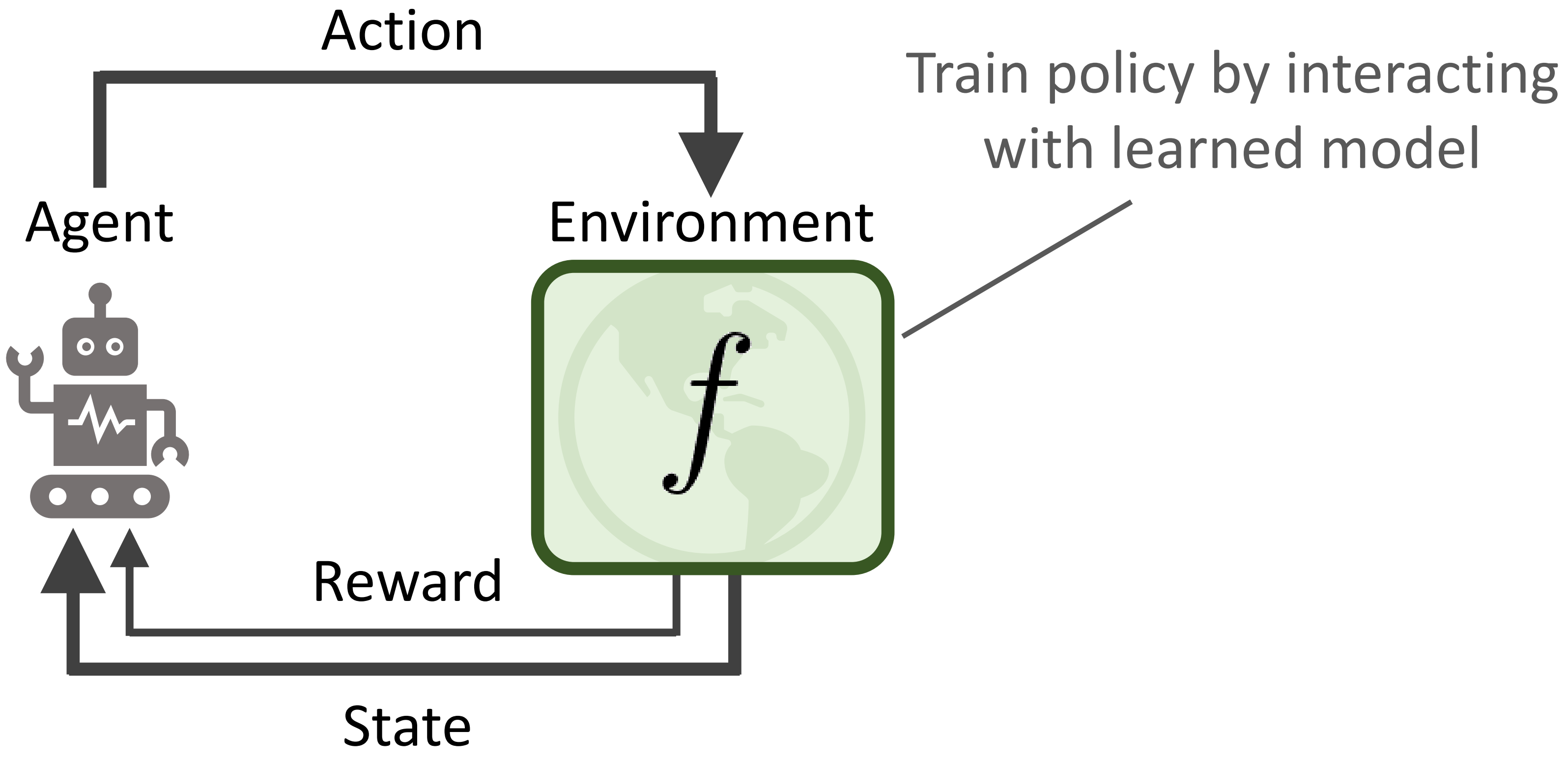
Simple Dynamics

Complex Dynamics

Dynamics Model

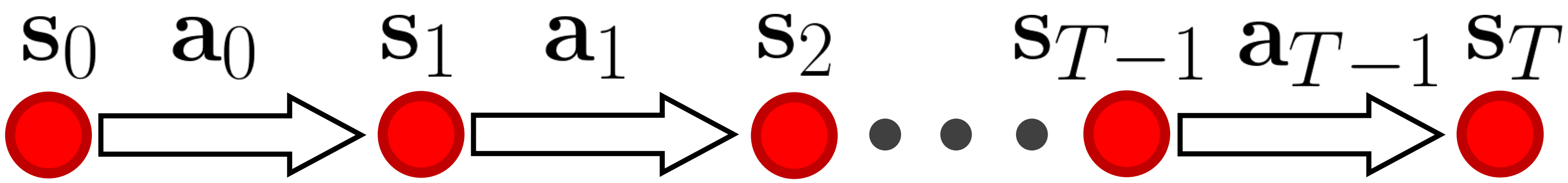
- Learn a dynamics model:

$$f(\mathbf{s}'|\mathbf{s},\mathbf{a}) \approx p(\mathbf{s}'|\mathbf{s},\mathbf{a})$$



Learning Dynamics Model

- Collect data with a base policy π_0

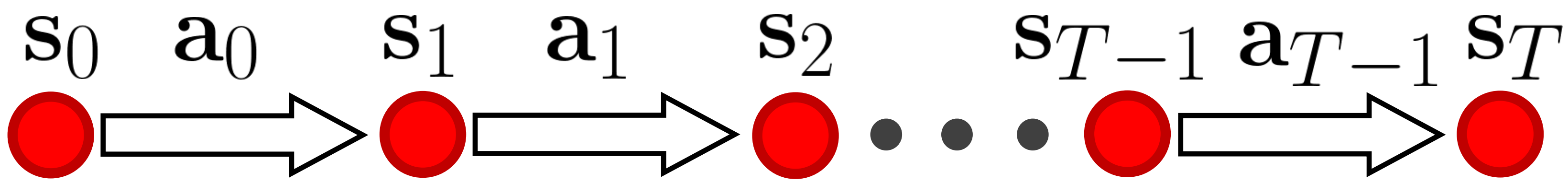


- Dataset: $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}')\}$
- Fit a dynamics model via supervised learning

$$\arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$$

Model-Based RL

- Collect data with a base policy π_0



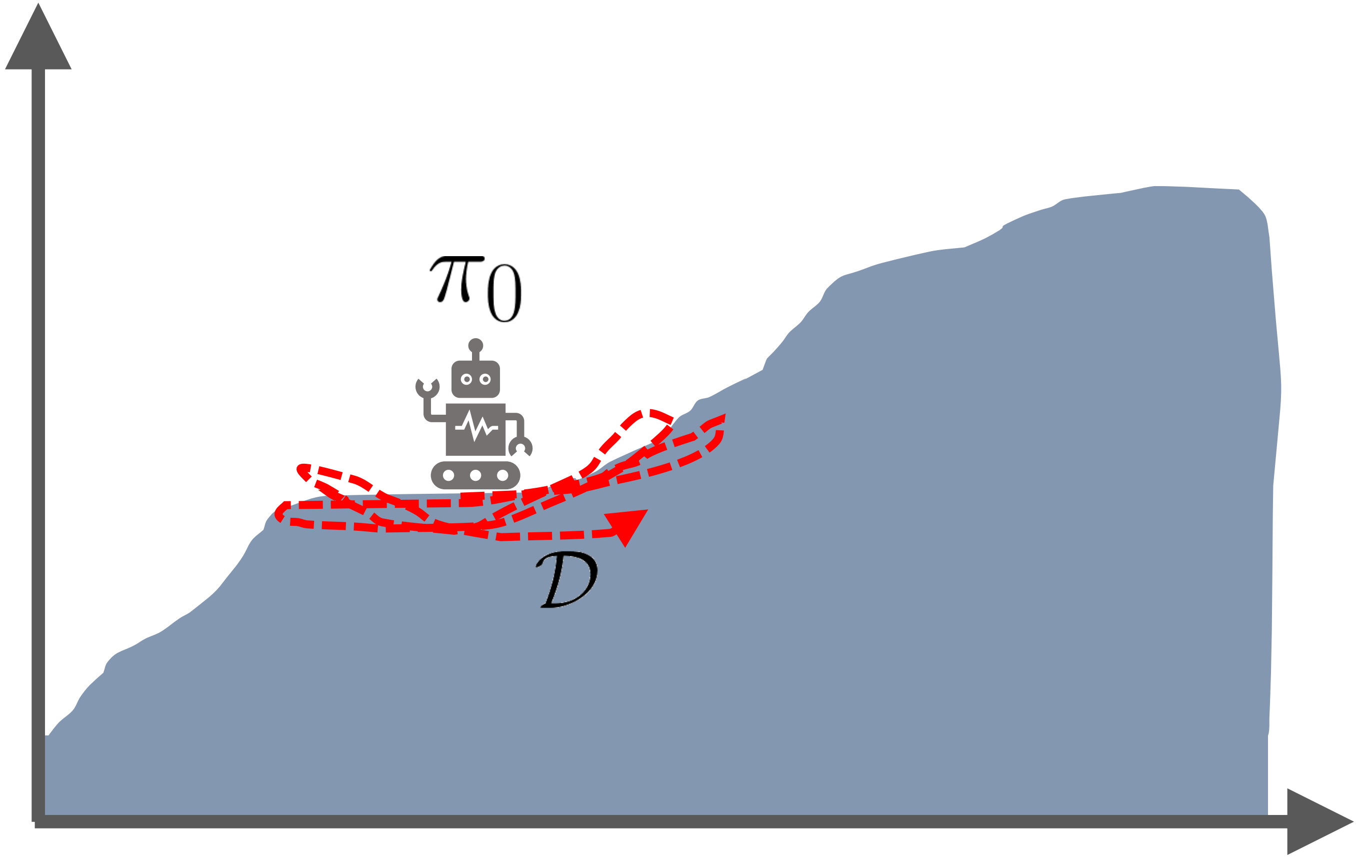
- Dataset: $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}')\}$
- Fit a dynamics model via supervised learning

$$\arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$$

- Train new policy π by simulating with $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$

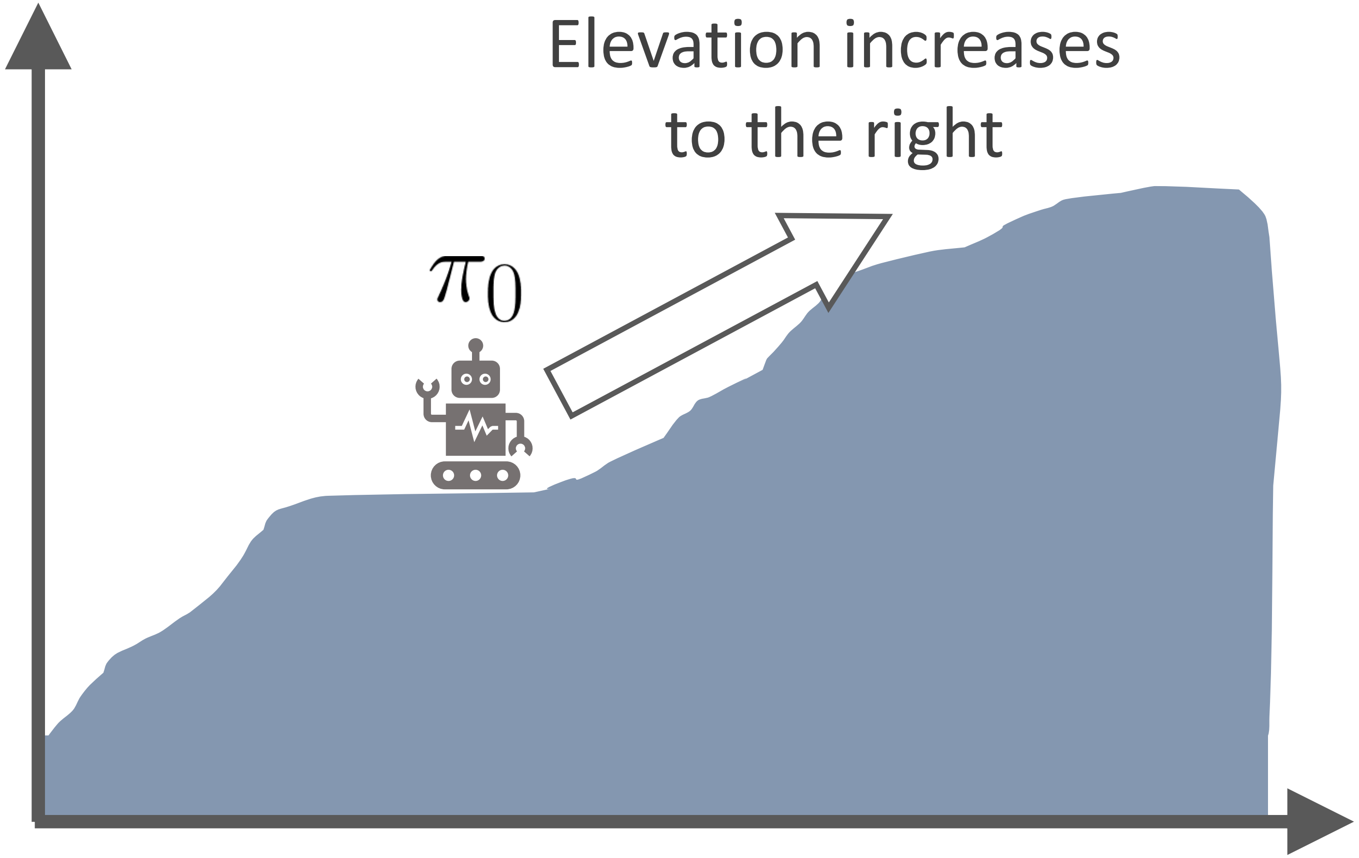
Problem

- Reward: climb as high as possible



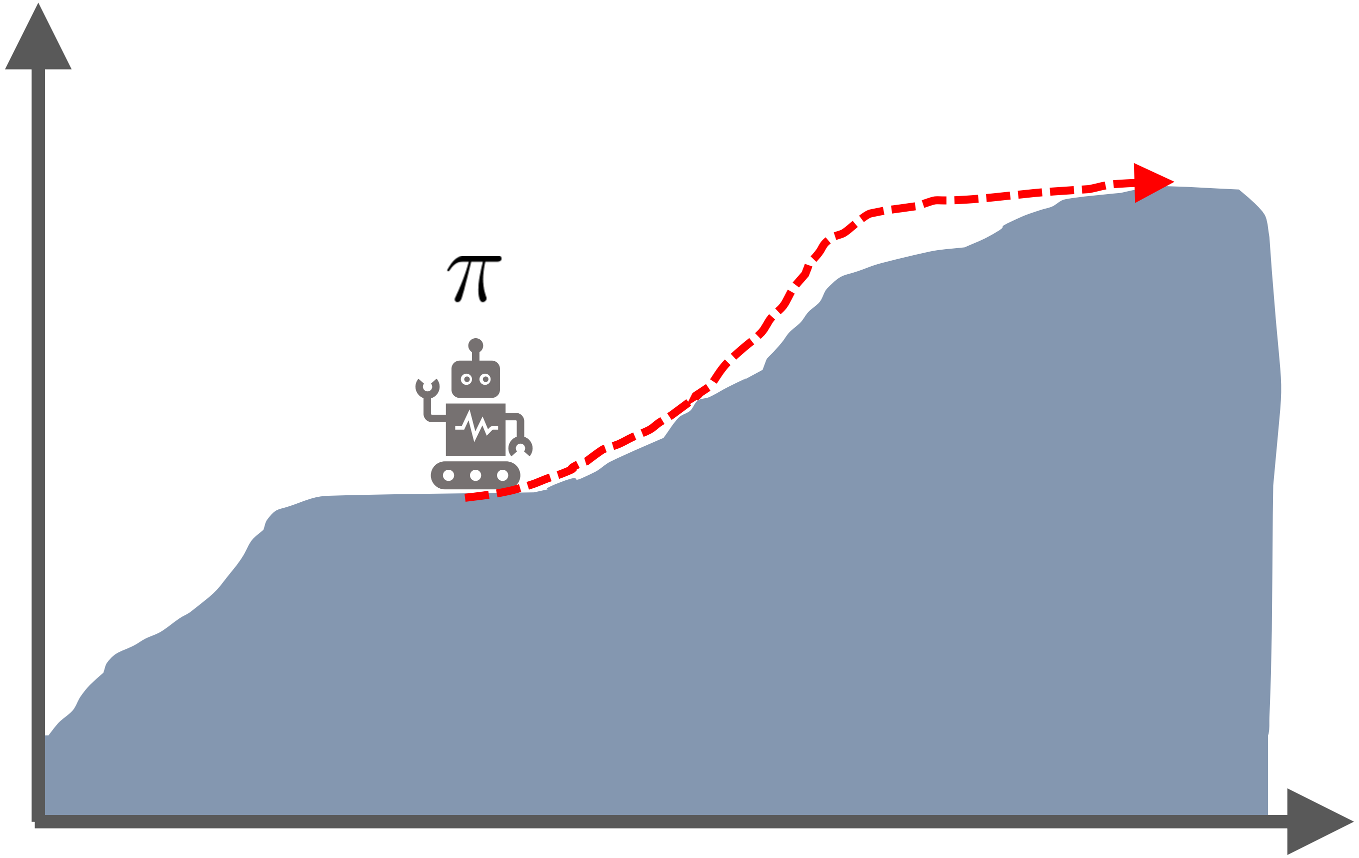
Problem

- Reward: climb as high as possible



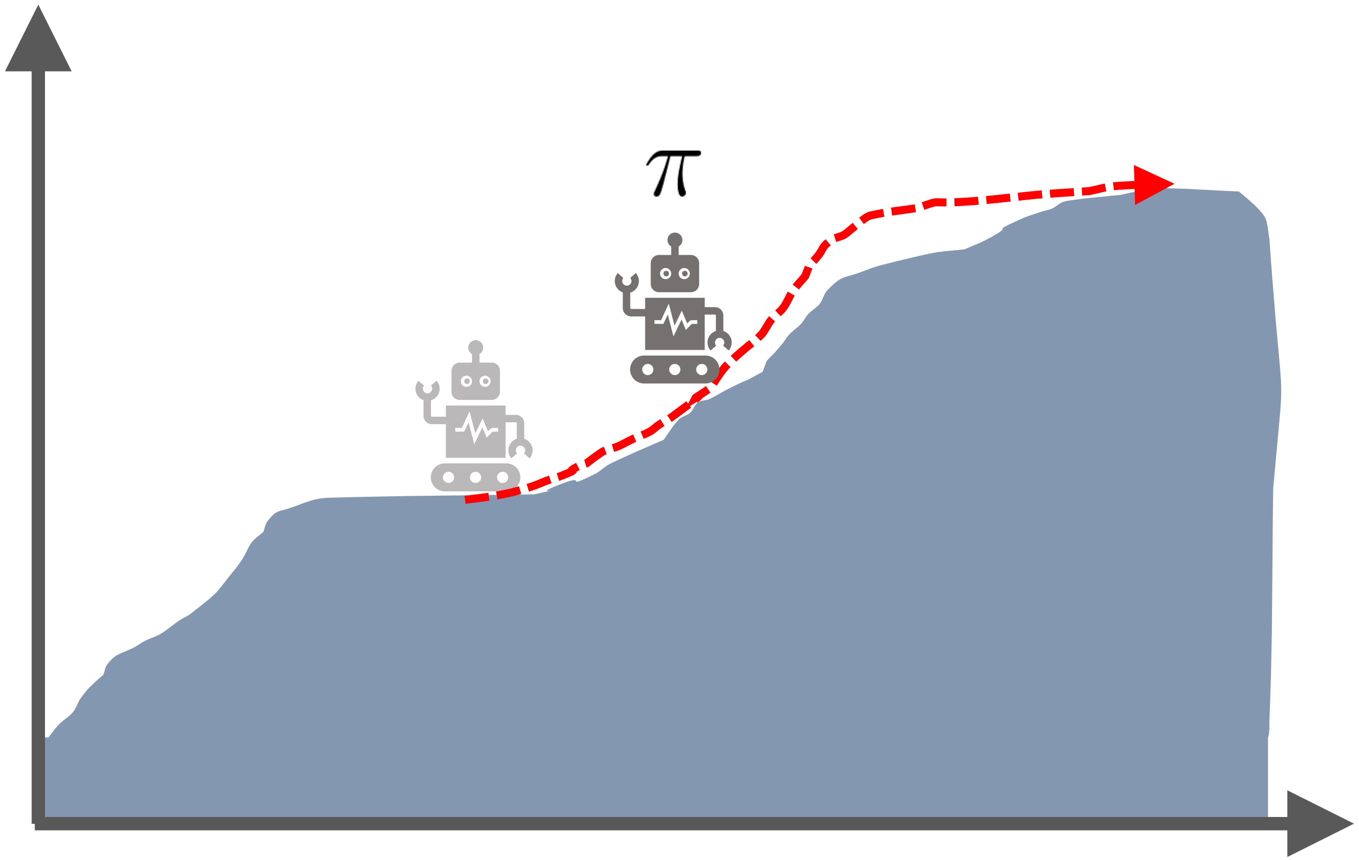
Problem

- Reward: climb as high as possible



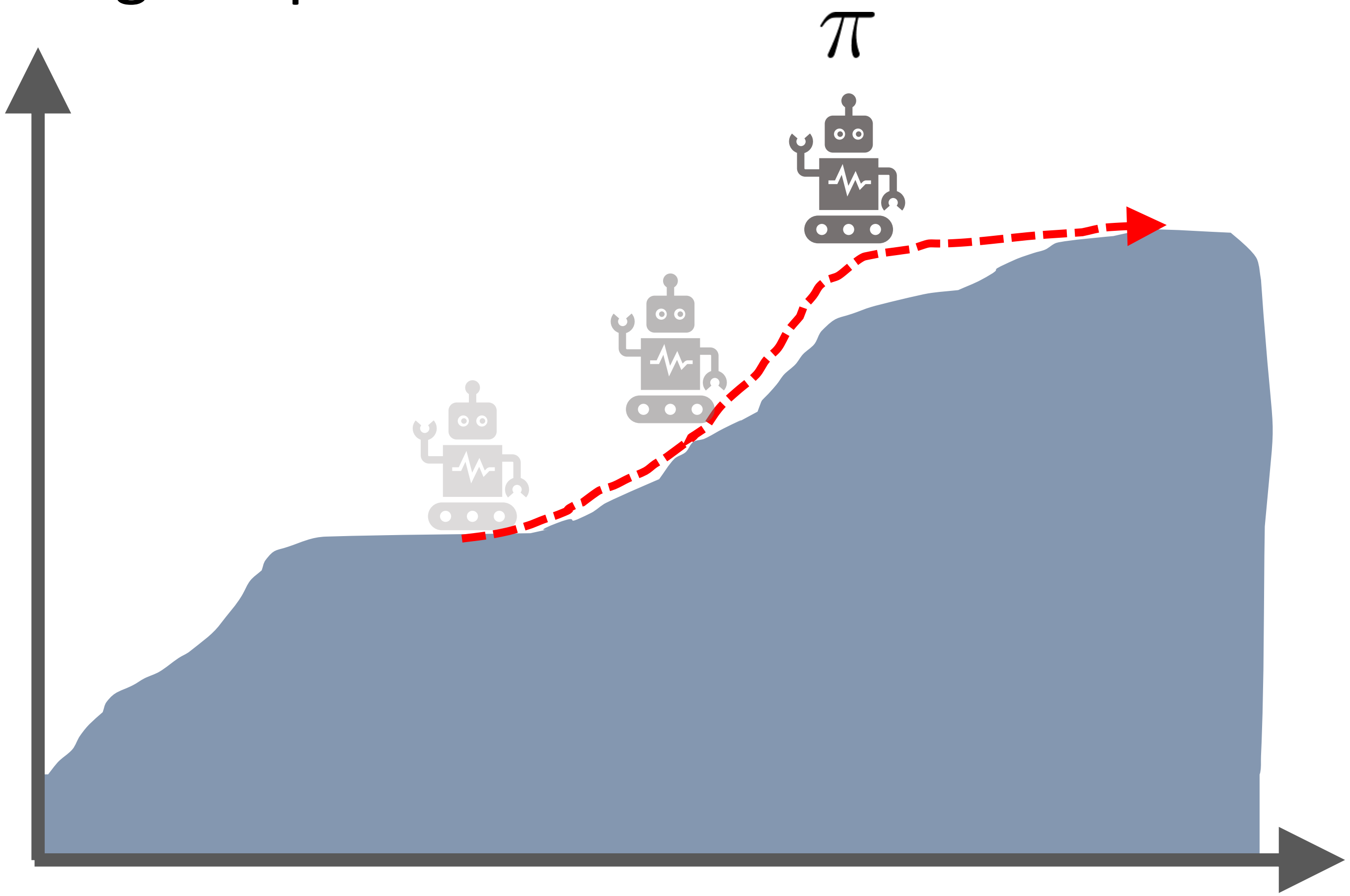
Problem

- Reward: climb as high as possible



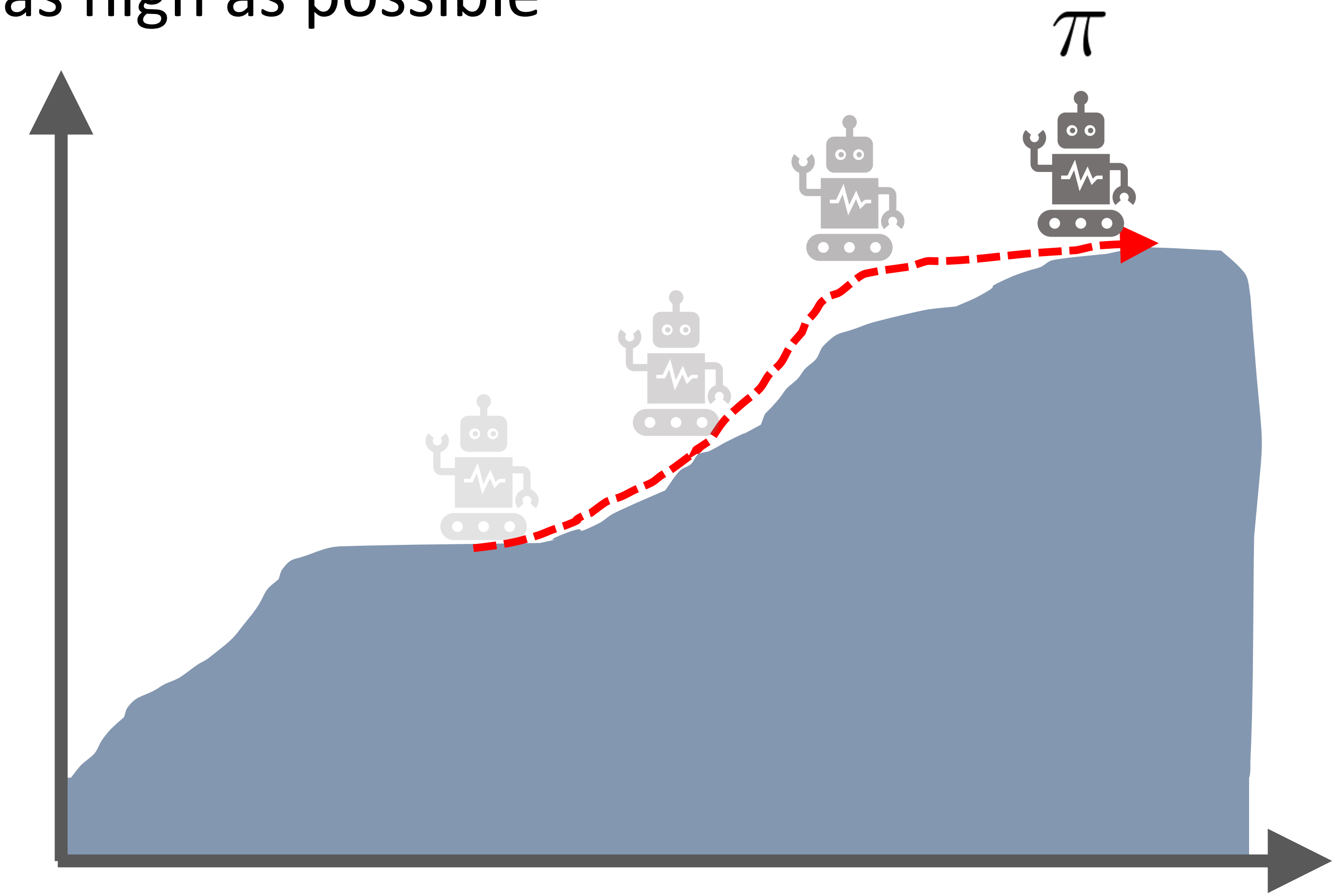
Problem

- Reward: climb as high as possible



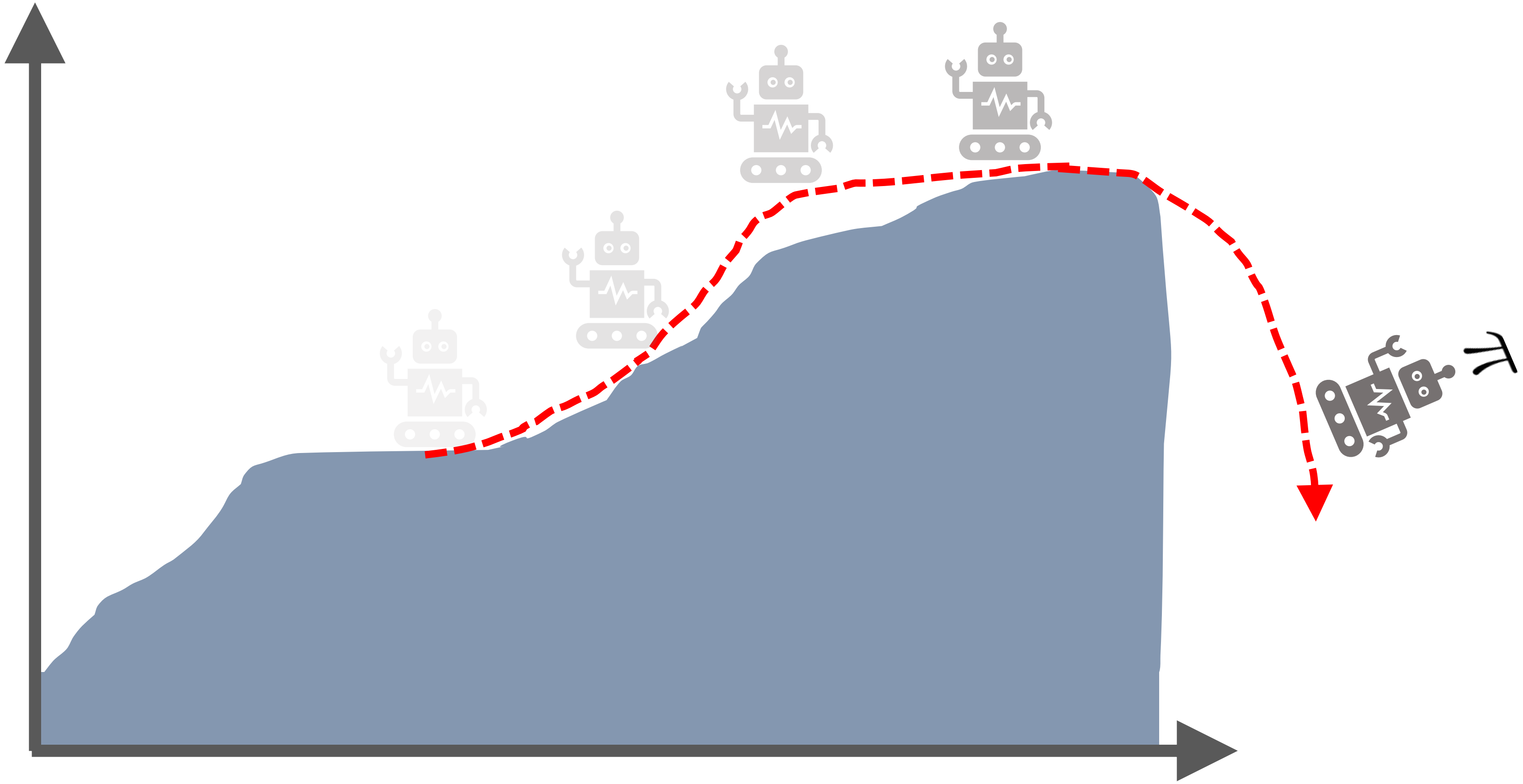
Problem

- Reward: climb as high as possible



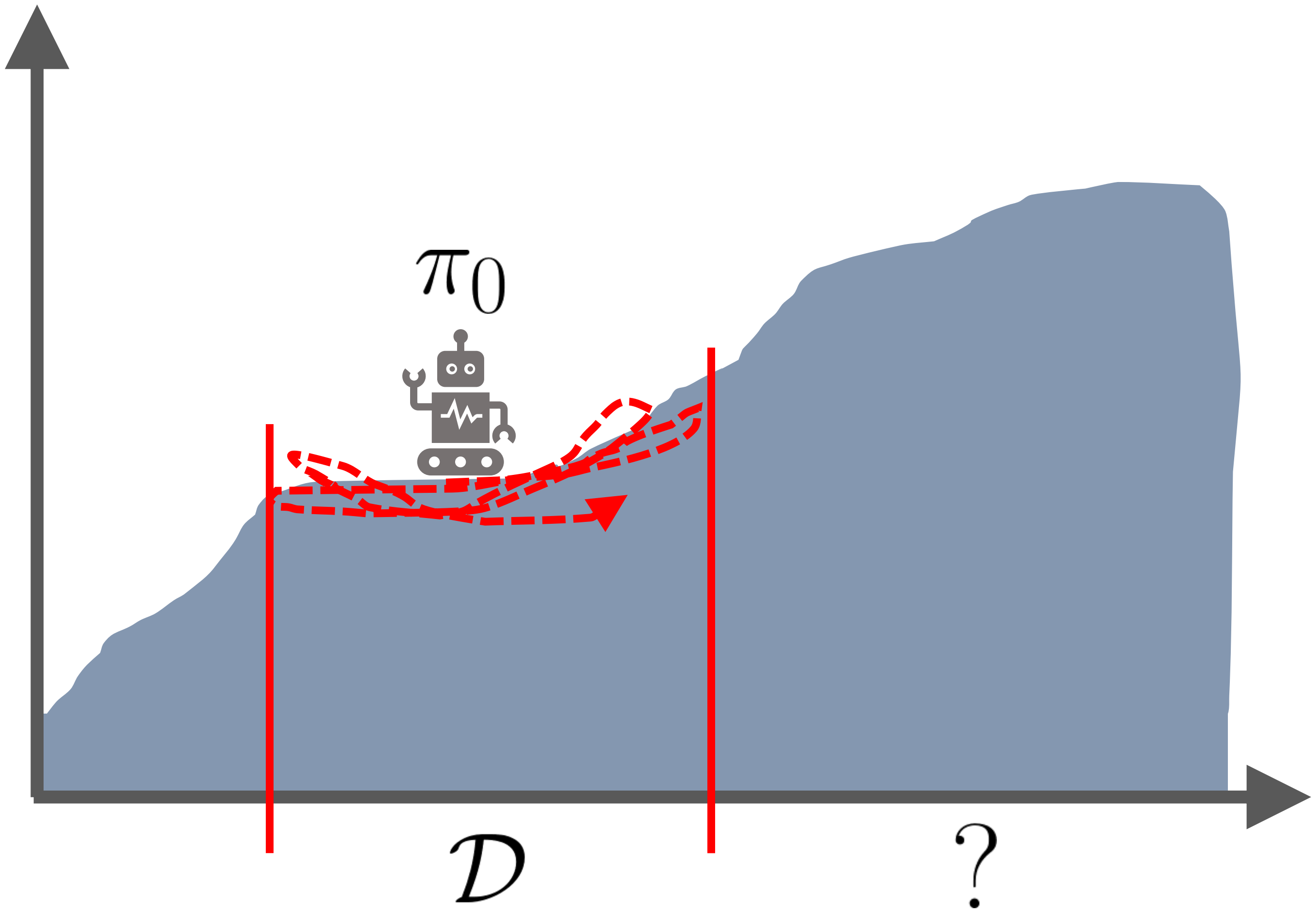
Problem

- Reward: climb as high as possible



Problem

- Reward: climb as high as possible

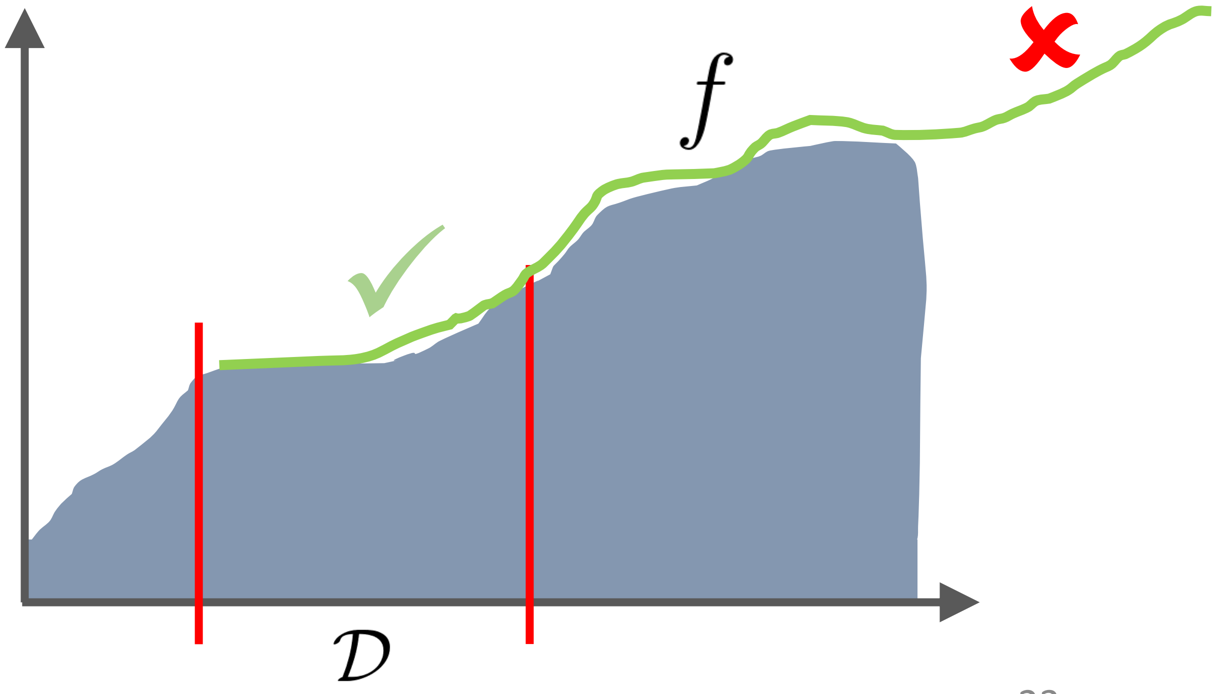


Distribution Shift

- Data distribution is different from the policy's distribution

$$\mathcal{D} \sim p(\mathbf{s}, \mathbf{a}|\pi_0) \neq p(\mathbf{s}, \mathbf{a}|\pi)$$

- Model $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ trained on $\mathcal{D}$
  - Low error under $p(\mathbf{s}, \mathbf{a}|\pi_0)$
  - High error under $p(\mathbf{s}, \mathbf{a}|\pi)$



- Can we make

$$p(\mathbf{s}, \mathbf{a}|\pi_0) = p(\mathbf{s}, \mathbf{a}|\pi)\,?$$

Model-Based RL

- Collect data with a base policy π_0

$$\mathbf{s}_0 \xrightarrow{\mathbf{a}_0} \mathbf{s}_1 \xrightarrow{\mathbf{a}_1} \mathbf{s}_2 \cdots \mathbf{s}_{T-1} \xrightarrow{\mathbf{a}_{T-1}} \mathbf{s}_T$$

- Dataset: $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}')\}$
- Fit a dynamics model via supervised learning

$$\arg\max_{f} \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$$

- Train new policy π by simulating with f

DYNA

ALGORITHM: DYNA

$\pi^0 \leftarrow$ initialize policy
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

for iteration $k = 0, ..., n-1$ **do**
    Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
    Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

    Fit dynamics model:
    $f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

    $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s},\mathbf{a})$
end for

return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} [\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})]$

7: $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
8: **end for**

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} [\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})]$

7: $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
8: **end for**

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$

7: $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
8: **end for**

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

$\pi^0 \leftarrow$ initialize policy
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

for iteration $k = 0, ..., n-1$ **do**
    Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
    Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

    Fit dynamics model:
    $f = \arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

    $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s},\mathbf{a})$
end for

return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}') \sim \mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$

7: $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
8: **end for**

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

$\pi^0 \leftarrow$ initialize policy
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

for iteration $k = 0, ..., n-1$ **do**
    Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
    Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

    Fit dynamics model:
    $f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

    $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s},\mathbf{a})$
end for

return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$

7: $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
8: **end for**

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting [Sutton 1991]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

7: $\boxed{\pi^{k+1} \leftarrow \text{train policy by simulating rollouts with } f(\mathbf{s}'|\mathbf{s},\mathbf{a})}$
8: **end for**

use any RL algorithm
(e.g. policy gradient, Q-learning, SAC, etc.)

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

DYNA

ALGORITHM: DYNA

$\pi^0 \leftarrow$ initialize policy
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

for iteration $k = 0, ..., n-1$ **do**
  Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
  $\boxed{\text{Add transitions to dataset } \mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}}$

keep data from all iterations

  Fit dynamics model:
  $f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

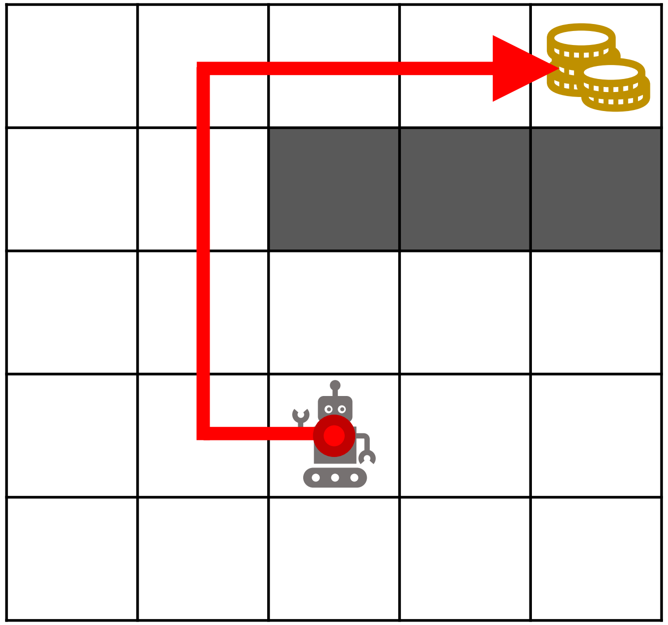
  $\pi^{k+1} \leftarrow$ train policy by simulating rollouts with $f(\mathbf{s}'|\mathbf{s},\mathbf{a})$
end for

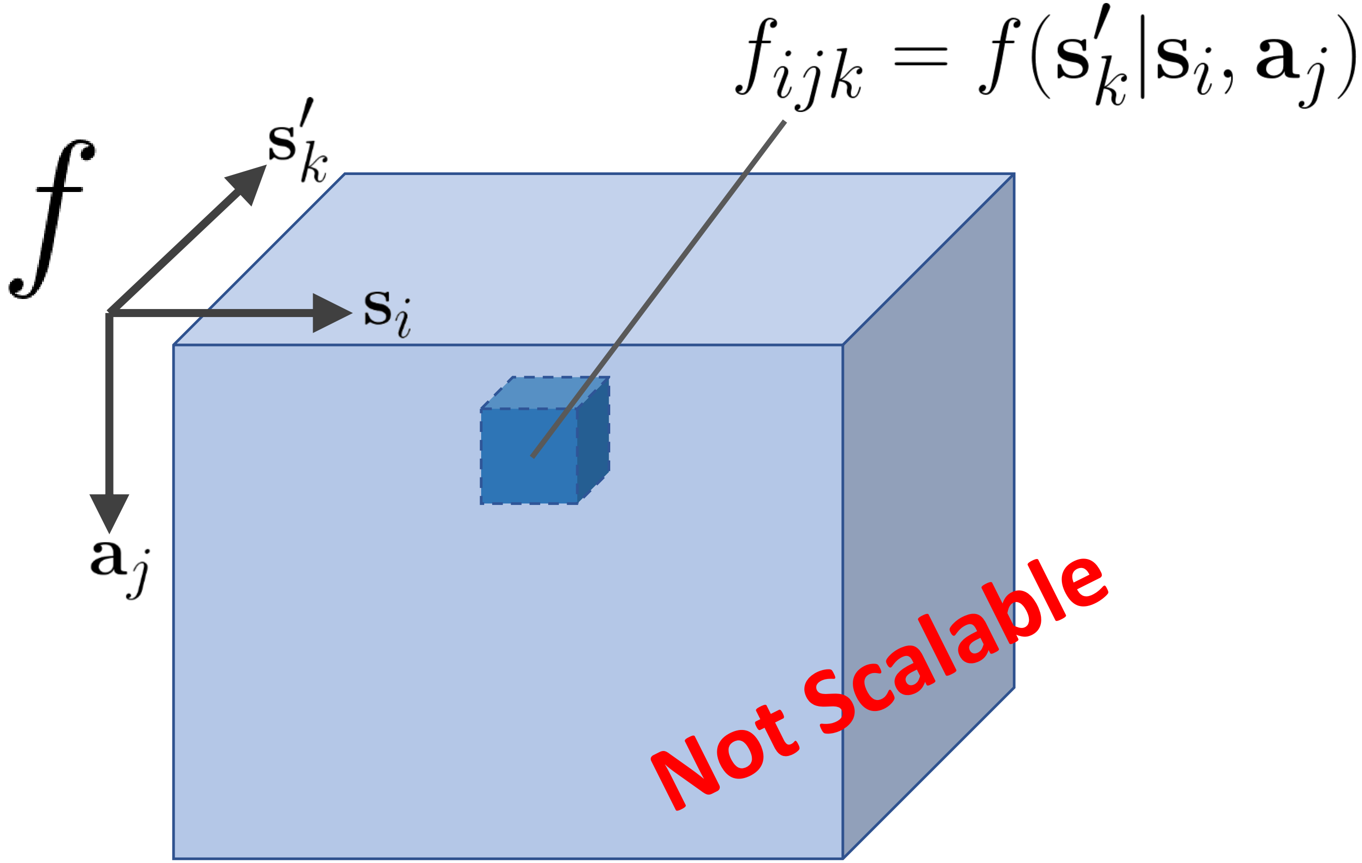
return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting [Sutton 1991]

Model Representation

- How do we represent $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$?
- MDP with small discrete states and actions → lookup table



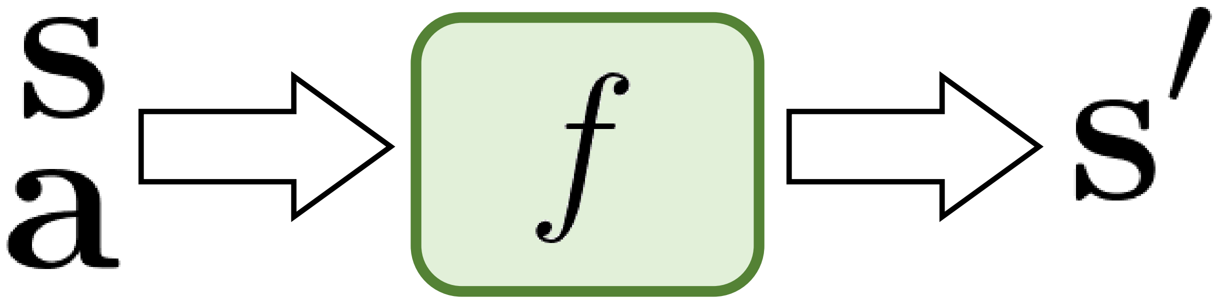


Deterministic Models

- How do we represent $f(\mathbf{s}'|\mathbf{s},\mathbf{a})$?

$$\arg\min_f \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[||\mathbf{s}' - f(\mathbf{s},\mathbf{a})||^2\right]$$

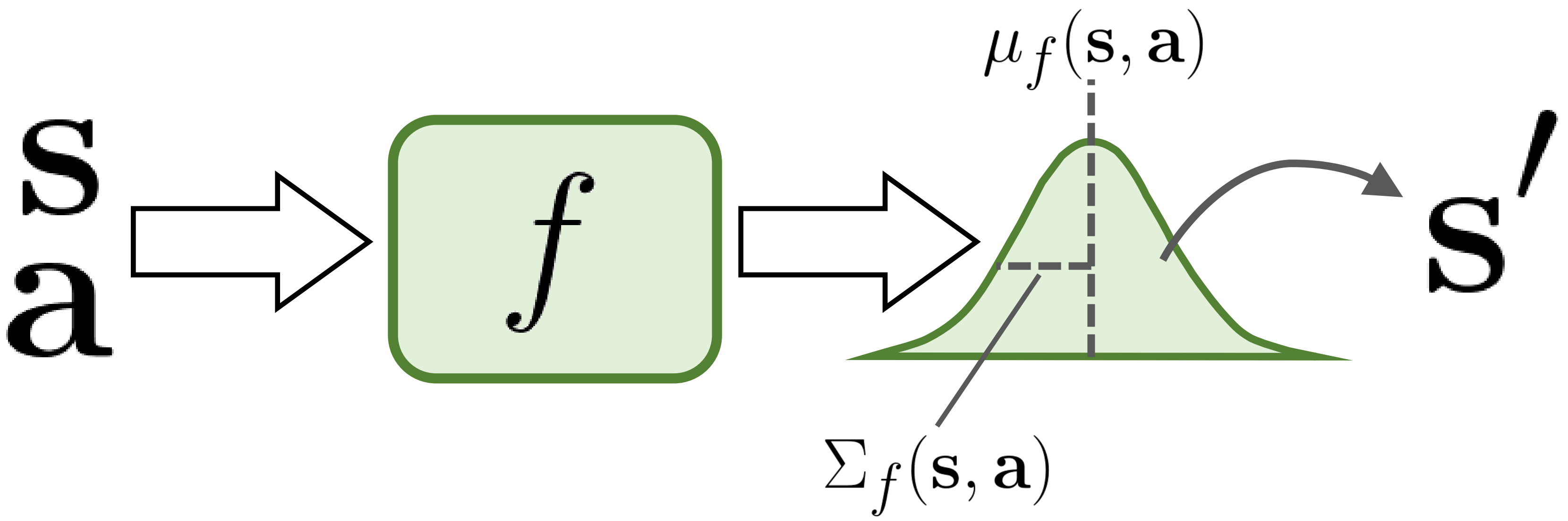
What if the dynamics are stochastic?



Stochastic Models

- How do we represent $f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$?

$$\arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s}, \mathbf{a})\right]$$



Stochastic Models

- How do we represent $f(\mathbf{s}'|\mathbf{s},\mathbf{a})$?

$$\arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\log \underbrace{f(\mathbf{s}'|\mathbf{s},\mathbf{a})}_{\text{Conditional Generative Model}}\right]$$

Conditional Generative Model

- Variational Autoencoders (VAEs)
- Generative Adversarial Networks (GANs)
- Flow Models
- Diffusion Models
- Etc.

Reward Model

- If reward function is unknown, augment model to predict both states and rewards
- For most tasks, reward function is available/specified by a human

dynamics model

$$f(\mathbf{s}'|\mathbf{s}, \mathbf{a})$$

reward model

$$h(r|\mathbf{s}, \mathbf{a}, \mathbf{s}')$$

Model-Based Rollout

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \; \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} \left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

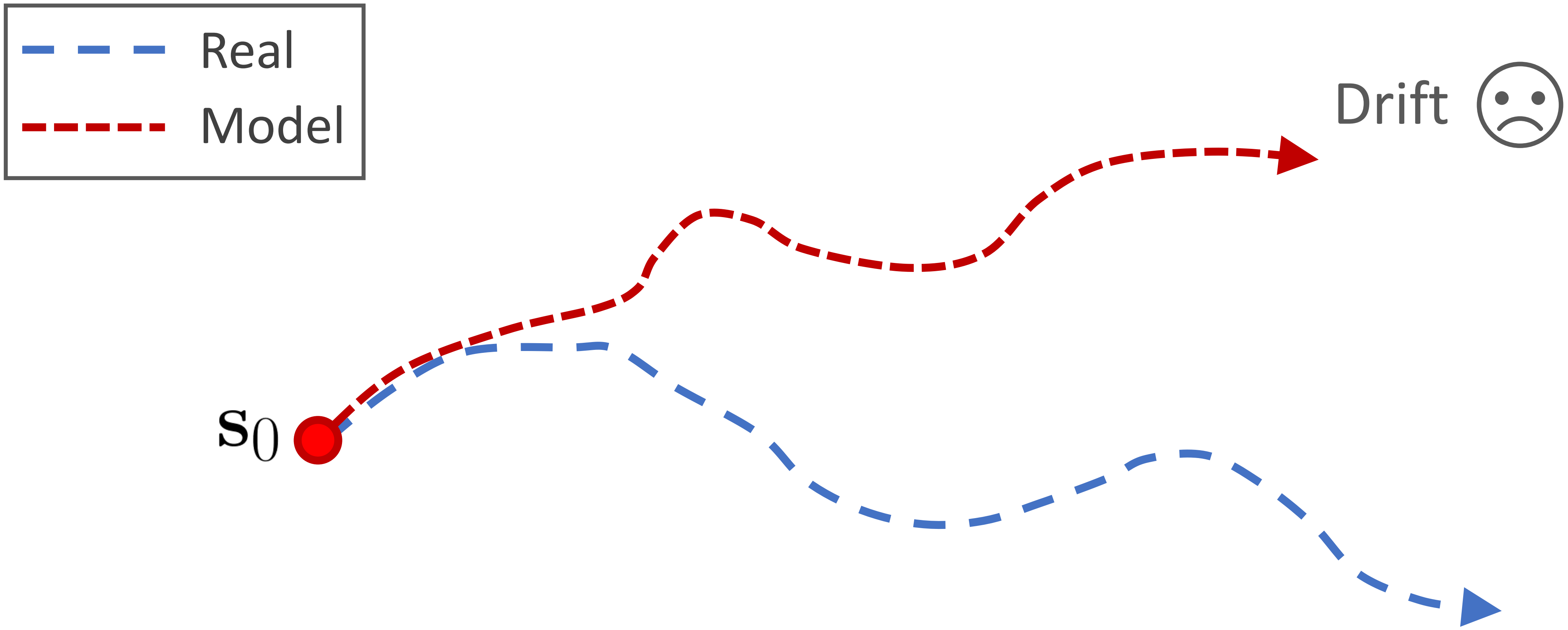
7: $\boxed{\pi^{k+1} \leftarrow \text{train policy by simulating rollouts with } f(\mathbf{s}'|\mathbf{s},\mathbf{a})}$
8: **end for**

9: return π^n

generate trajectories
with model

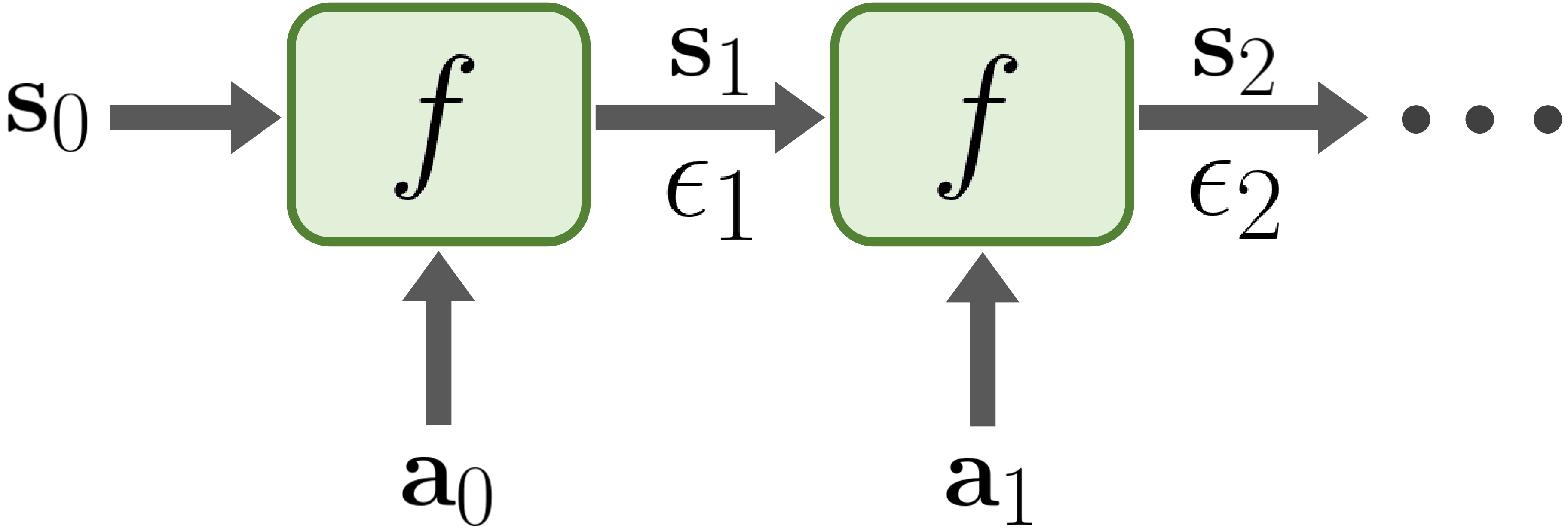
Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

Model-Based Rollout



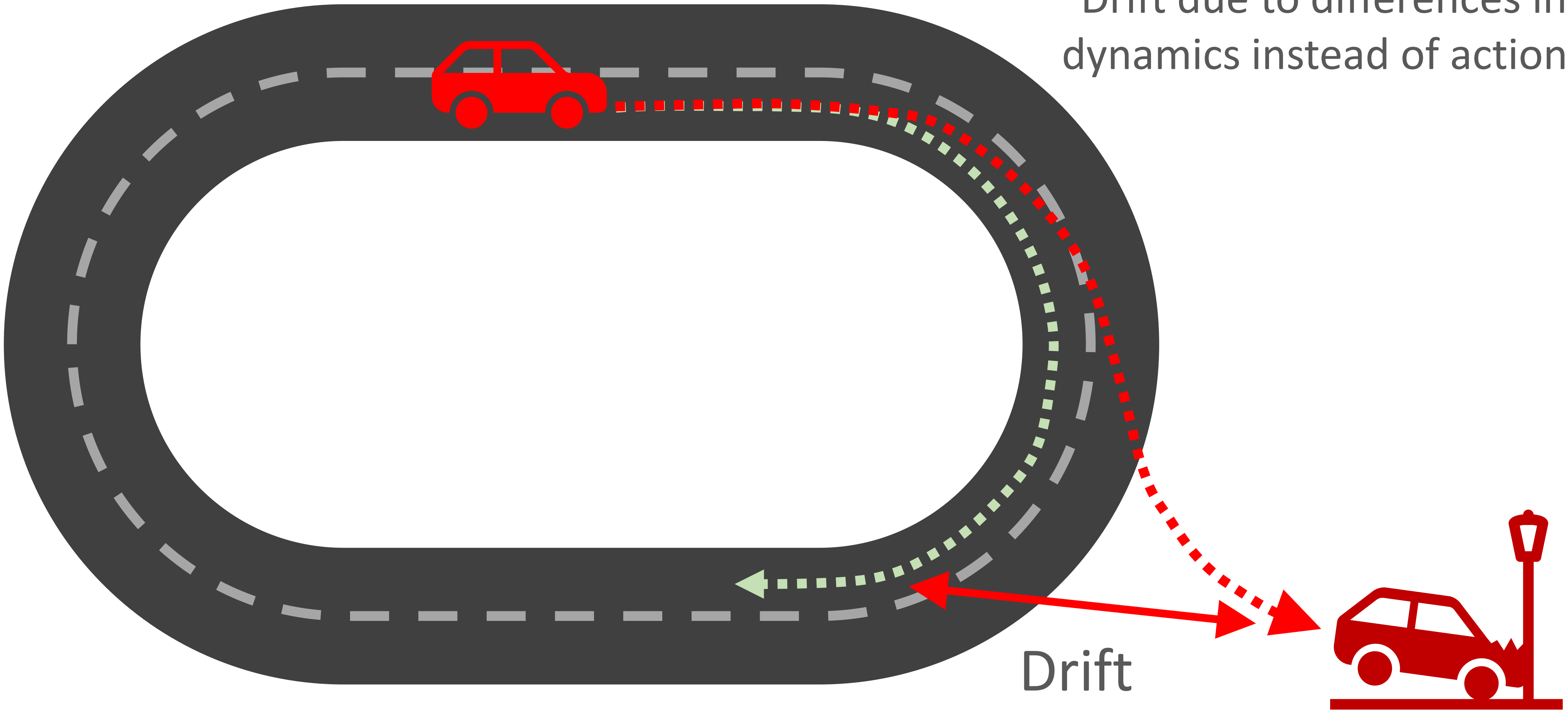
Drift

- Same action sequence in the real env and the model can lead to very different trajectories
- Autoregressive model → compounding error

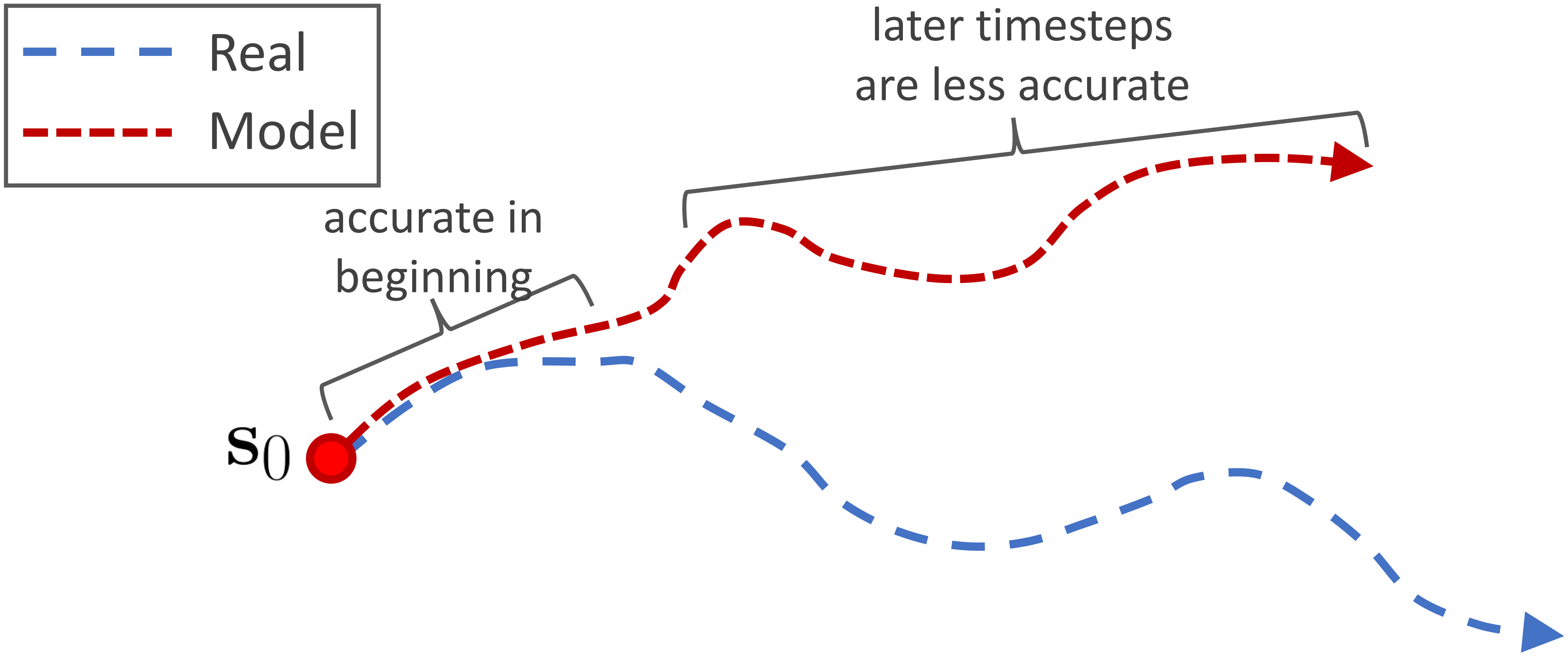


Drift

Drift due to differences in dynamics instead of actions

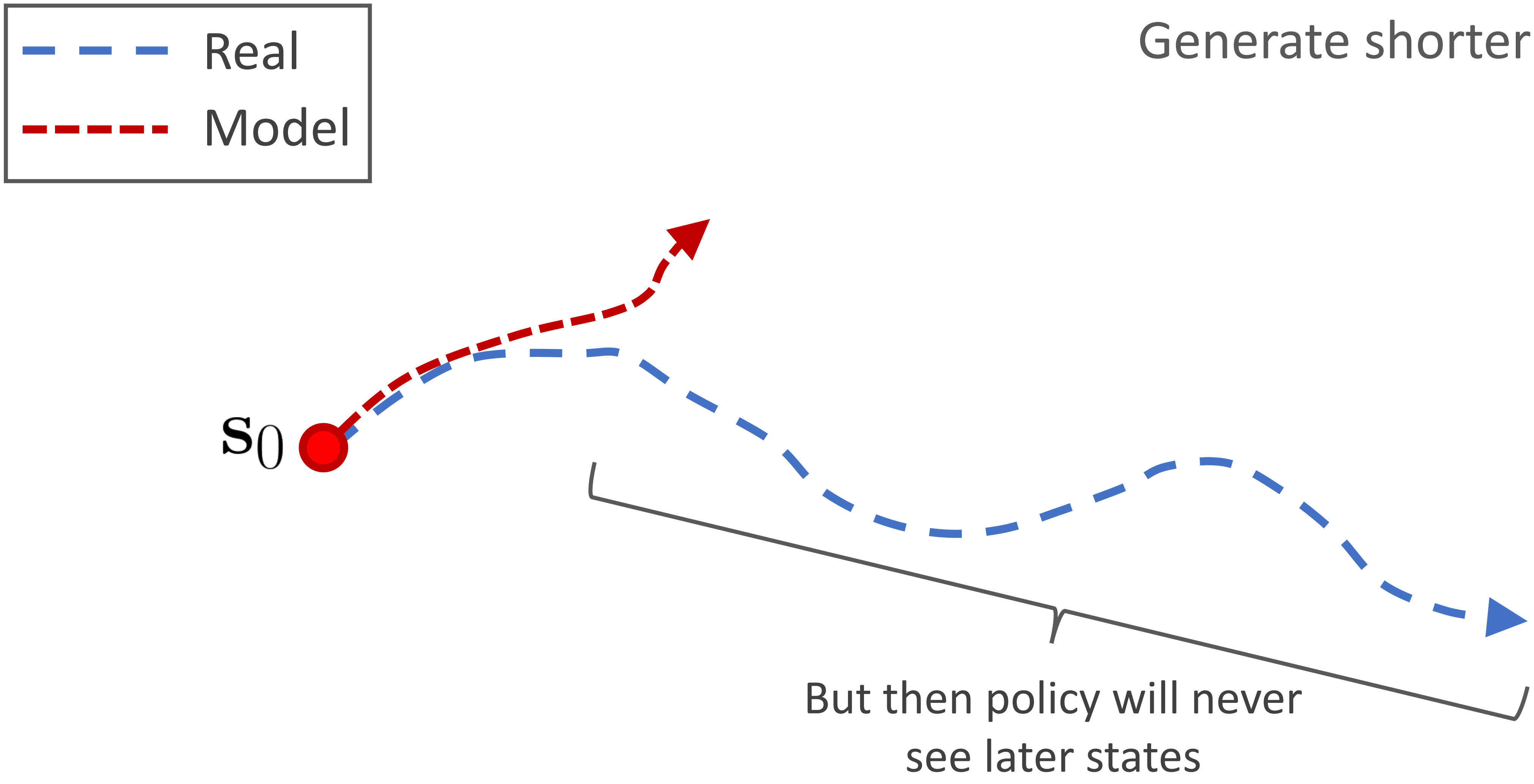


Model-Based Rollout

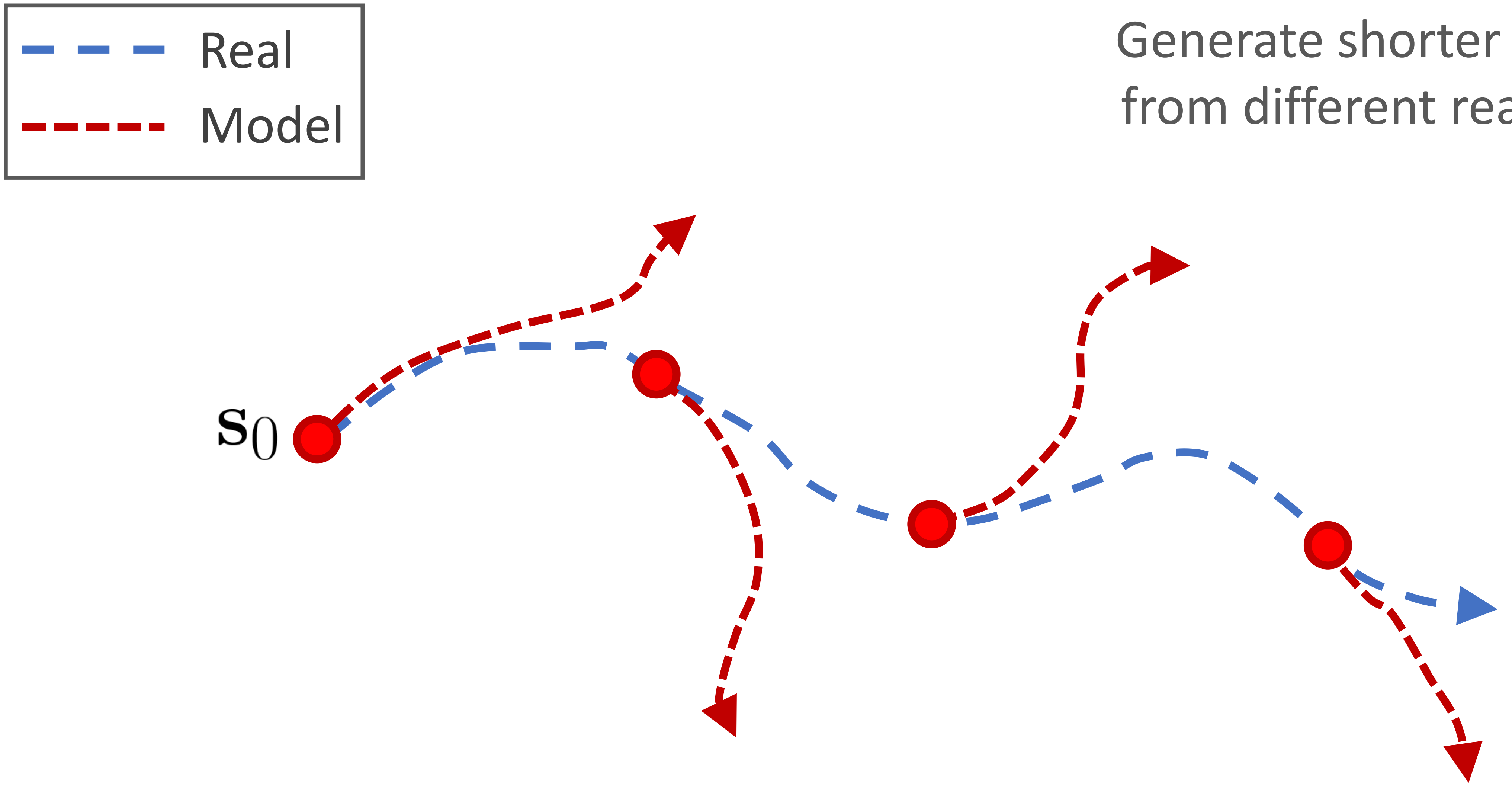


Model-Based Rollout

Generate shorter rollouts

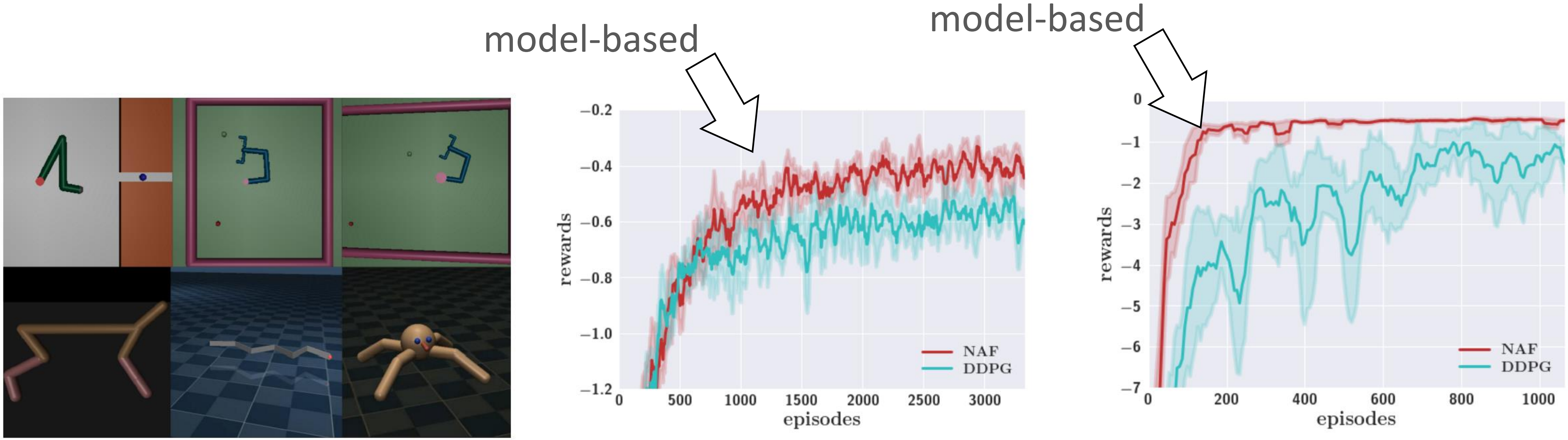


Model-Based Rollout



Generate shorter rollouts from different real states

Model-Based RL



(a) Example task domains. (b) NAF and DDPG on multi-target reacher. (c) NAF and DDPG on peg insertion.

Continuous Deep Q-Learning with Model-based Acceleration
[Gu et al. 2016]

DYNA

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$

7: $\boxed{\pi^{k+1} \leftarrow \text{train policy by simulating rollouts with } f(\mathbf{s}'|\mathbf{s},\mathbf{a})}$
8: **end for**

use any RL algorithm
(e.g. policy gradient, Q-learning, SAC, etc.)

9: return π^n

Dyna, an Integrated Architecture for Learning, Planning, and Reacting
[Sutton 1991]

Differentiable Dynamics

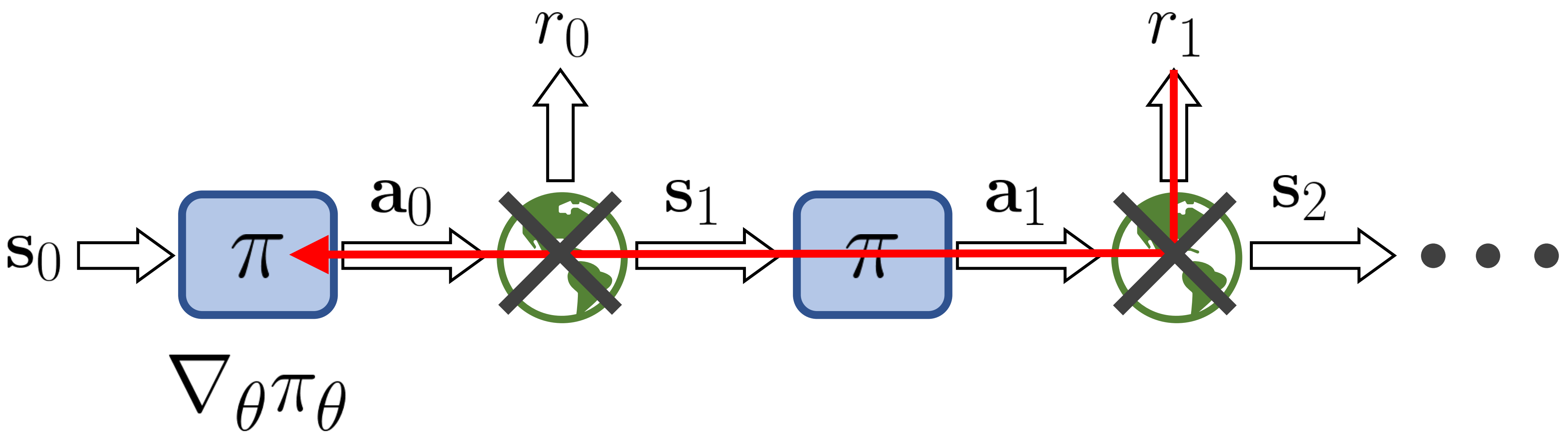
$$\arg\max_{\pi} \; \underline{\mathbb{E}_{\tau \sim p(\tau|\pi)}} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

$$\Downarrow$$

$$\arg\max_{\pi} \; \underline{\mathbb{E}_{\tau \sim f(\tau|\pi)}} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

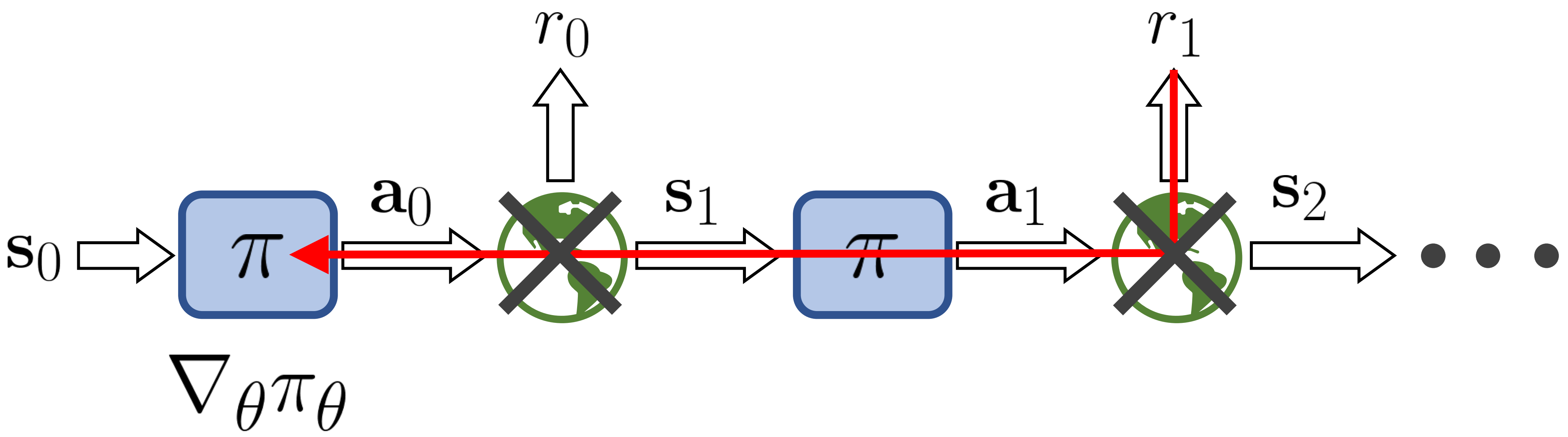
use any RL algorithm
(e.g. policy gradient, Q-learning, SAC, etc.)

Differentiable Dynamics



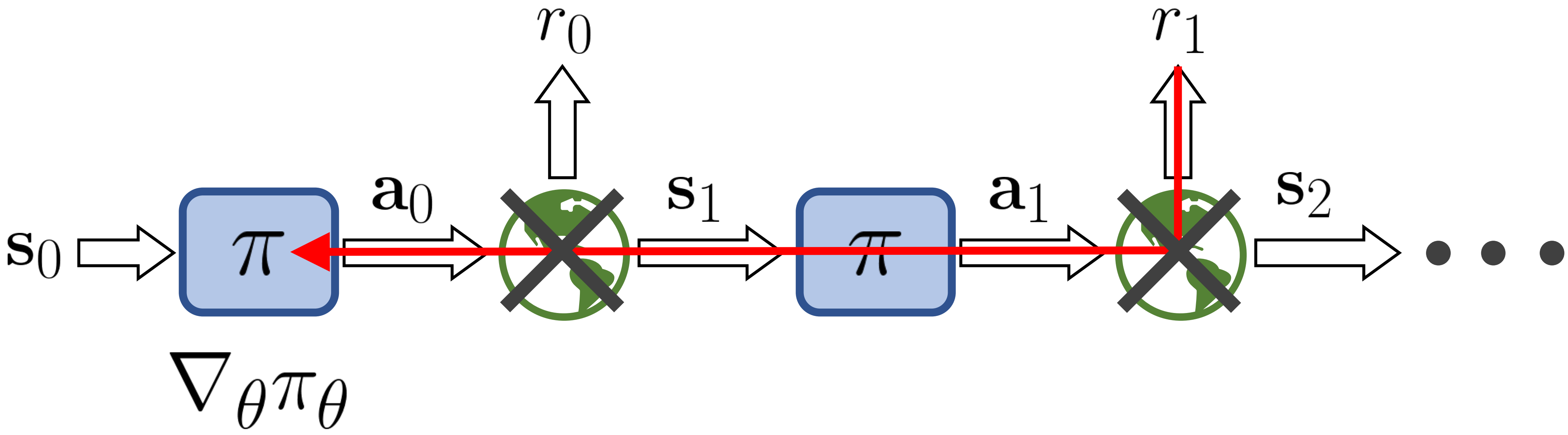
$$\frac{\partial r_1}{\partial \theta} = \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \theta} + \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \mathbf{s}_1}\frac{\partial \mathbf{s}_1}{\partial \mathbf{a}_0}\frac{\partial \mathbf{a}_0}{\partial \theta} + \dots$$

Differentiable Dynamics



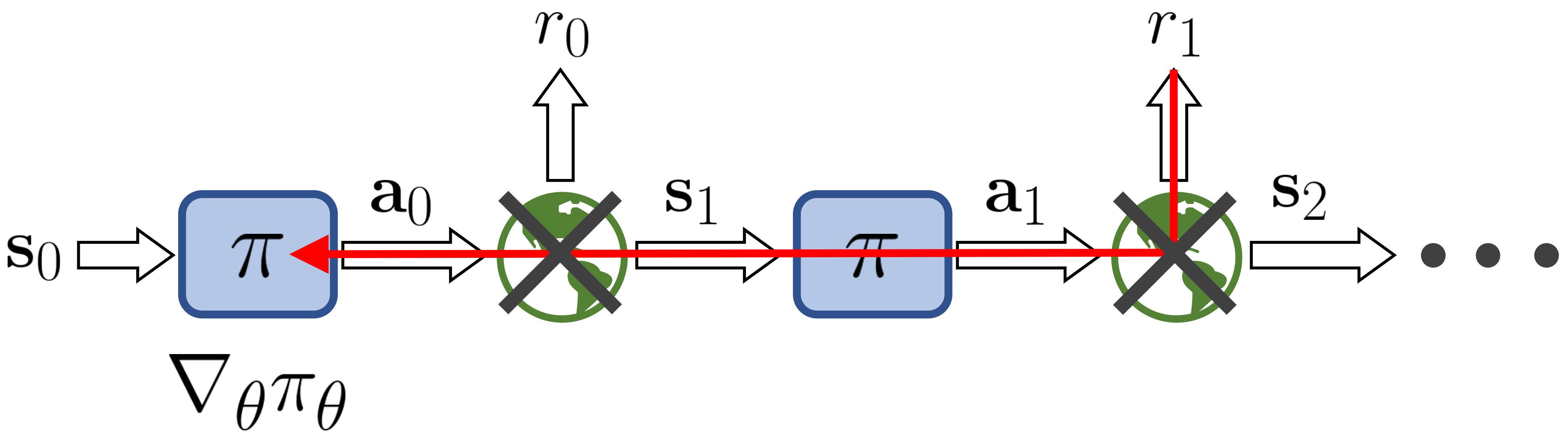
$$\frac{\partial r_1}{\partial \theta} = \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \theta} + \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \mathbf{s}_1}\frac{\partial \mathbf{s}_1}{\partial \mathbf{a}_0}\frac{\partial \mathbf{a}_0}{\partial \theta} + \ldots$$

Differentiable Dynamics



$$\frac{\partial r_1}{\partial \theta} = \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \theta} + \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \mathbf{s}_1}\underbrace{\boxed{\frac{\partial \mathbf{s}_1}{\partial \mathbf{a}_0}}}_{\text{Dynamics}}\frac{\partial \mathbf{a}_0}{\partial \theta} + \ldots$$

Differentiable Dynamics



Fully Differentiable!

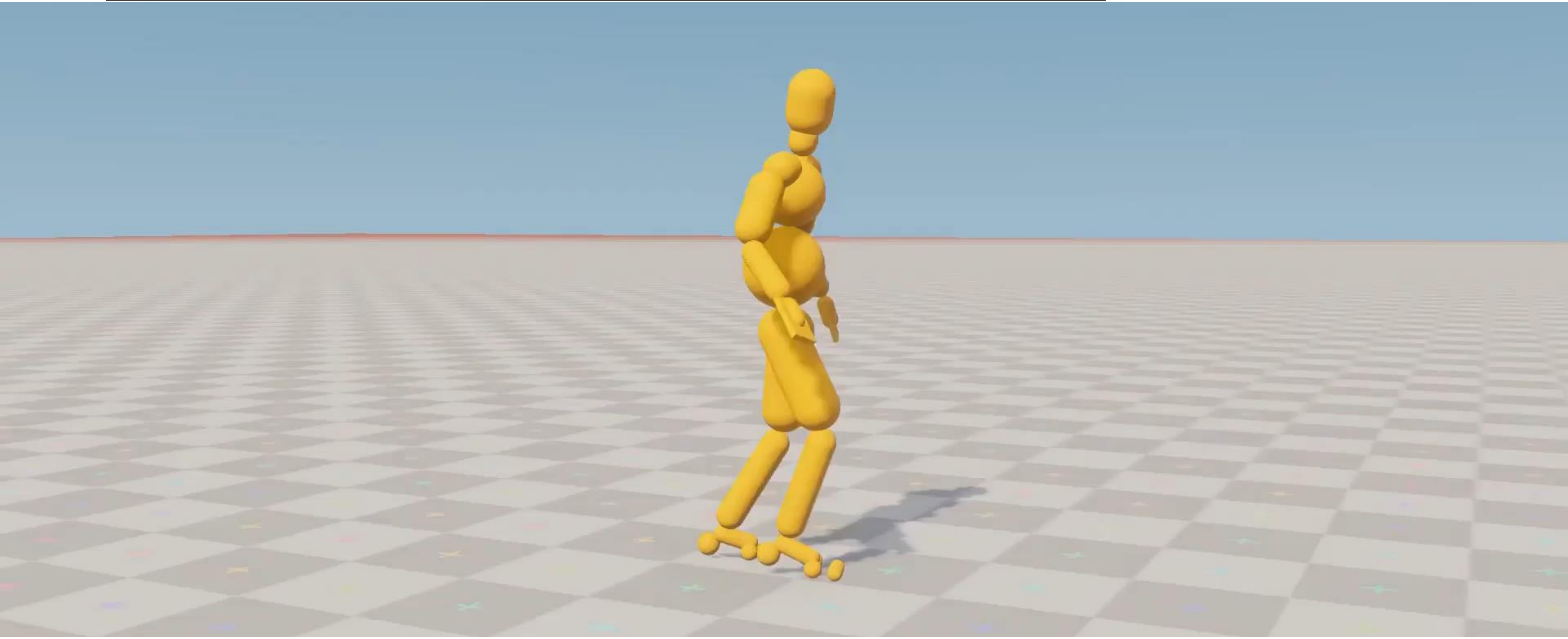
$$\frac{\partial r_1}{\partial \theta} = \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \theta} + \frac{\partial r_1}{\partial \mathbf{a}_1}\frac{\partial \mathbf{a}_1}{\partial \mathbf{s}_1}\frac{\partial \mathbf{s}_1}{\partial \mathbf{a}_0}\frac{\partial \mathbf{a}_0}{\partial \theta} + \ldots$$

Differentiable Dynamics

$$\arg\max_{\pi} \ \mathbb{E}_{\tau \sim f(\tau|\pi)} \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$$

Compute gradients using autodiff
and solve with gradient ascent

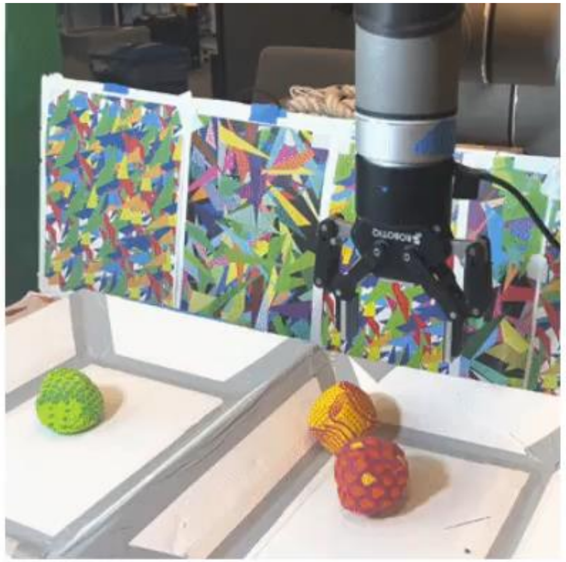
Differentiable Dynamics



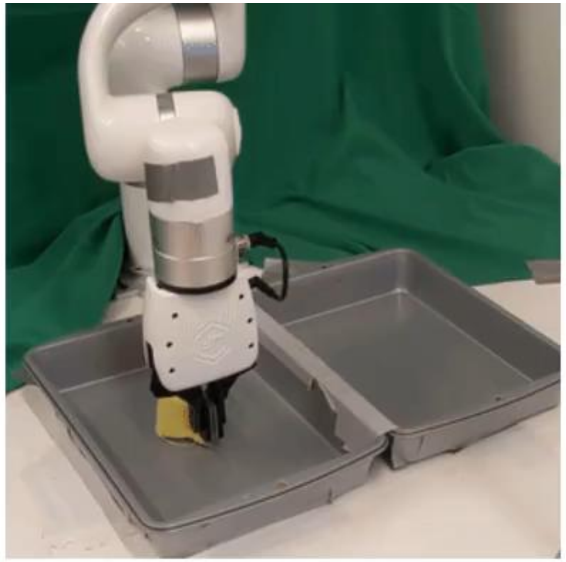
SuperTrack: Motion Tracking for Physically Simulated Characters Using Supervised Learning
[Fussell et al. 2021]

Differentiable Dynamics

A1 Quadruped Walking



UR5 Multi-Object Visual Pick Place



XArm Visual Pick and Place

Sphero Ollie Visual Navigation

DayDreamer: World Models for Physical Robot Learning
[Wu et al. 2022]

Differentiable Dynamics

DayDreamer: World Models for Physical Robot Learning
[Wu et al. 2022]

Differentiable Dynamics

DayDreamer: World Models for Physical Robot Learning
[Wu et al. 2022]

Model Exploitation

ALGORITHM: DYNA

1: $\pi^0 \leftarrow$ initialize policy
2: $\mathcal{D} \leftarrow \{\emptyset\}$ initialize dataset

3: **for** iteration $k = 0, ..., n-1$ **do**
4: Sample trajectory τ according to $\pi^k(\mathbf{a}|\mathbf{s})$
5: Add transitions to dataset $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{s}'_i)\}$

6: Fit dynamics model:
$f = \arg\max_f \ \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}} [\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})]$

7: $\boxed{\pi^{k+1} \leftarrow \text{train policy by simulating rollouts with } f(\mathbf{s}'|\mathbf{s},\mathbf{a})}$
8: **end for**

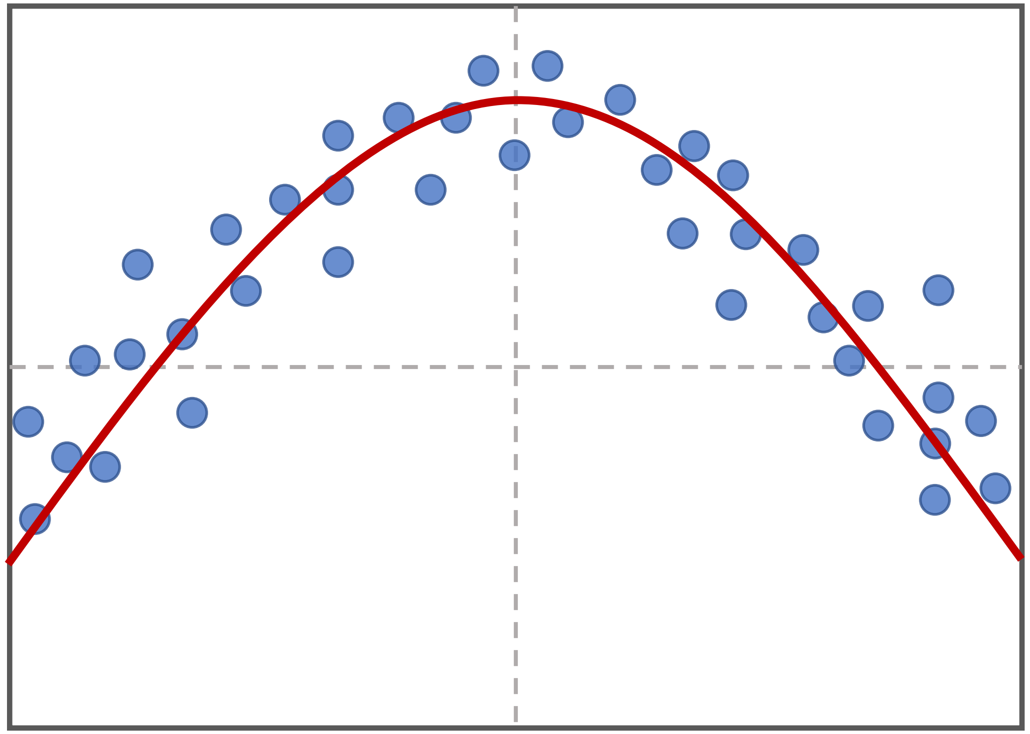
9: return π^n

Policy can exploit errors in model

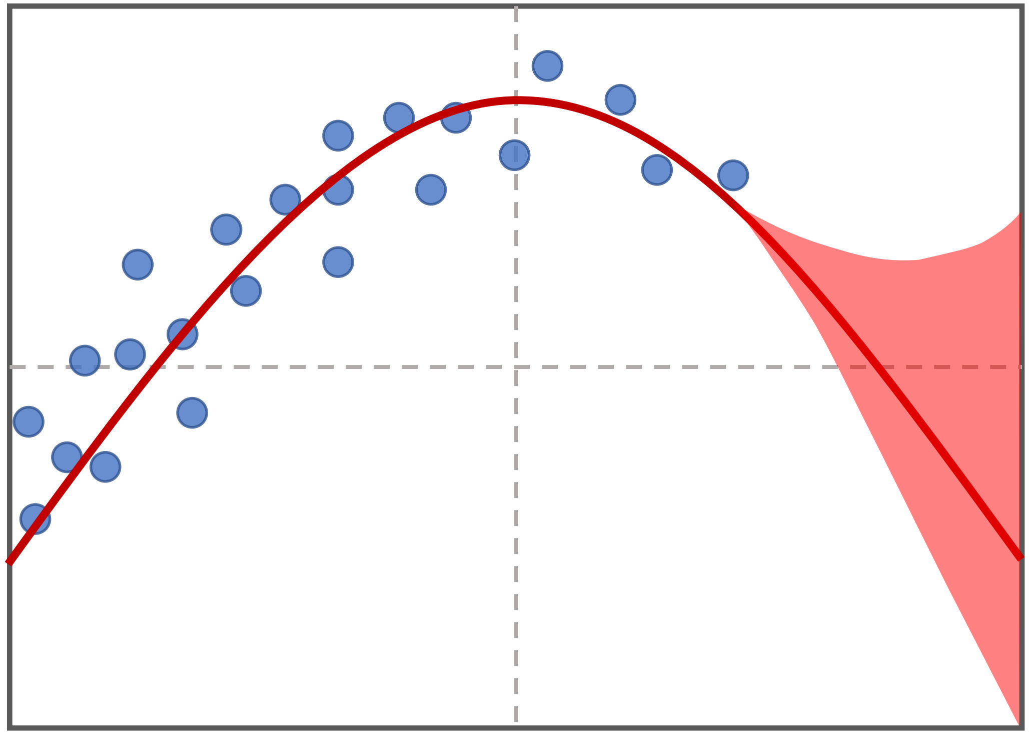
Dyna, an Integrated Architecture for Learning, Planning, and Reacting [Sutton 1991]

2 Types of Uncertainty

Aleatoric
(Statistical Uncertainty)

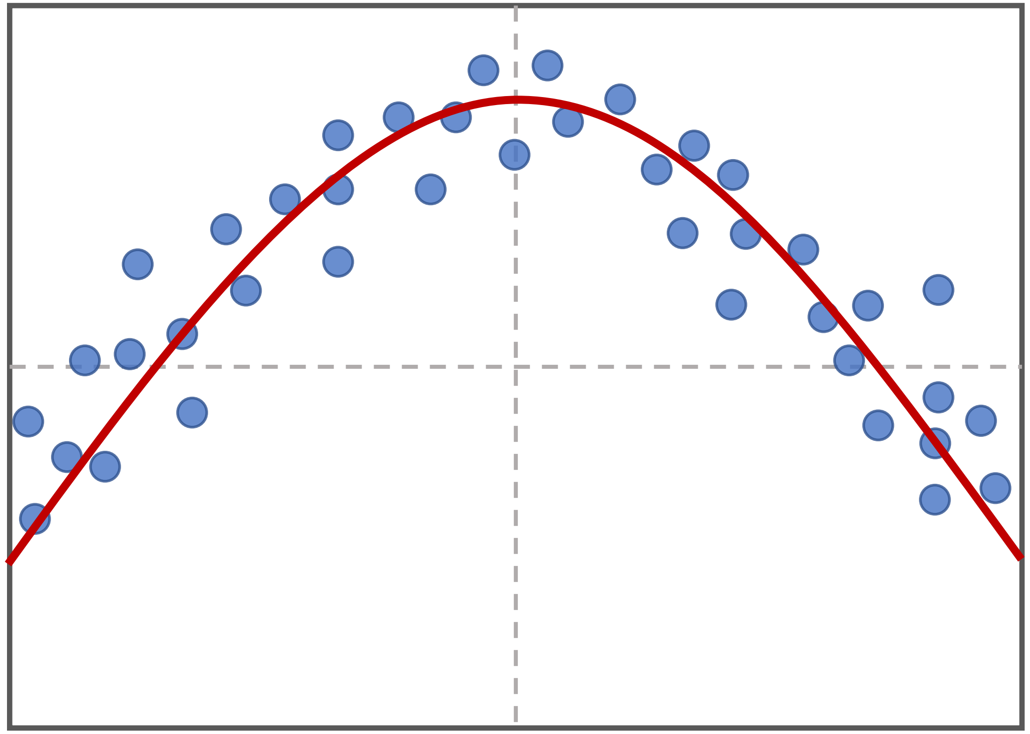


Epistemic
(Model Uncertainty)



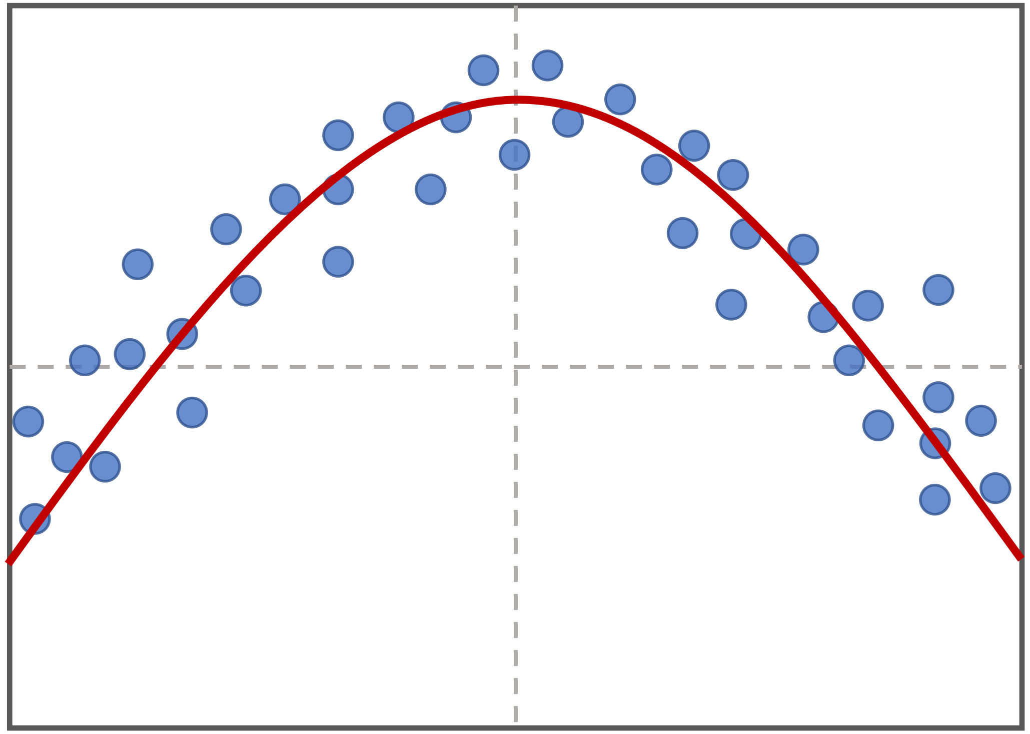
2 Types of Uncertainty

Aleatoric
(Statistical Uncertainty)

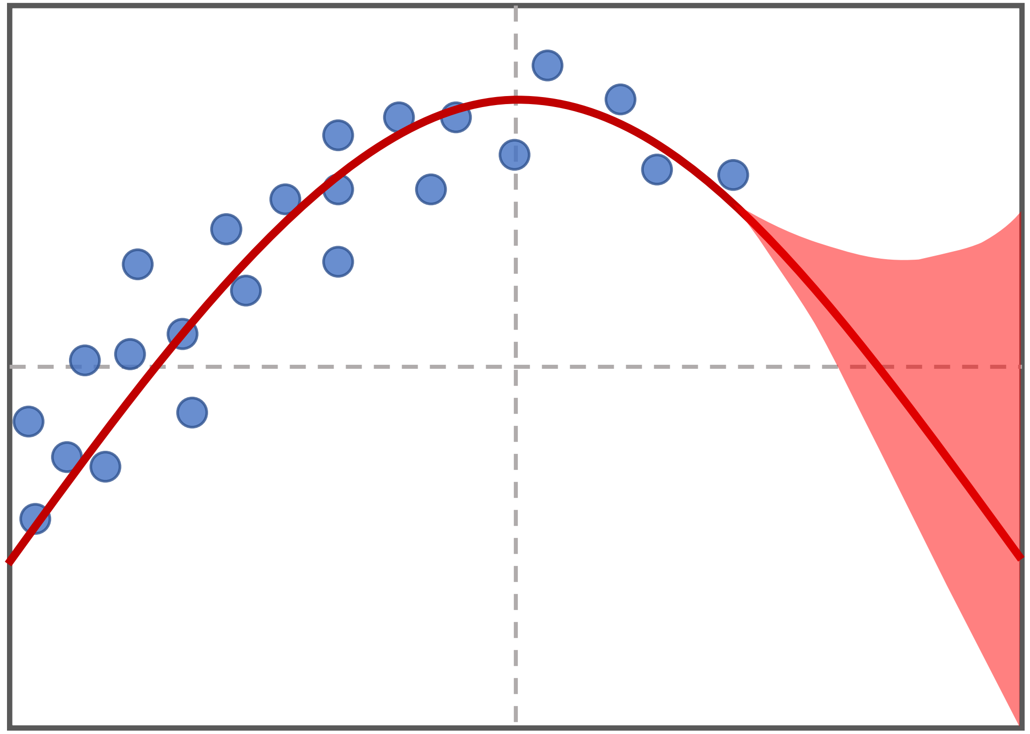


2 Types of Uncertainty

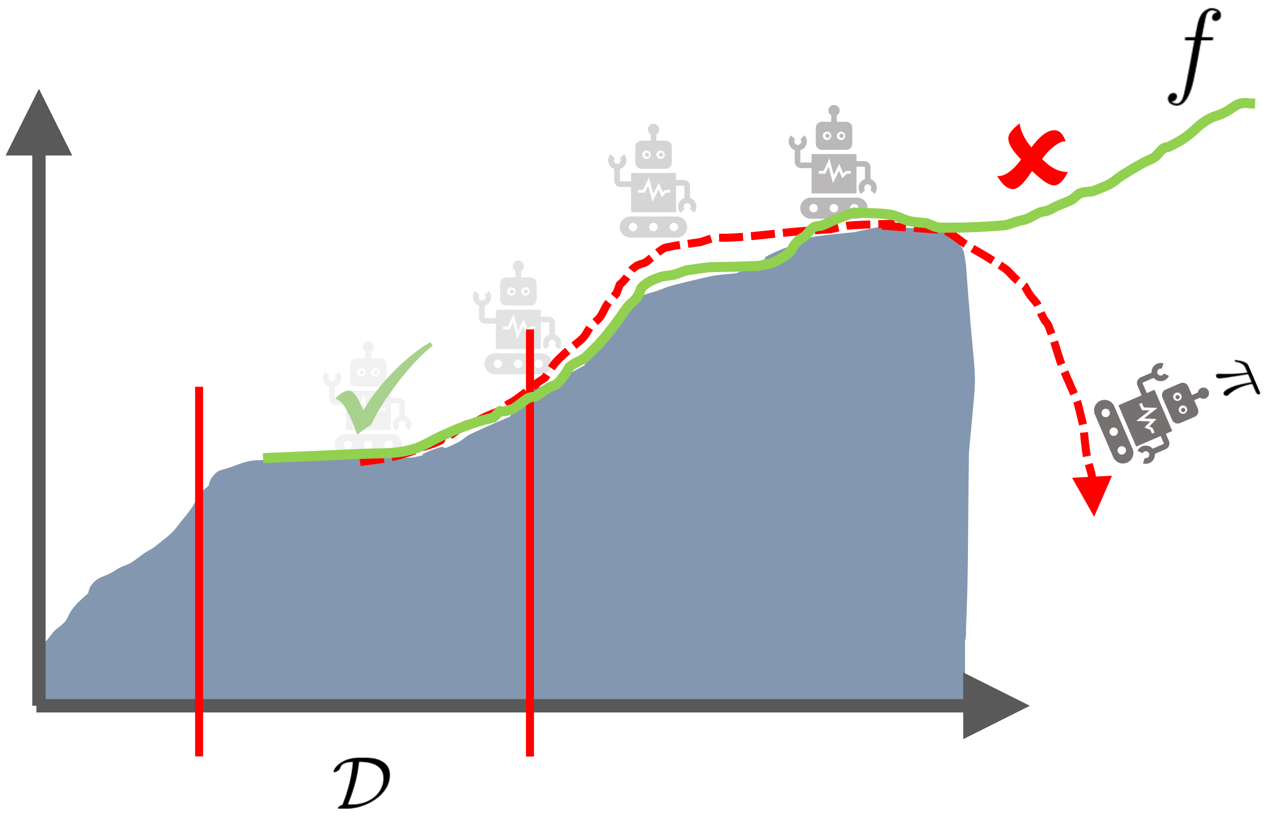
Aleatoric
(Statistical Uncertainty)



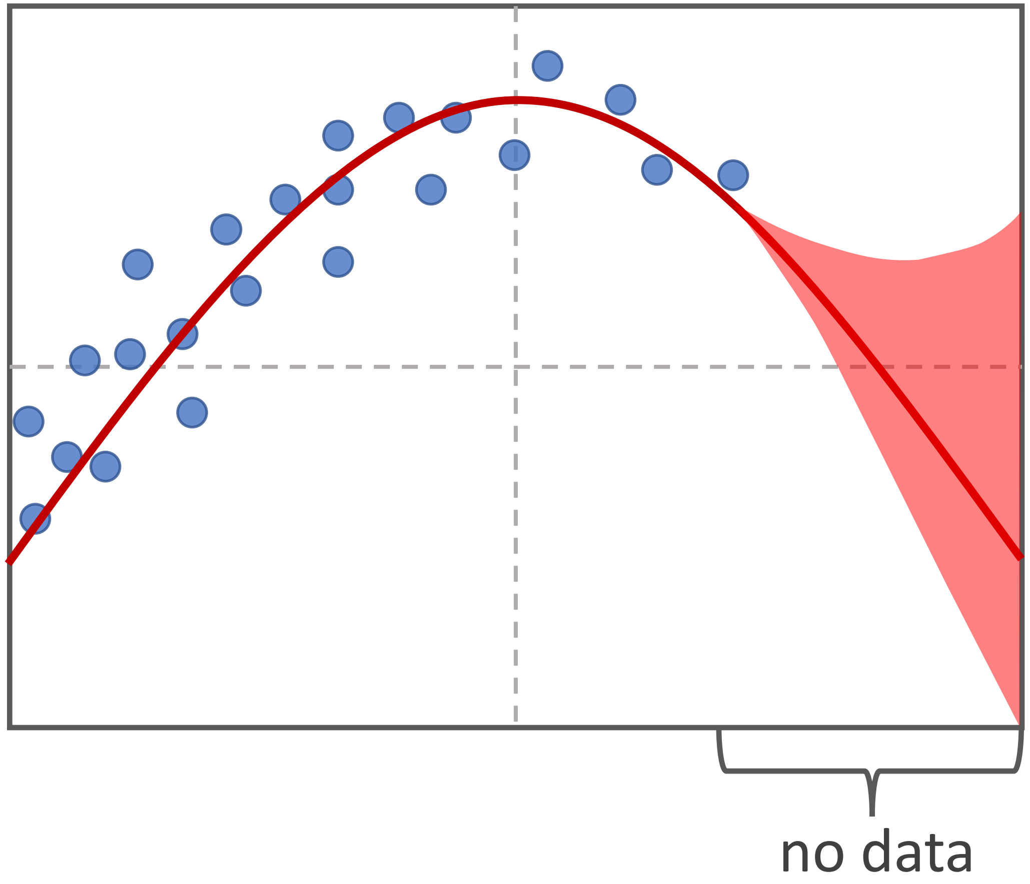
Epistemic
(Model Uncertainty)



2 Types of Uncertainty

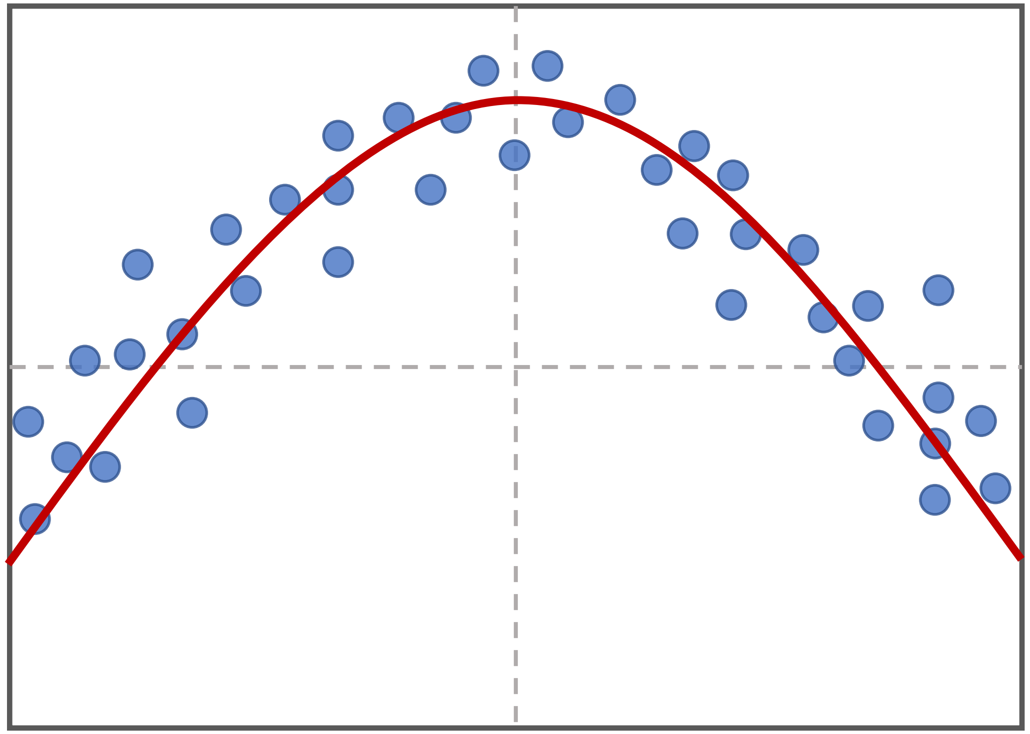


Epistemic
(Model Uncertainty)

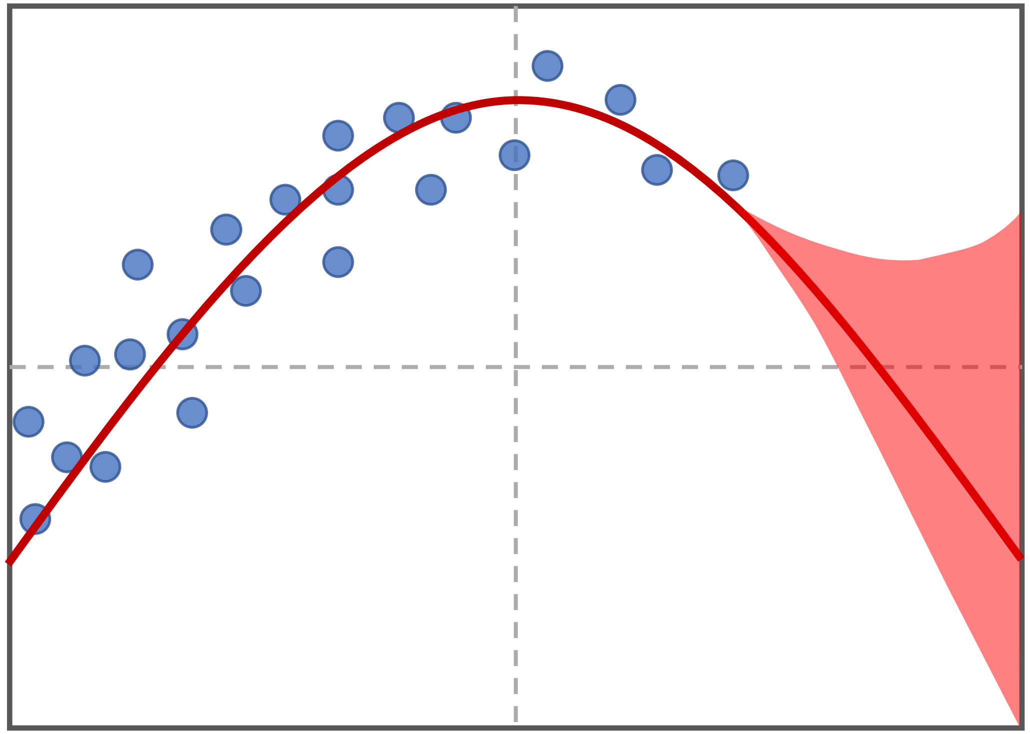


2 Types of Uncertainty

Aleatoric
(Statistical Uncertainty)



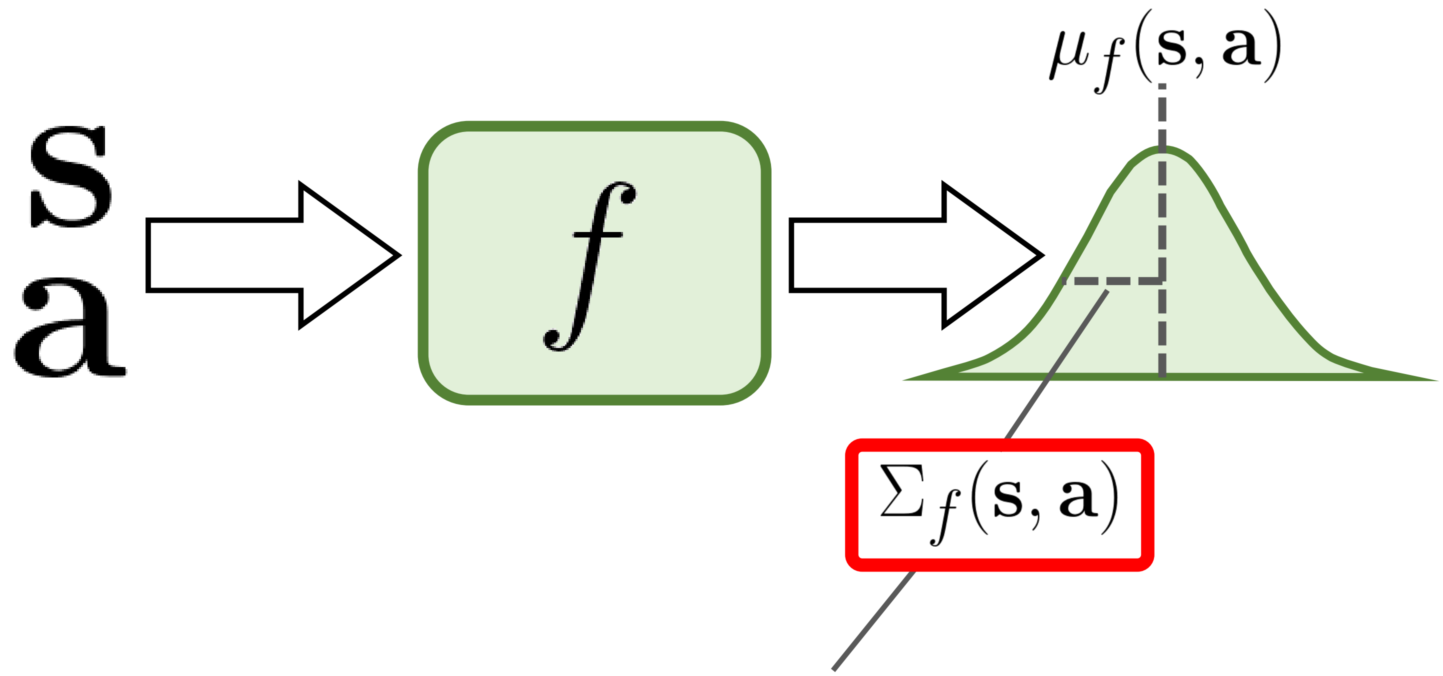
Epistemic
(Model Uncertainty)



Policy can exploit model uncertainty

Uncertainty Estimation

- Can we estimate the model uncertainty?



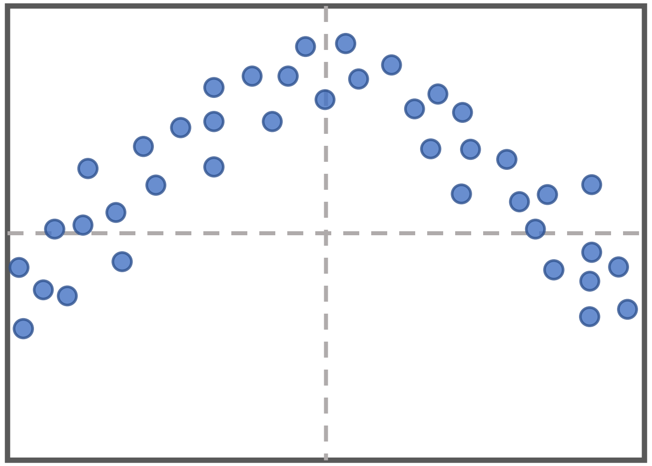
This only estimates *statistical* uncertainty

Uncertainty Estimation

- Can we estimate the model uncertainty?
- Bayesian inference:

$$\underline{p(f|\mathcal{D})}$$

What is the likelihood of
a function given the data

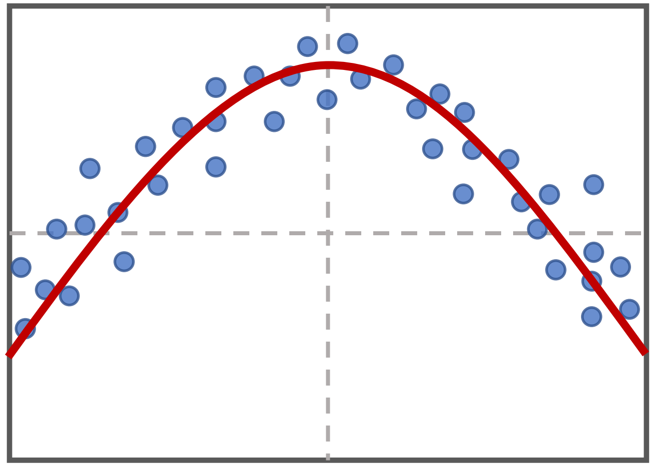


Uncertainty Estimation

- Can we estimate the model uncertainty?
- Bayesian inference:

$$\underline{p(f|\mathcal{D})}$$

What is the likelihood of
a function given the data



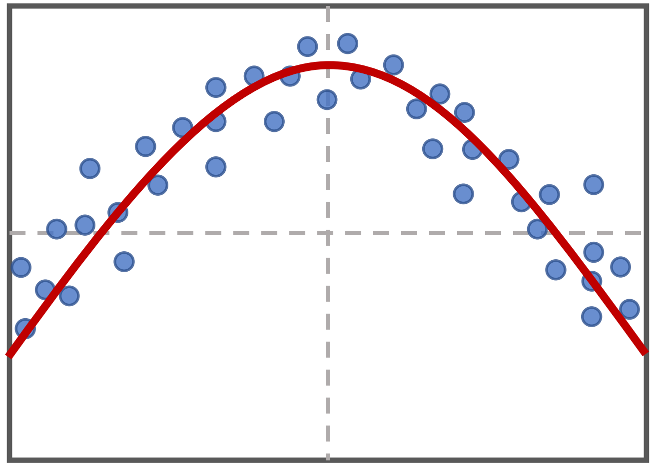
High Likelihood

Uncertainty Estimation

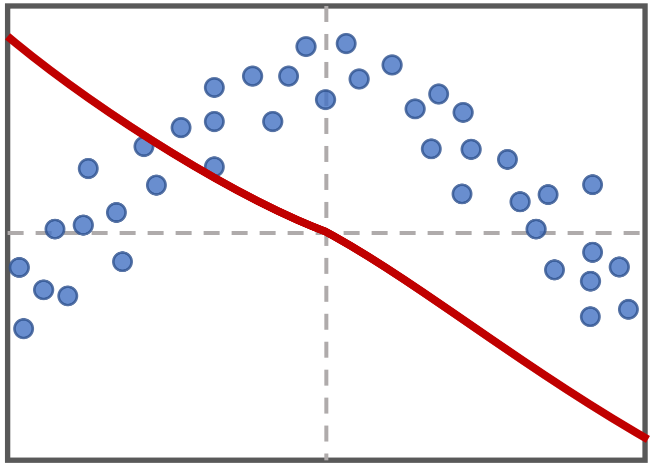
- Can we estimate the model uncertainty?
- Bayesian inference:

$$\underline{p(f|\mathcal{D})}$$

What is the likelihood of
a function given the data



High Likelihood



Low Likelihood

Uncertainty Estimation

- Can we estimate the model uncertainty?
- Bayesian inference:

$$p(f|\mathcal{D}) = \frac{p(f, \mathcal{D})}{p(\mathcal{D})}$$

Uncertainty Estimation

- Can we estimate the model uncertainty?
- Bayesian inference:

$$p(f|\mathcal{D}) = \frac{p(f,\mathcal{D})}{p(\mathcal{D})}$$

$$= \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

Supervised Learning

$$\arg\max_{f}\ \log p(f|\mathcal{D}) = \arg\max_{f}\ \log\frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

Supervised Learning

$$\arg\max_f \underbrace{\log p(f|\mathcal{D})}_{\text{Posterior}} = \arg\max_f \log \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

$$= \arg\max_f \underbrace{\log p(\mathcal{D}|f)}_{\text{Likelihood}} + \underbrace{\log p(f)}_{\text{Prior}} - \underbrace{\log p(\mathcal{D})}_{\text{Constant}}$$

Supervised Learning

$$\arg\max_f \underline{\log p(f|\mathcal{D})} = \arg\max_f \log \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$

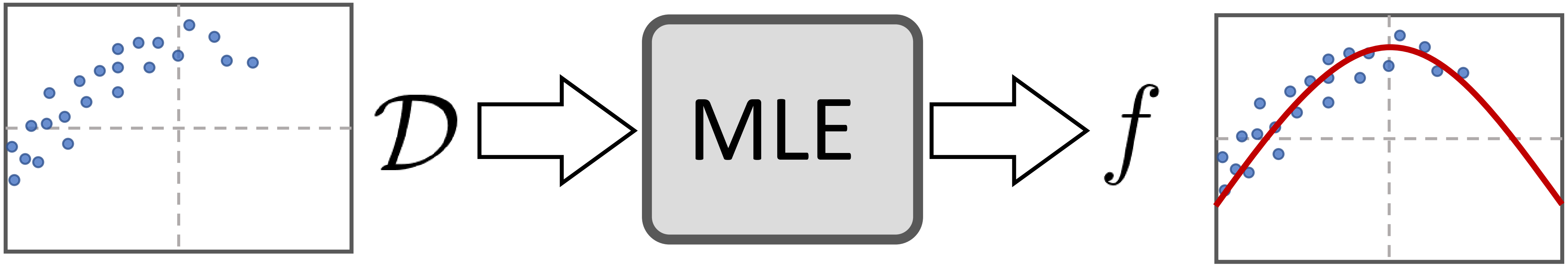
$$= \arg\max_f \underbrace{\log p(\mathcal{D}|f)}_{\text{Likelihood}} + \underline{\log p(f)} - \underline{\log p(\mathcal{D})}$$

$\Downarrow$

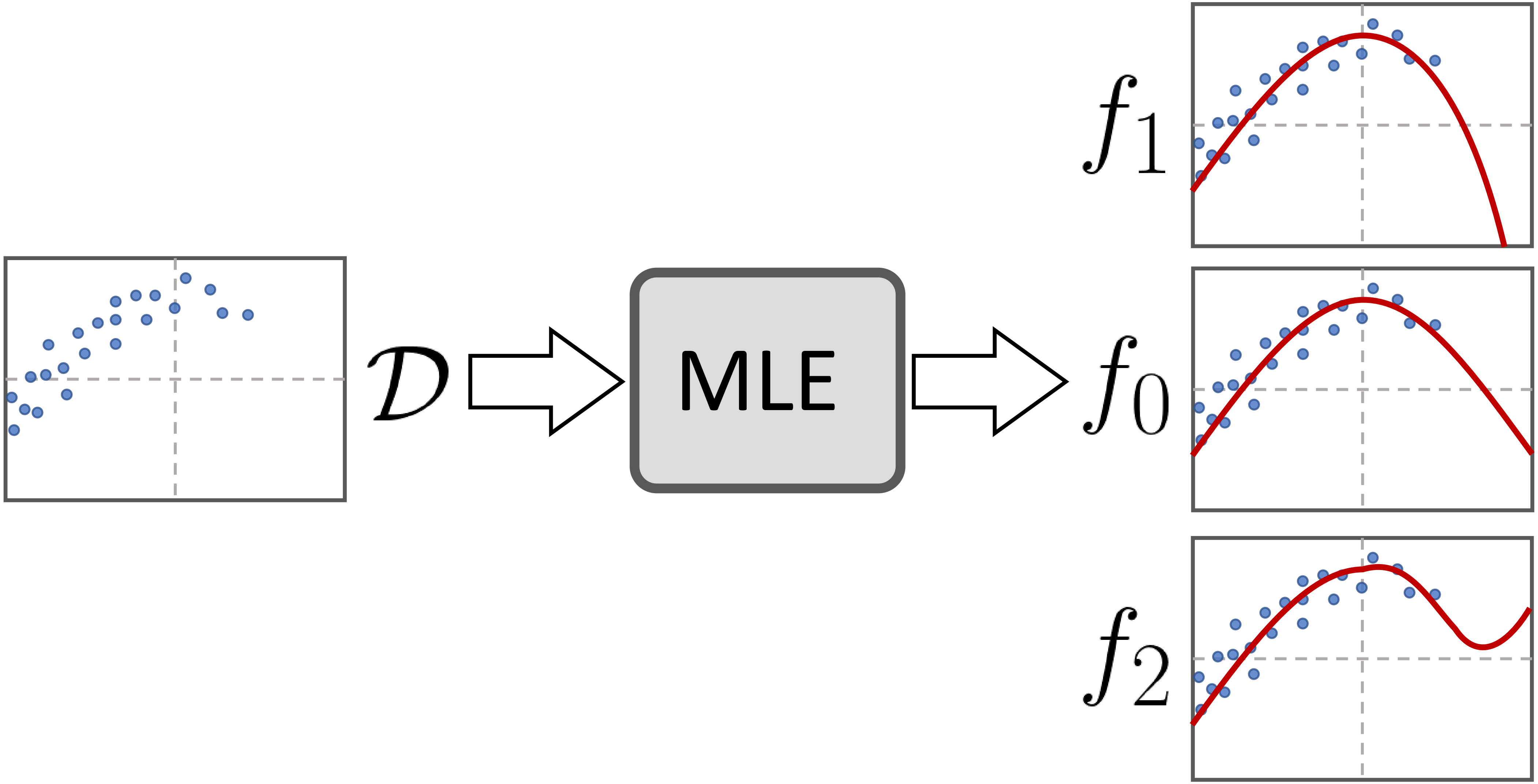
Maximum Likelihood

$$\arg\max_f \mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim\mathcal{D}}\left[\log f(\mathbf{s}'|\mathbf{s},\mathbf{a})\right]$$

Supervised Learning



Supervised Learning



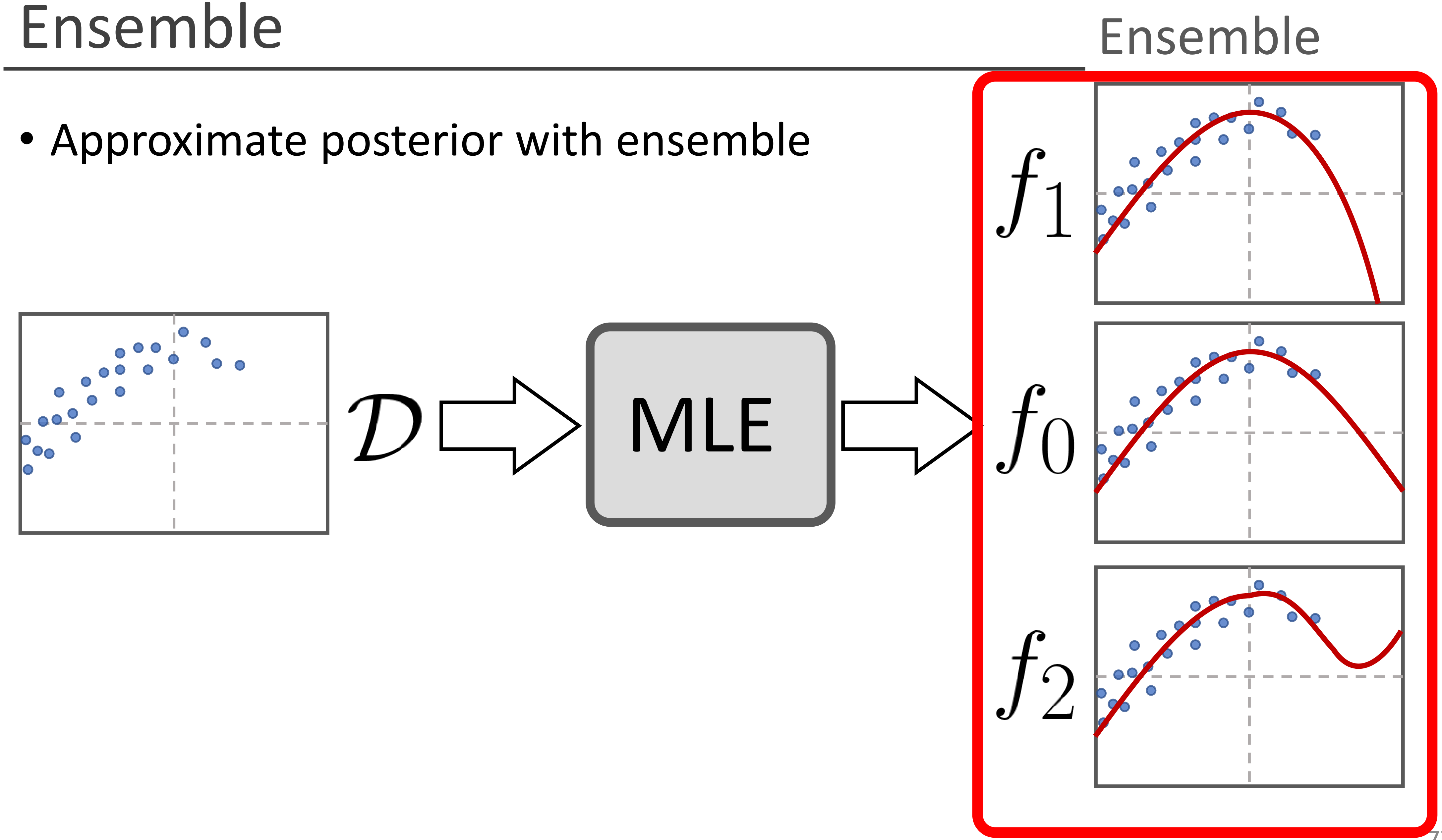
Uncertainty Estimation

- Maximum likelihood only gives a *point-wise* approximation of the posterior
- To estimate model uncertainty, need to approximate the *full* posterior

$$\underbrace{\arg\max_{f}}_{\text{Point-wise approximation}} \log \underbrace{p(f|\mathcal{D})}_{\text{Posterior}}$$

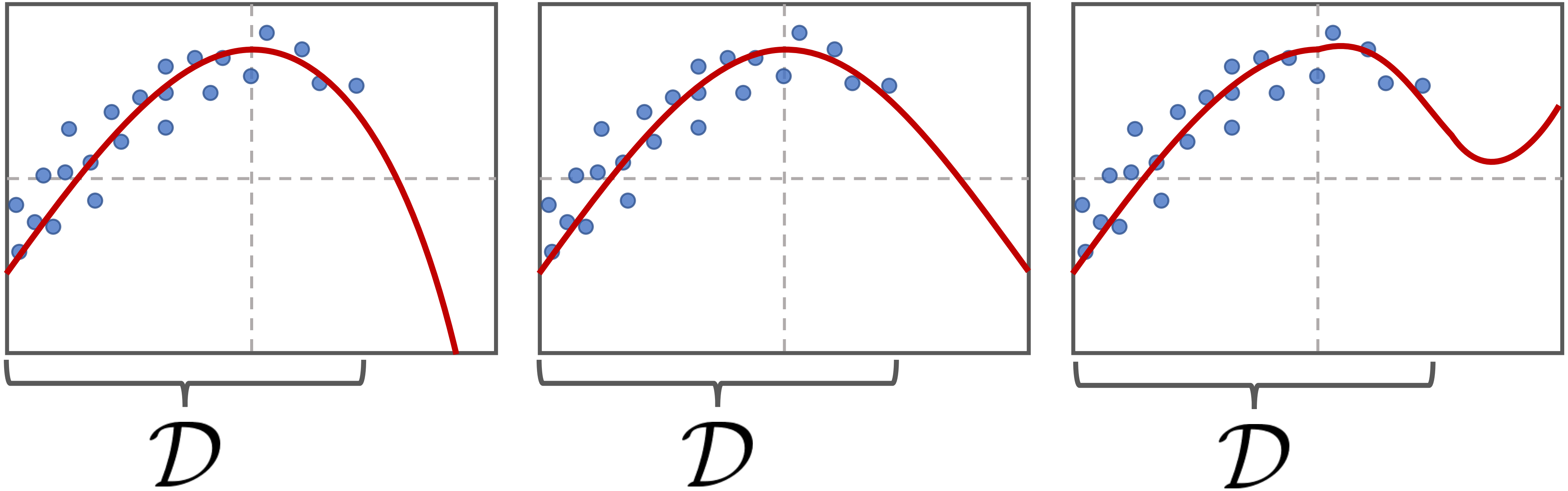
Ensemble

- Approximate posterior with ensemble



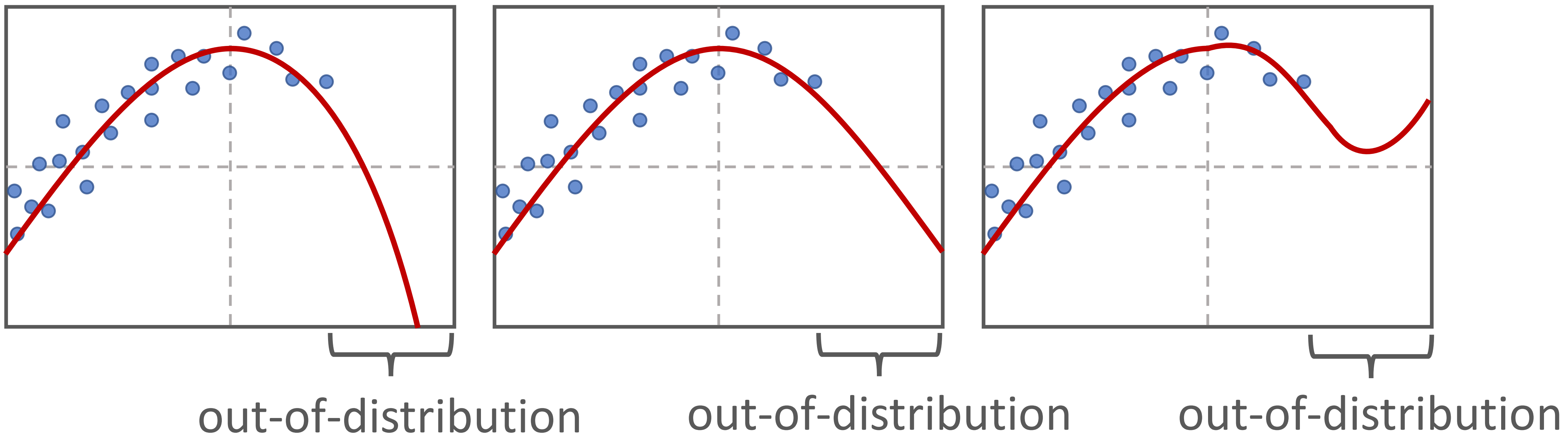
Ensemble

- Approximate posterior with ensemble
- Models should be consistent under the data distribution



Ensemble

- Approximate posterior with ensemble
- Models should be consistent under the data distribution
- Models will hopefully disagree on out-of-distribution samples



How to train ensemble?

Bootstrapping

- Split dataset into subsets
- Train a separate model for each subset

Reduces data available to train each model

$$
\begin{tikzcd}
 & \mathcal{D}_0 \arrow[r, "\text{MLE}"] & f_0 \\
\mathcal{D} \arrow[ur] \arrow[r] \arrow[dr] & \mathcal{D}_1 \arrow[r, "\text{MLE}"] & f_1 \\
 & \mathcal{D}_2 \arrow[r, "\text{MLE}"] & f_2
\end{tikzcd}
$$

How to train ensemble?

Bootstrapping

- Split dataset into subsets
- Train a separate model for each subset

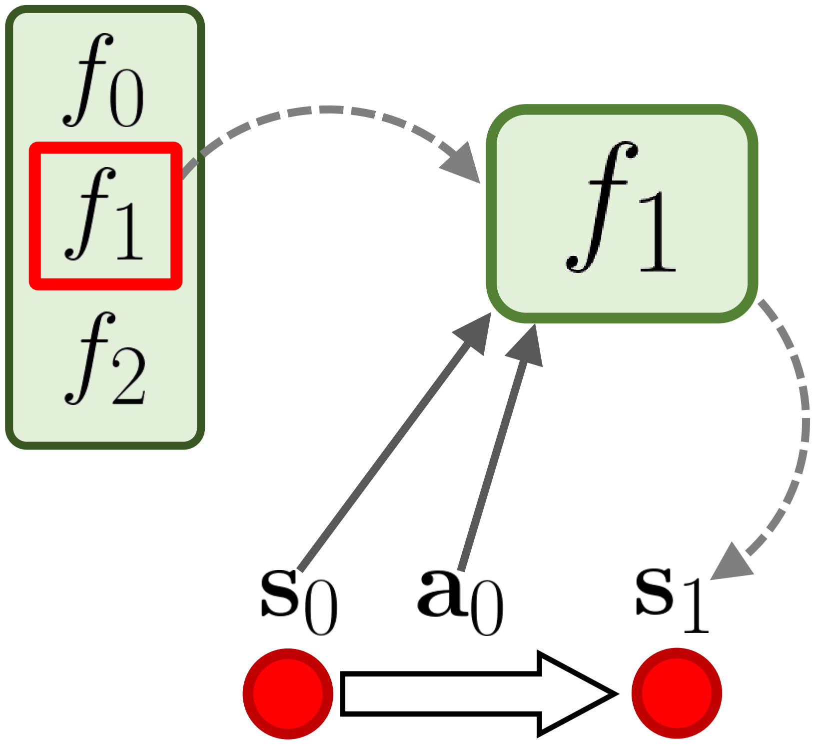
✘ Reduces data available to train each model

In practice:

- Initialize models with different random parameters
- Train all models using the same dataset
- Stochasticity from SGD leads to diverse models

How to use ensemble?

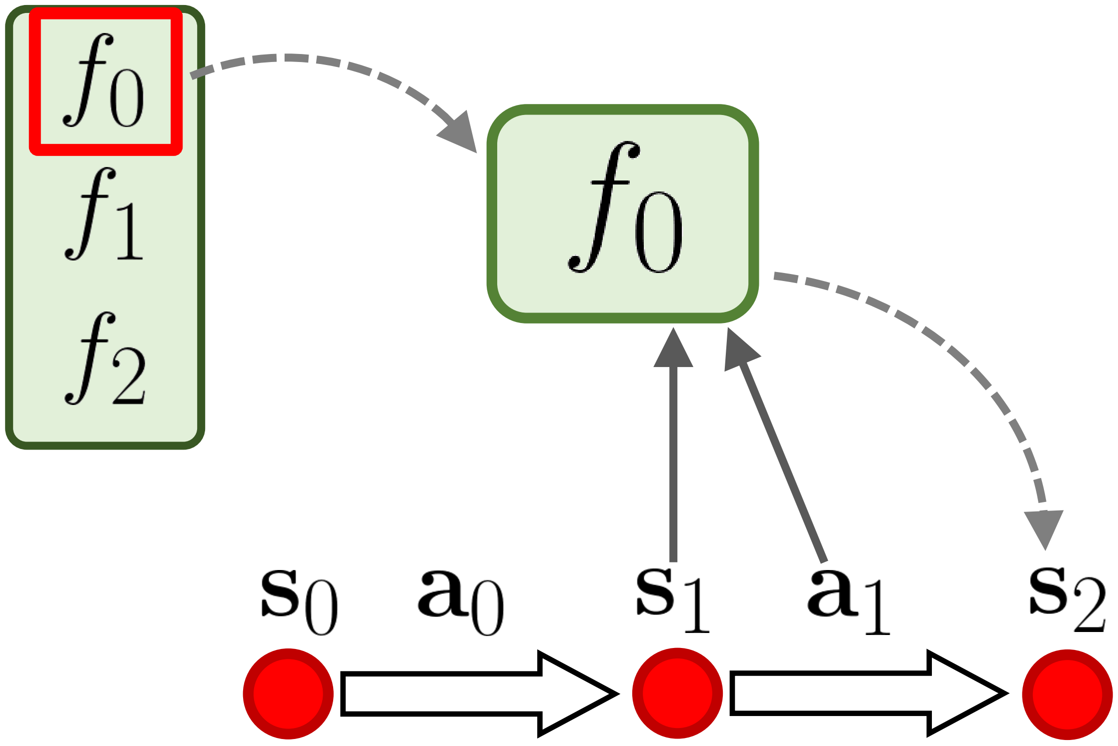
- Sample random model for every transition



When to Trust Your Model: Model-Based Policy Optimization
[Janner et al. 2019]

How to use ensemble?

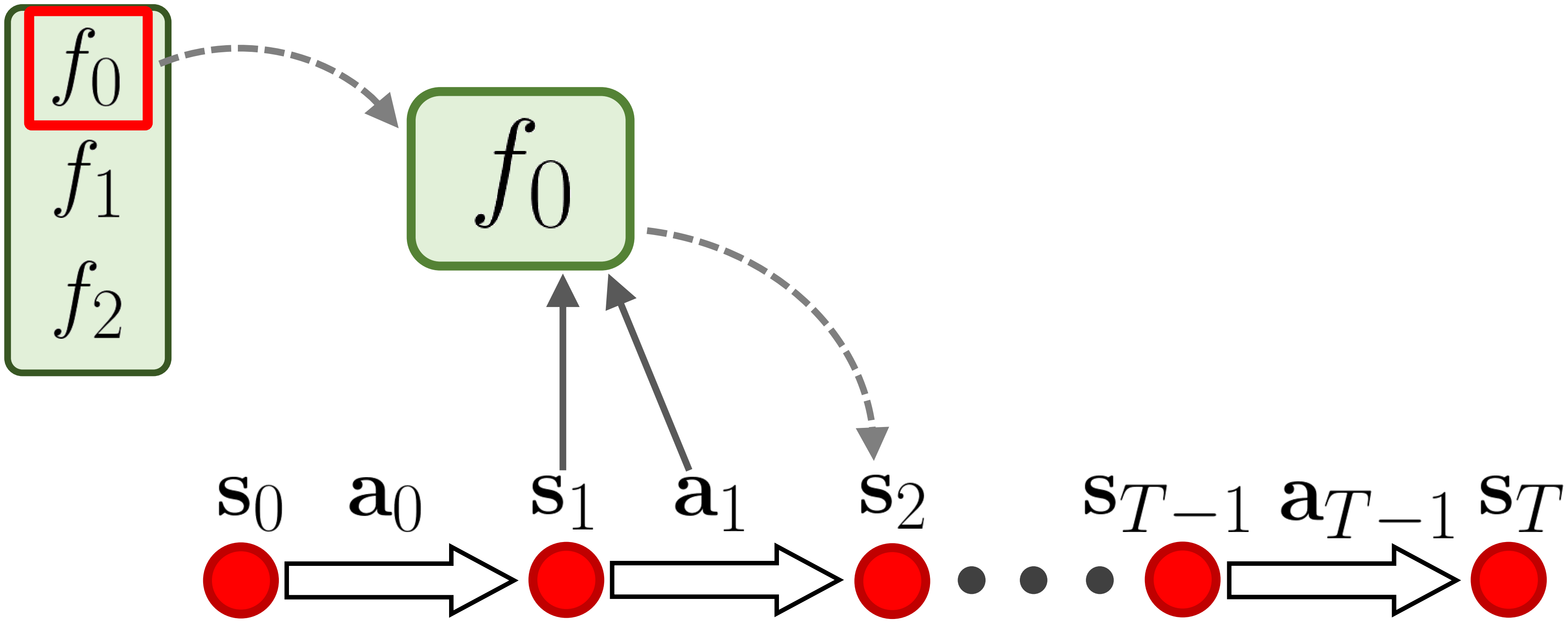
- Sample random model for every transition



When to Trust Your Model: Model-Based Policy Optimization
[Janner et al. 2019]

How to use ensemble?

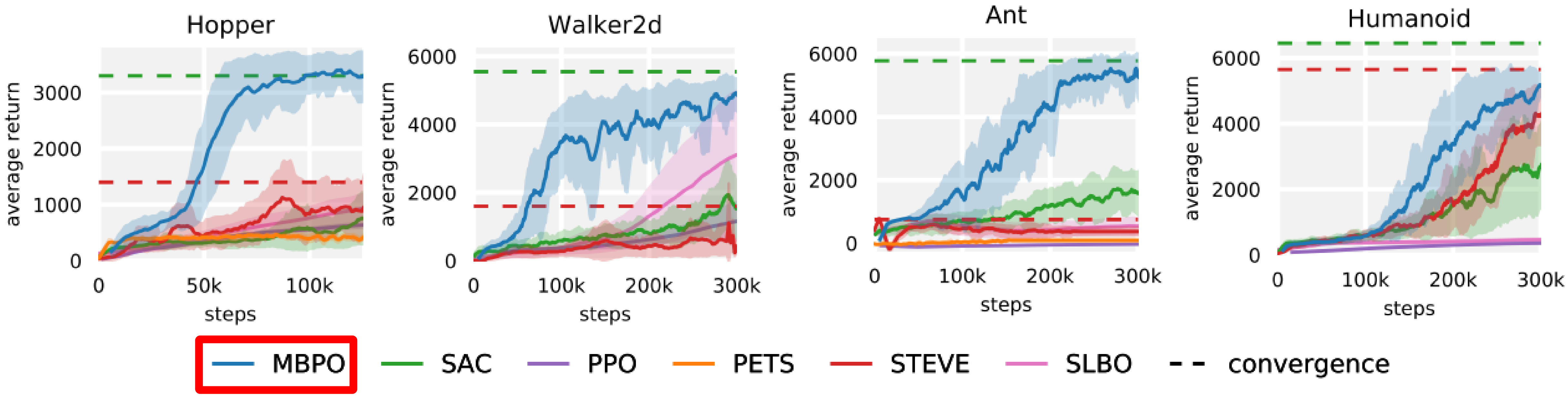
- Sample random model for every transition



When to Trust Your Model: Model-Based Policy Optimization
[Janner et al. 2019]

How to use ensemble?

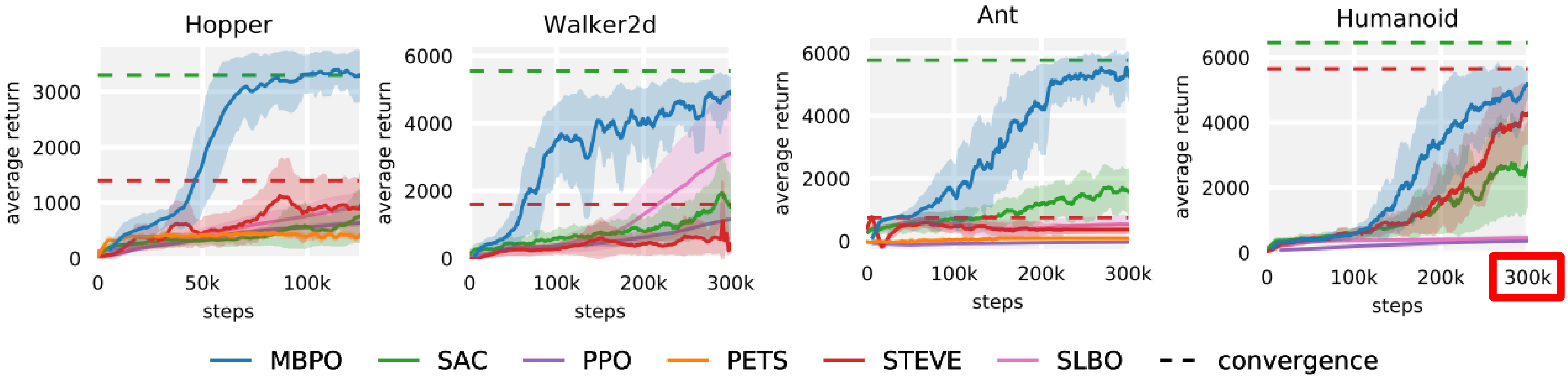
- Sample random model for every transition



When to Trust Your Model: Model-Based Policy Optimization
[Janner et al. 2019]

How to use ensemble?

- Sample random model for every transition



When to Trust Your Model: Model-Based Policy Optimization
[Janner et al. 2019]

How to use ensemble?

- Sample random model for every transition
- Penalize policy for model disagreement

$$r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$$

MOReL: Model-Based Offline Reinforcement Learning
[Kidambi et al. 2020]

How to use ensemble?

- Sample random model for every transition
- Penalize policy for model disagreement

$$r_p(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \begin{cases} -\kappa & \text{if } \underline{d(\mathbf{s}, \mathbf{a})} > \alpha \\ r(\mathbf{s}, \mathbf{a}, \mathbf{s}') & \text{otherwise} \end{cases}$$

MOReL: Model-Based Offline Reinforcement Learning
[Kidambi et al. 2020]

How to use ensemble?

- Sample random model for every transition
- Penalize policy for model disagreement

$$r_p(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \begin{cases} -\kappa & \text{if } d(\mathbf{s}, \mathbf{a}) \underline{> \alpha} \\ r(\mathbf{s}, \mathbf{a}, \mathbf{s}') & \text{otherwise} \end{cases}$$

MOReL: Model-Based Offline Reinforcement Learning
[Kidambi et al. 2020]

How to use ensemble?

- Sample random model for every transition
- Penalize policy for model disagreement

$$r_p(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \begin{cases} \underline{-\kappa} & \text{if } d(\mathbf{s}, \mathbf{a}) > \alpha \\ r(\mathbf{s}, \mathbf{a}, \mathbf{s}') & \text{otherwise} \end{cases}$$

MOReL: Model-Based Offline Reinforcement Learning
[Kidambi et al. 2020]

How to use ensemble?

- Sample random model for every transition
- Penalize policy for model disagreement

$$r_p(\mathbf{s}, \mathbf{a}, \mathbf{s}') = \begin{cases} -\kappa & \text{if } d(\mathbf{s}, \mathbf{a}) > \alpha \\ r(\mathbf{s}, \mathbf{a}, \mathbf{s}') & \text{otherwise} \end{cases}$$

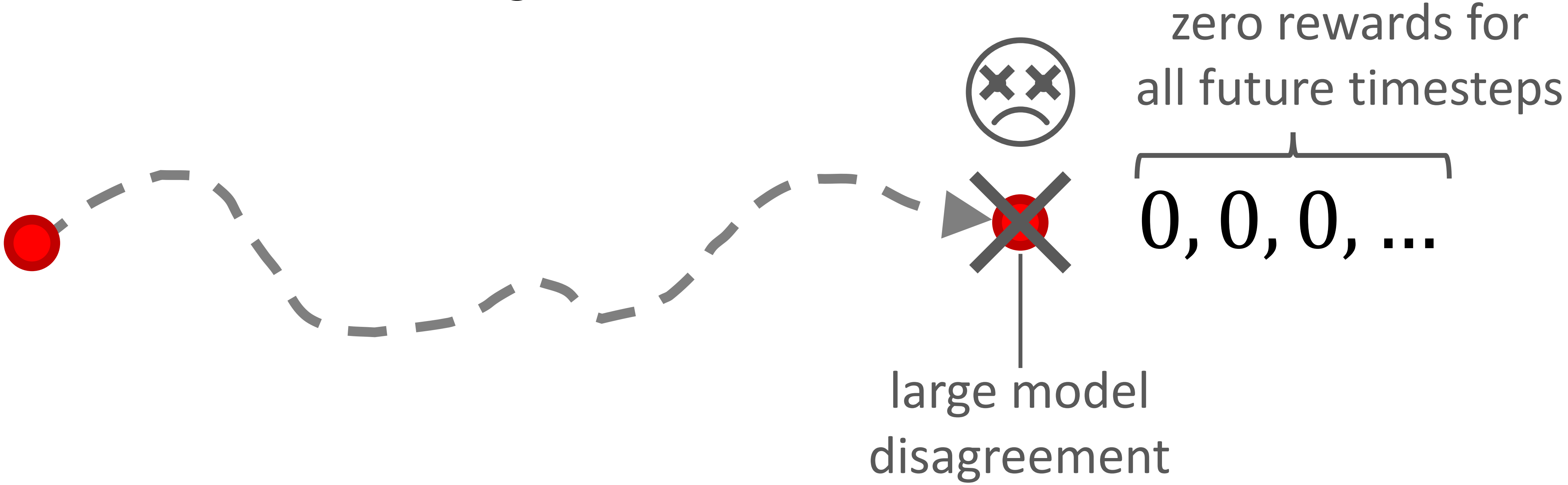
Model disagreement:

$$d(\mathbf{s}, \mathbf{a}) = \max_{i,j} \; D\left(f_i(\cdot|\mathbf{s}, \mathbf{a}), f_j(\cdot|\mathbf{s}, \mathbf{a})\right)$$

MOReL: Model-Based Offline Reinforcement Learning
[Kidambi et al. 2020]

How to use ensemble?

- Sample random model for every transition
- Penalize policy for model disagreement
- Termination based on disagreement



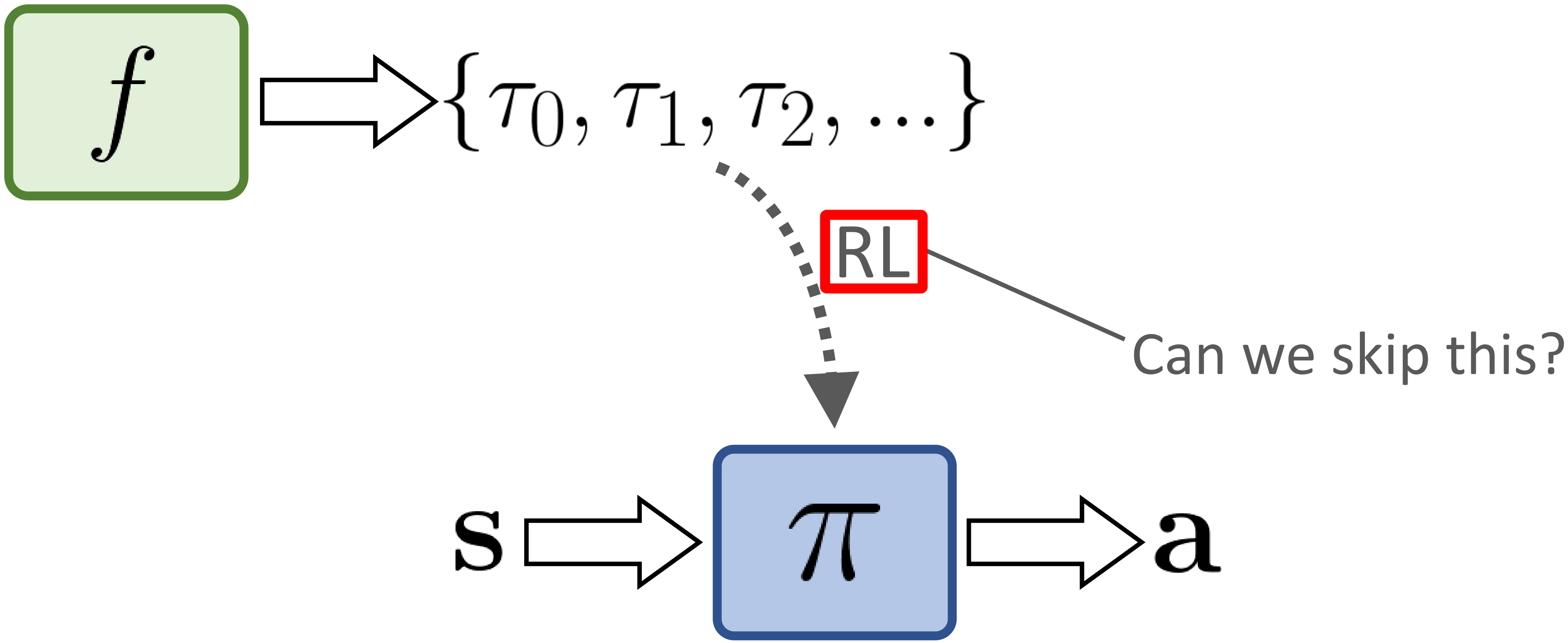
MOReL: Model-Based Offline Reinforcement Learning
[Kidambi et al. 2020]

Uncertainty Estimation

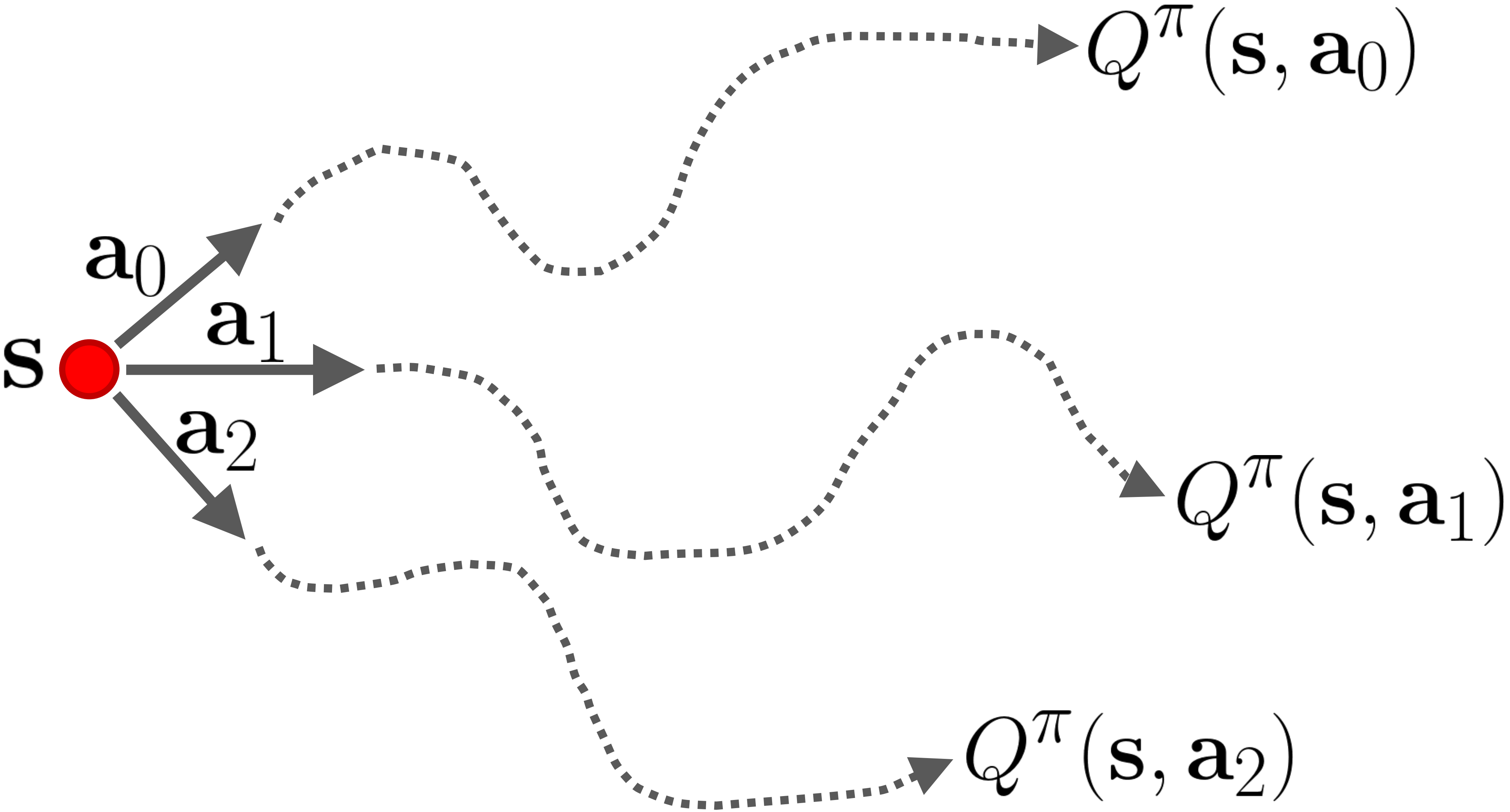
- Ensembles
- Bayesian Neural Networks
- Dropout
- Normalized Maximum Likelihood
- Test Time Augmentation
- Etc…

Model-Predictive Control

Model-Based Policy Learning

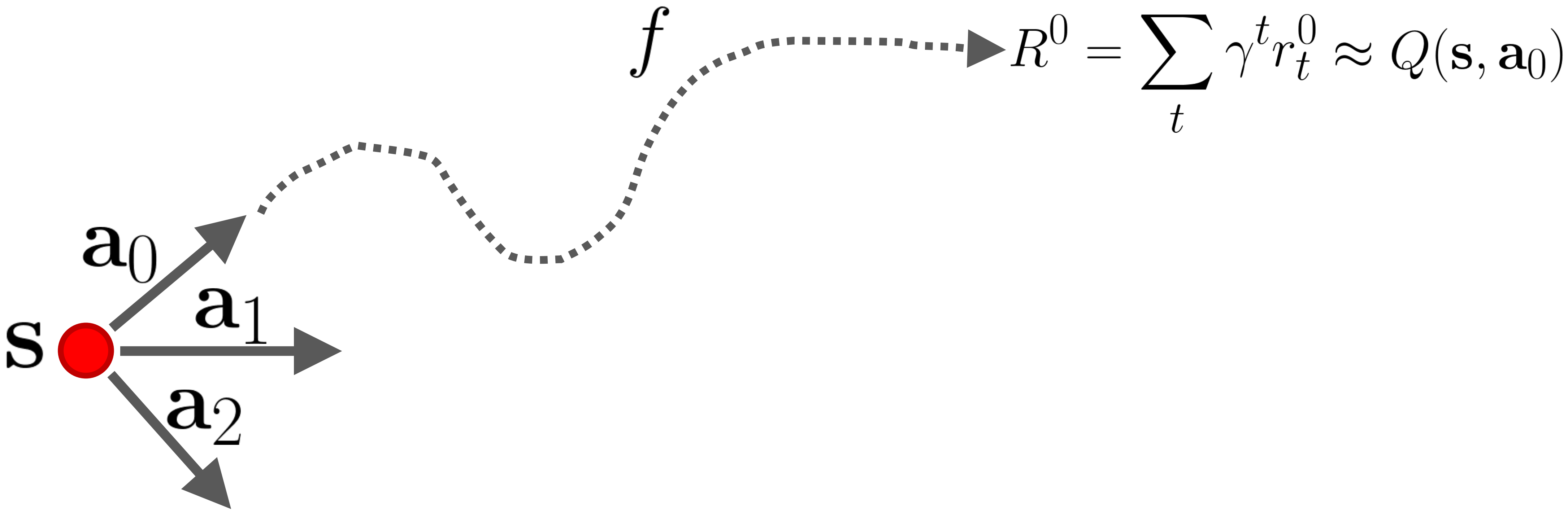


Q-Learning

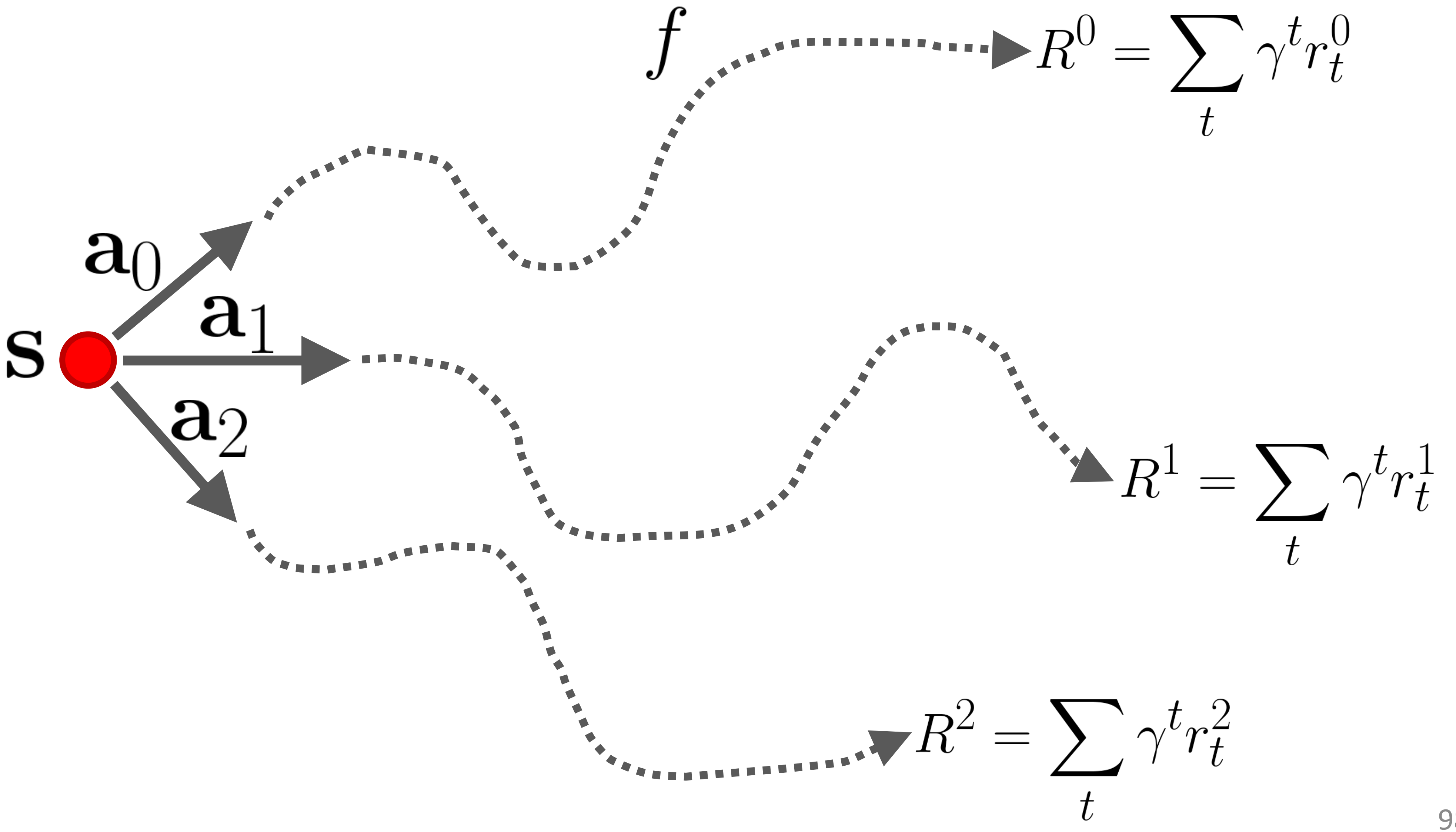


$\mathbf{s}$
$\mathbf{a}_0$
$\mathbf{a}_1$
$\mathbf{a}_2$
$Q^{\pi}(\mathbf{s}, \mathbf{a}_0)$
$Q^{\pi}(\mathbf{s}, \mathbf{a}_1)$
$Q^{\pi}(\mathbf{s}, \mathbf{a}_2)$

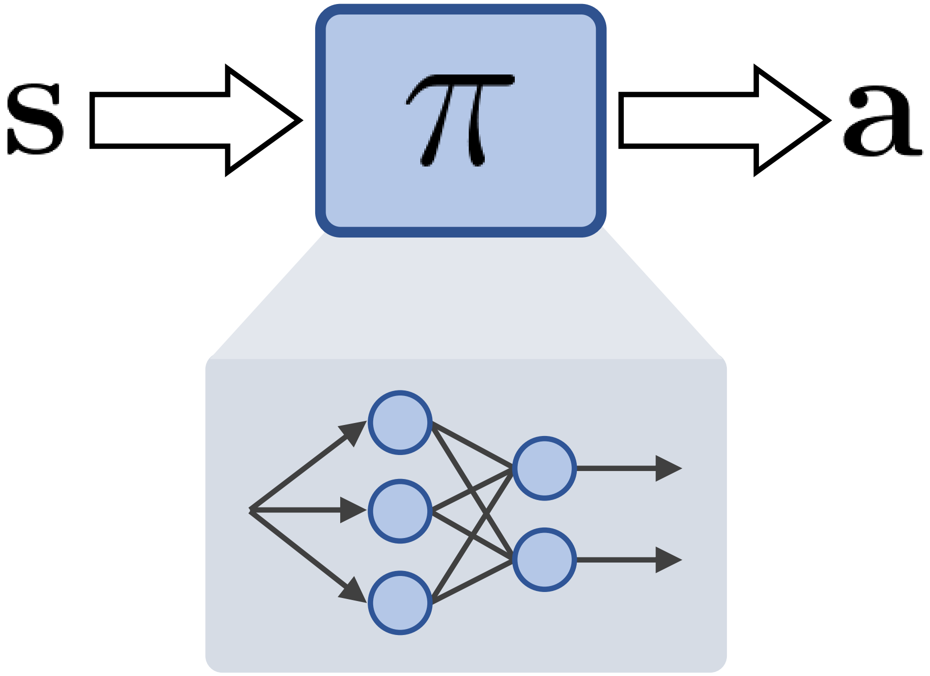
Online Planning



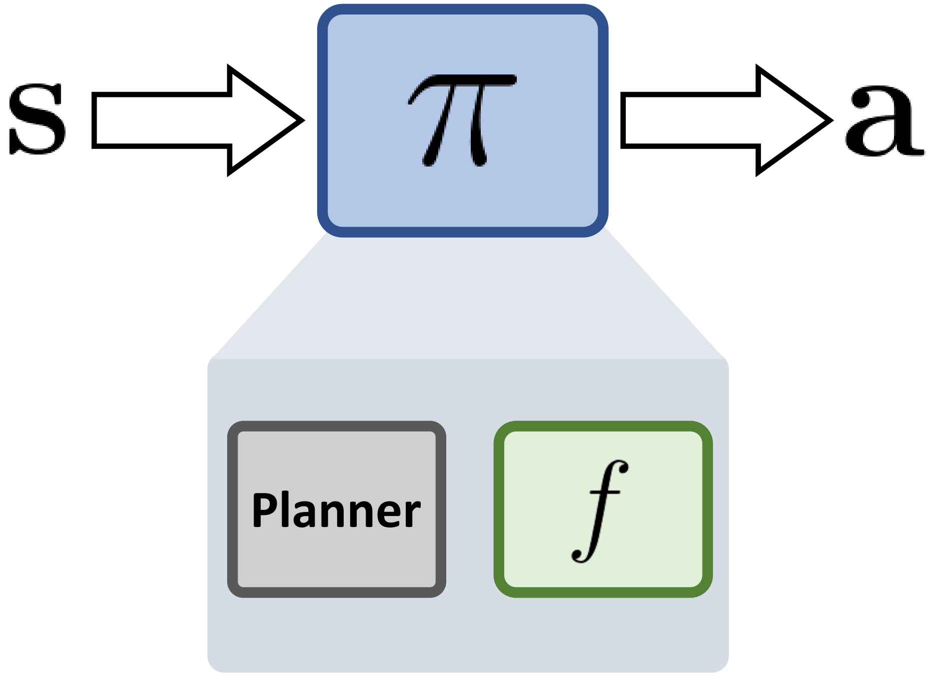
Online Planning



Online Planning



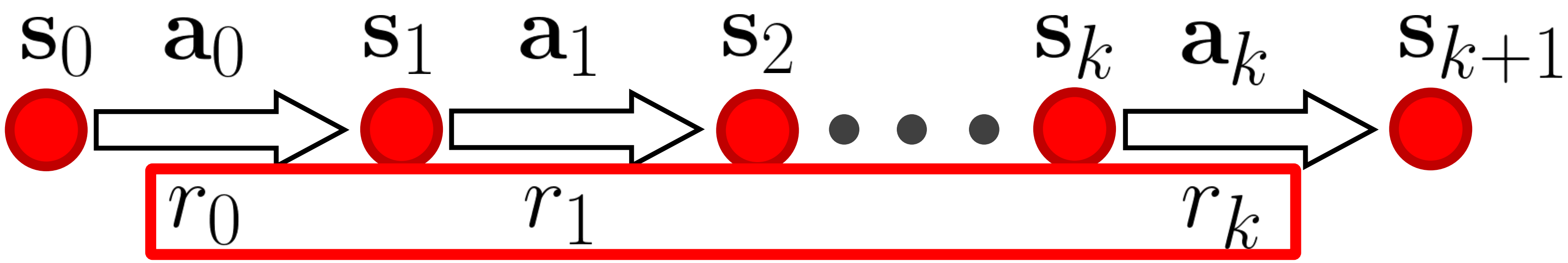
Online Planning



Online Planning

- Use dynamics model to predict expected return of every action

$$\arg\max_{\mathbf{a}_{0:k}} \mathbb{E}_{\tau \sim f(\tau | \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_{0:k})} [R(\tau)]$$

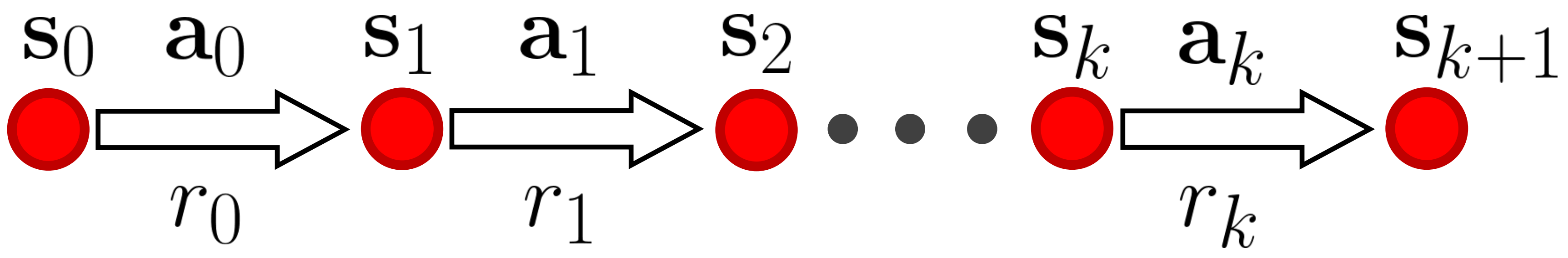


Online Planning

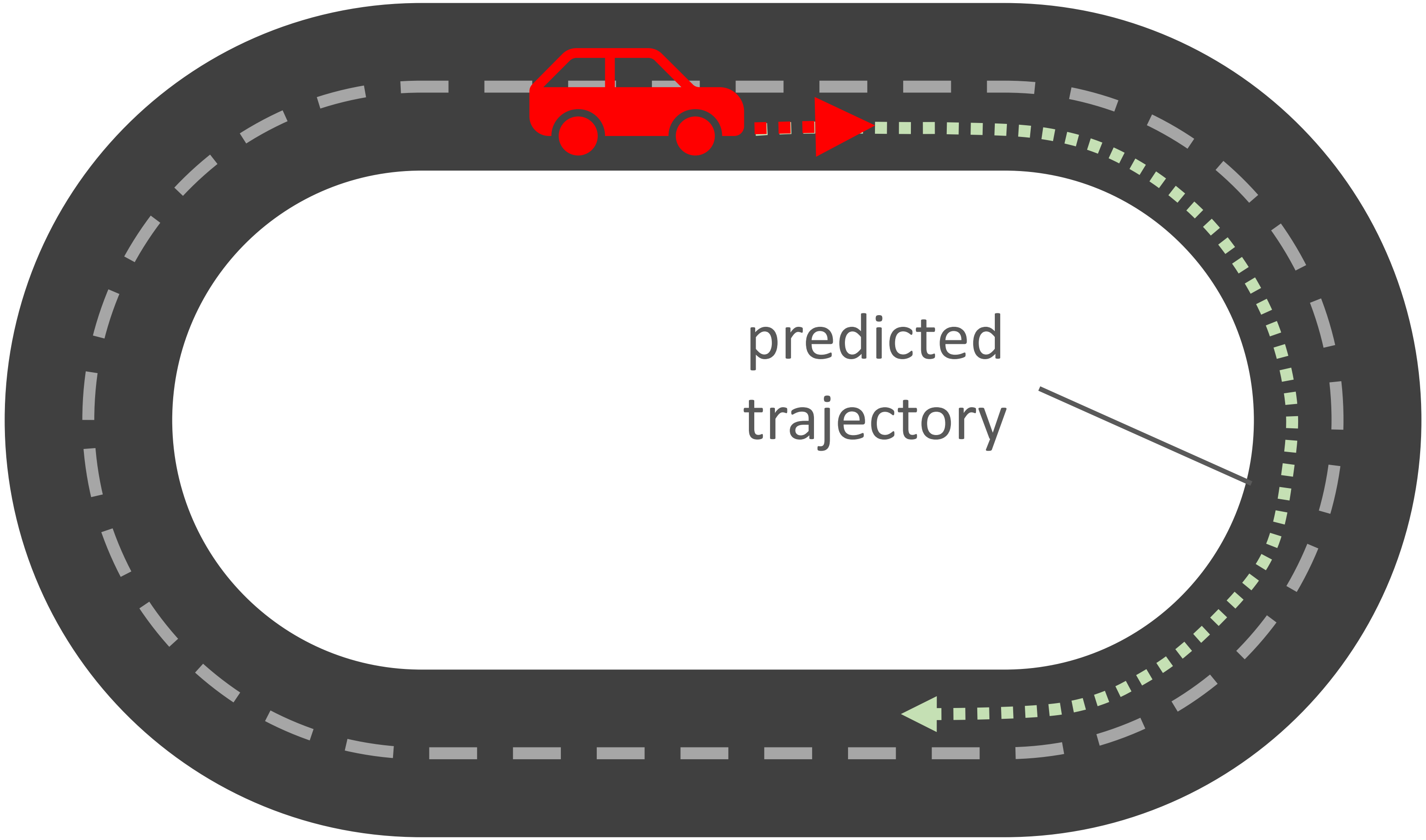
- Use dynamics model to predict expected return of every action

$$\arg\max_{\mathbf{a}_{0:k}} \mathbb{E}_{\tau\sim f(\tau|\mathbf{s}_0=\mathbf{s},\mathbf{a}_{0:k})}\left[R(\tau)\right]$$

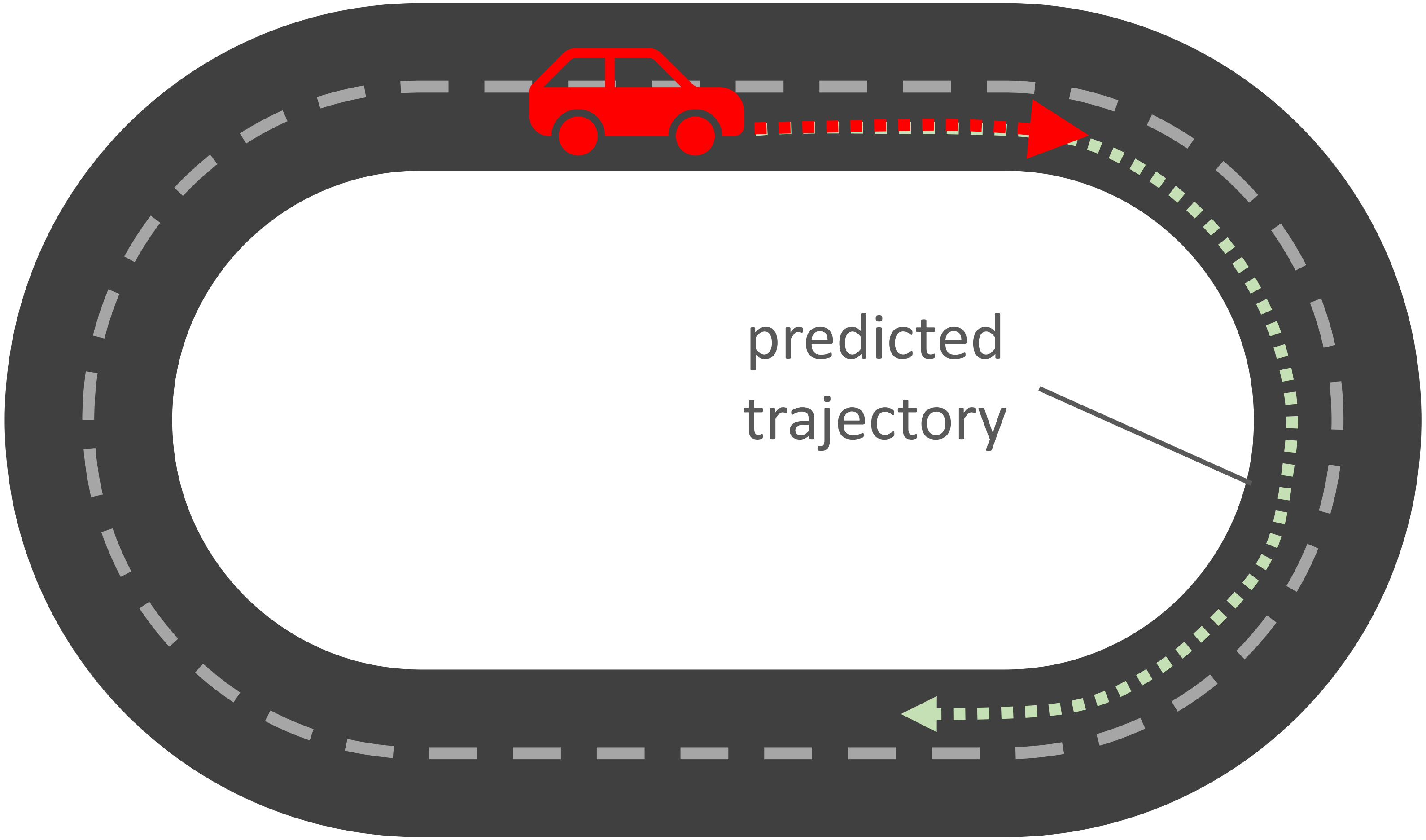
- Apply optimal action sequence $\mathbf{a}^*_{0:k}$ in real environment



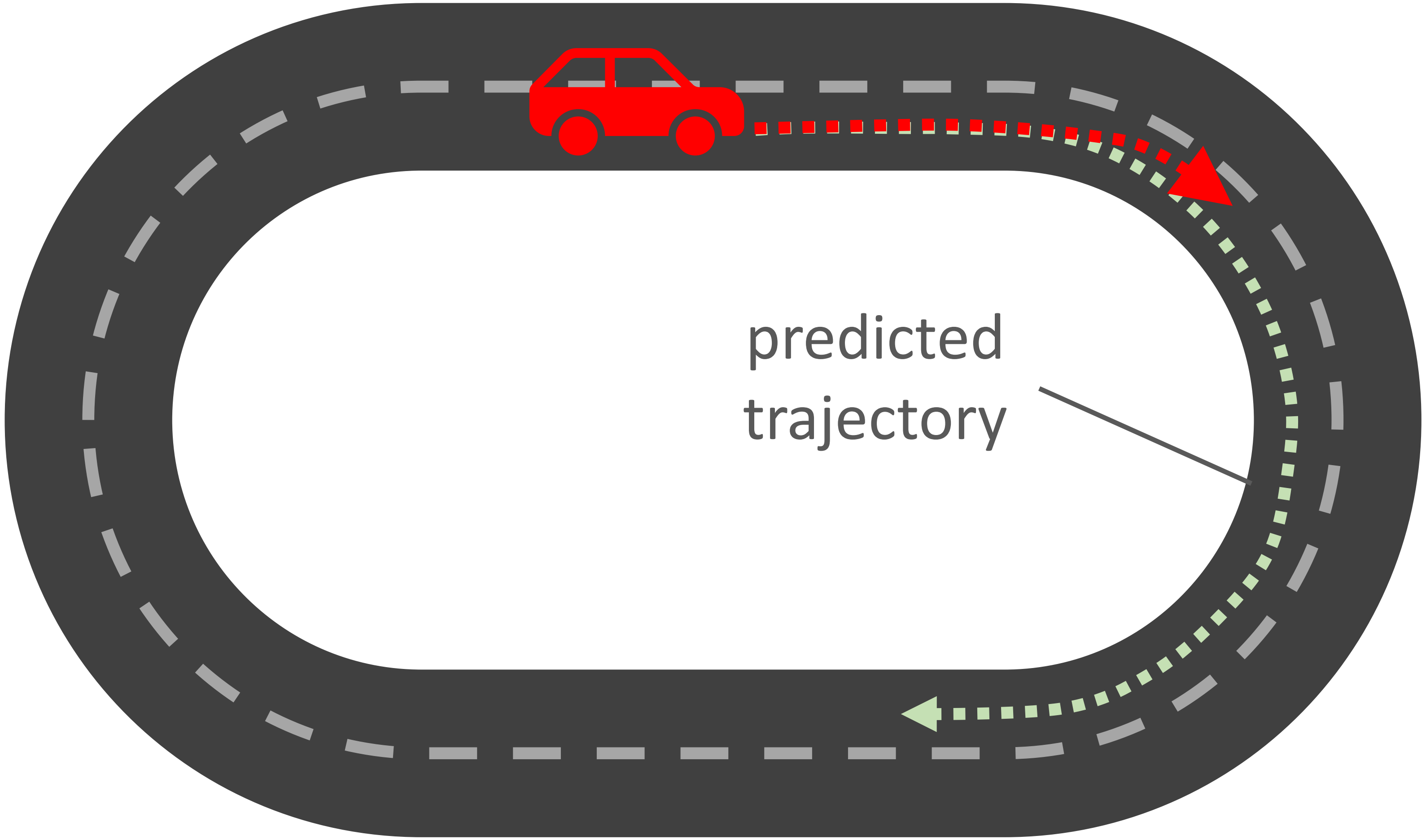
Drift



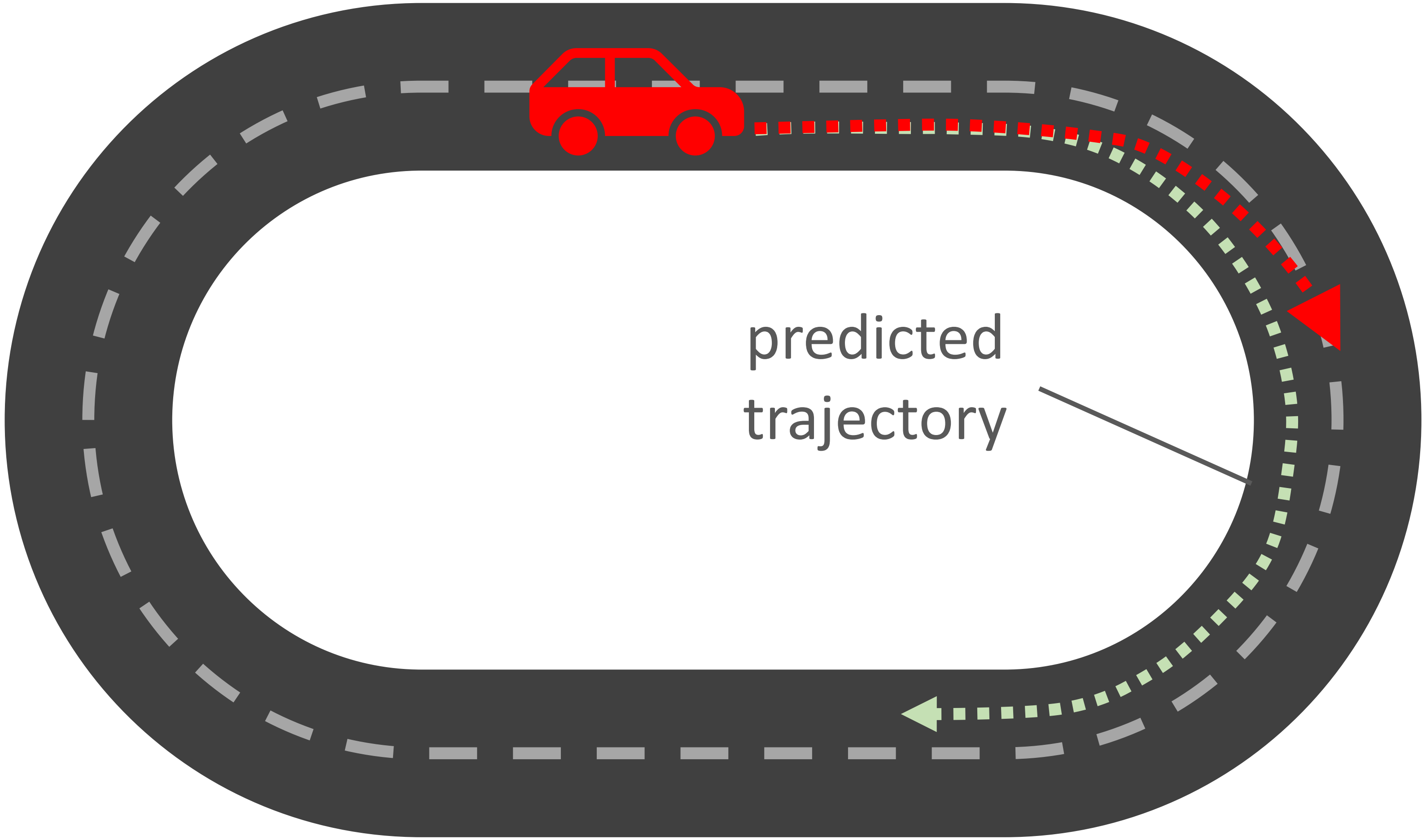
Drift



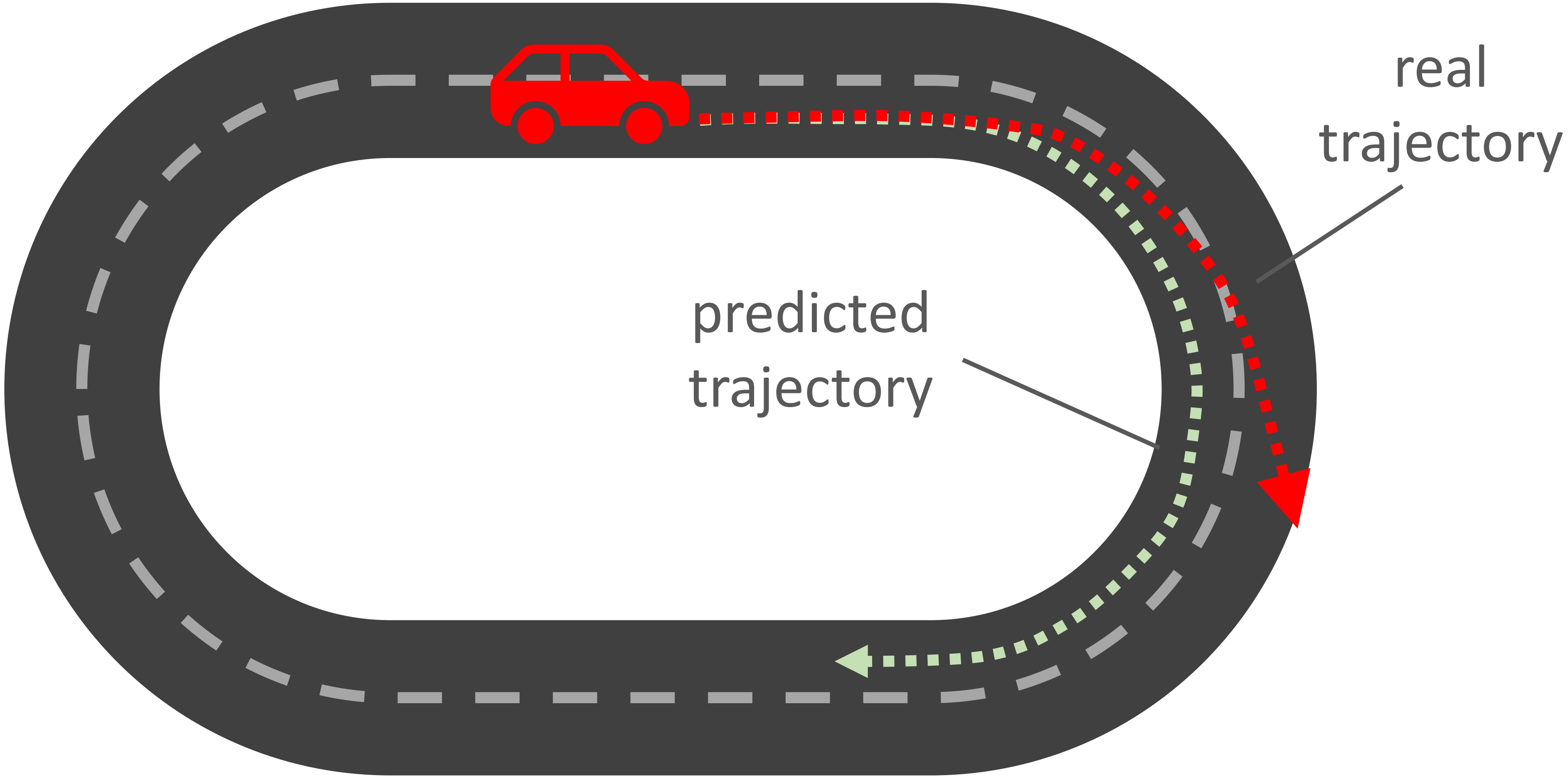
Drift



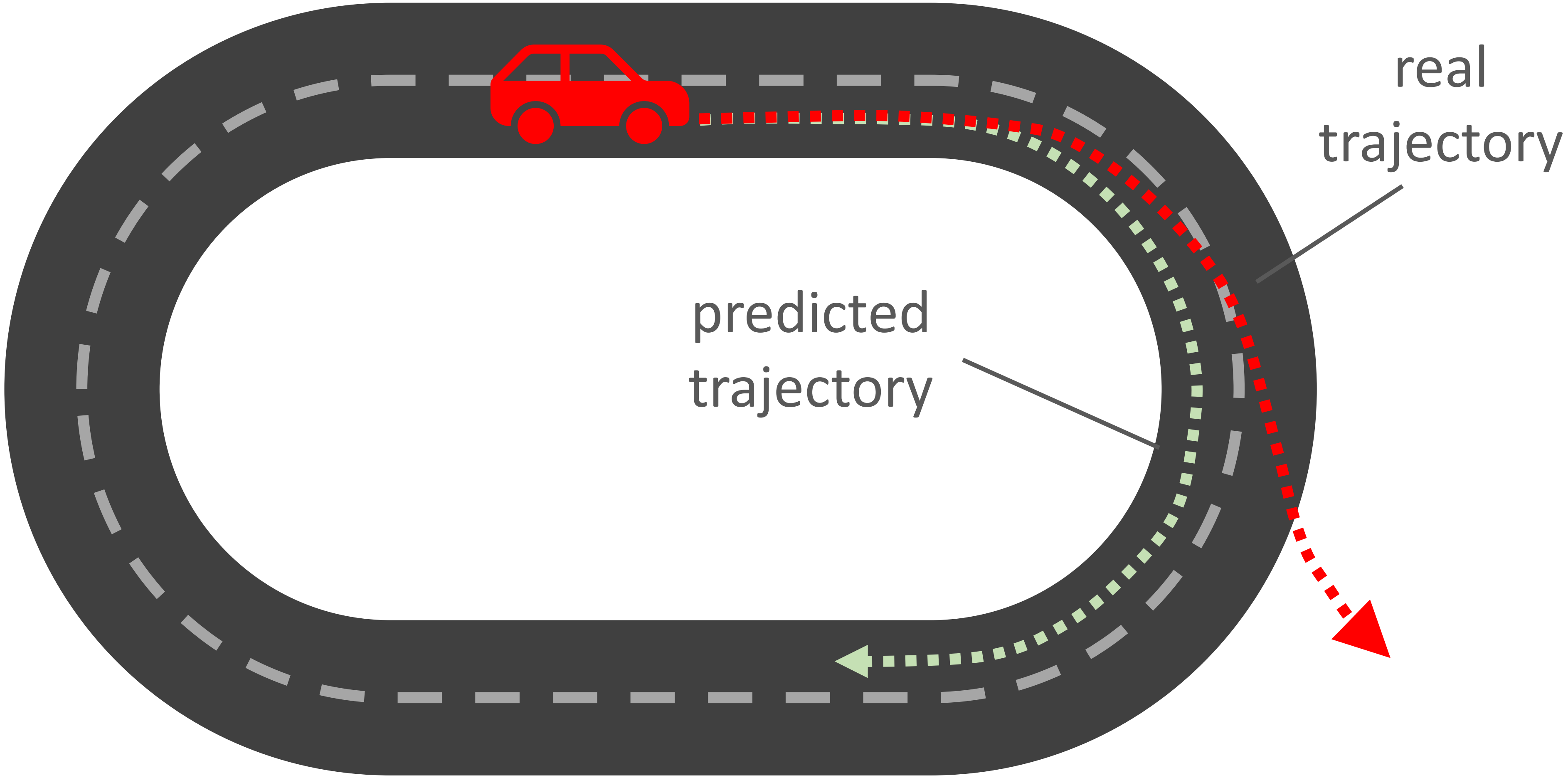
Drift



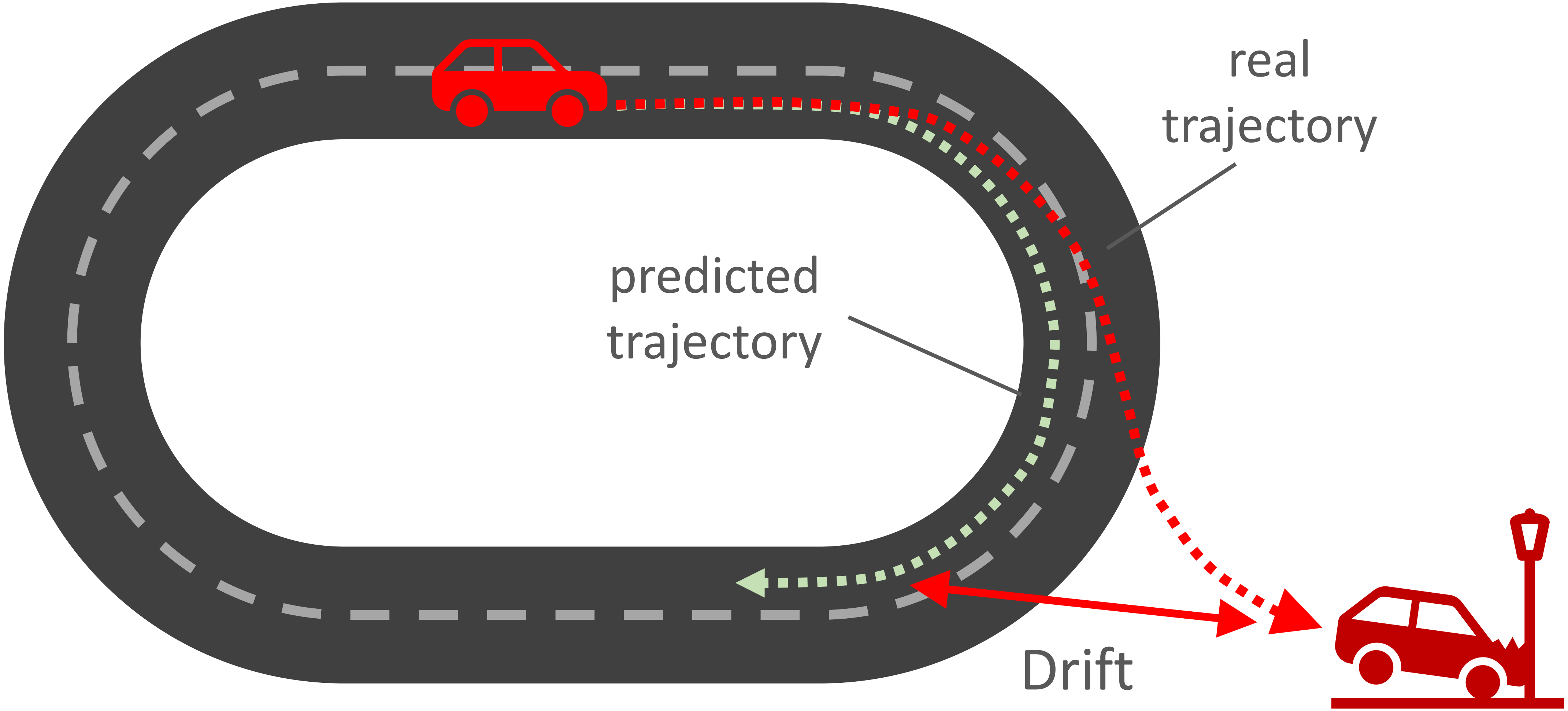
Drift



Drift



Drift



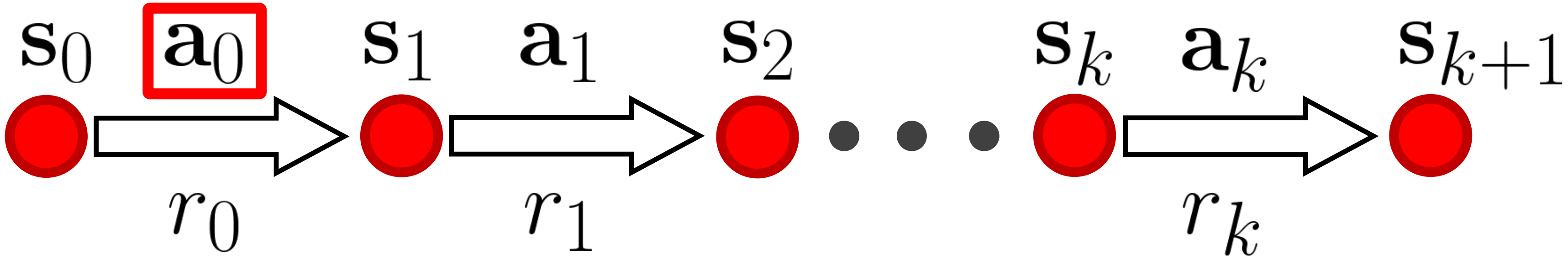
(Open-Loop Control)

MPC

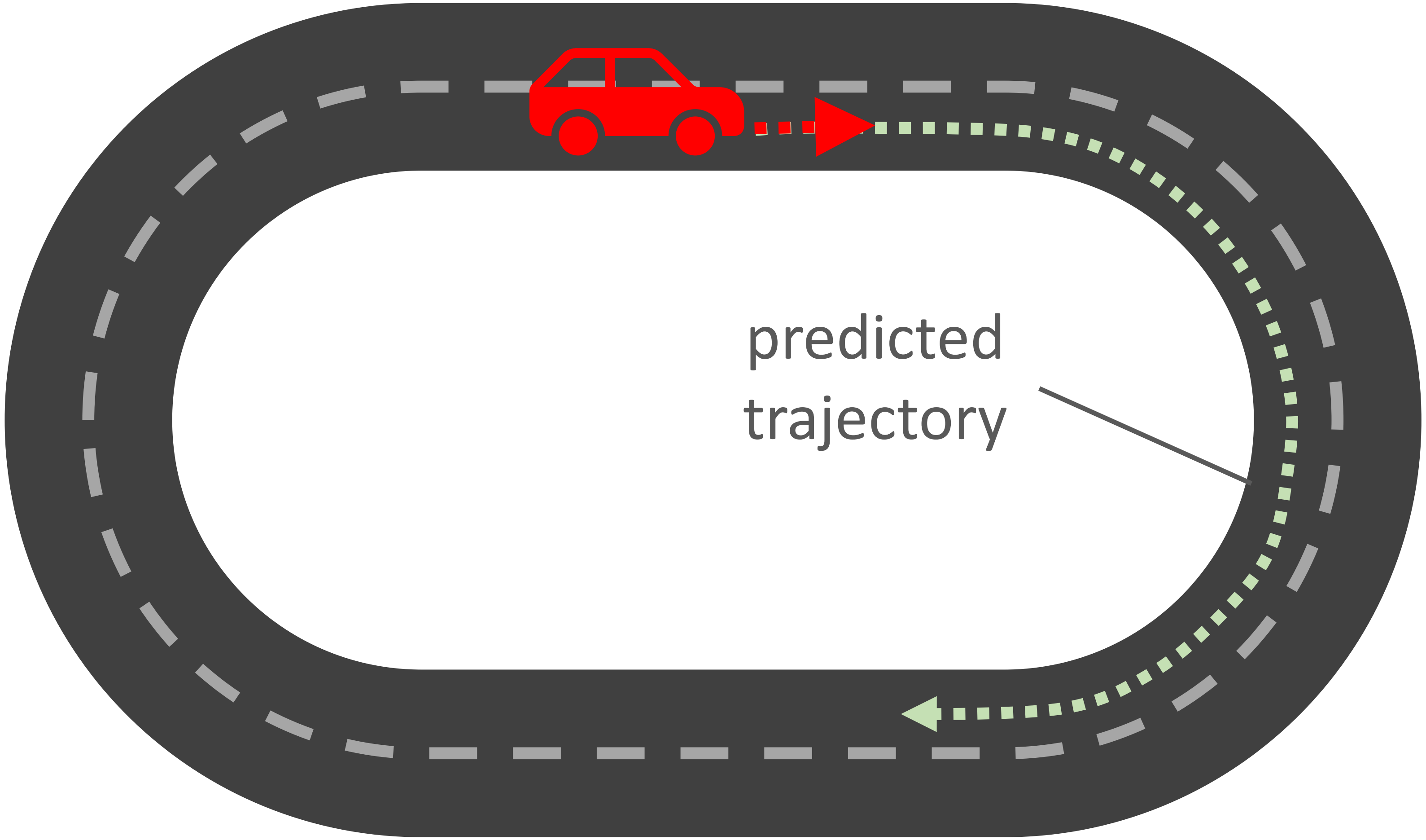
- Use dynamics model to predict expected return of every action

$$\arg\max_{\mathbf{a}_{0:k}} \mathbb{E}_{\tau \sim f(\tau | \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_{0:k})} [R(\tau)]$$

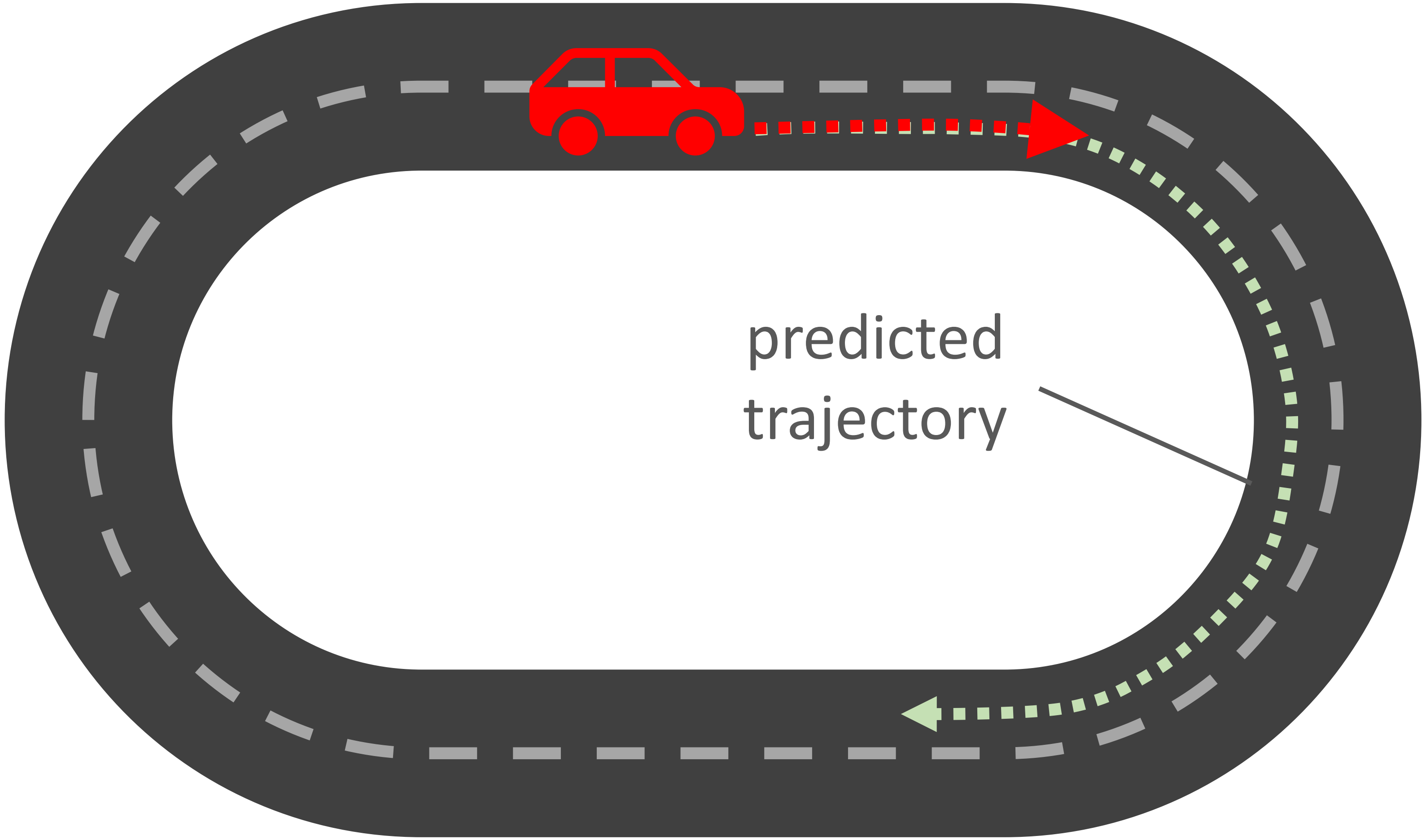
- ~~Apply optimal action sequence $\mathbf{a}^*_{0:k}$ in real environment~~
- Model Predictive Control (MPC)
  - Apply only the first action in the real environment
  - Replan every timestep



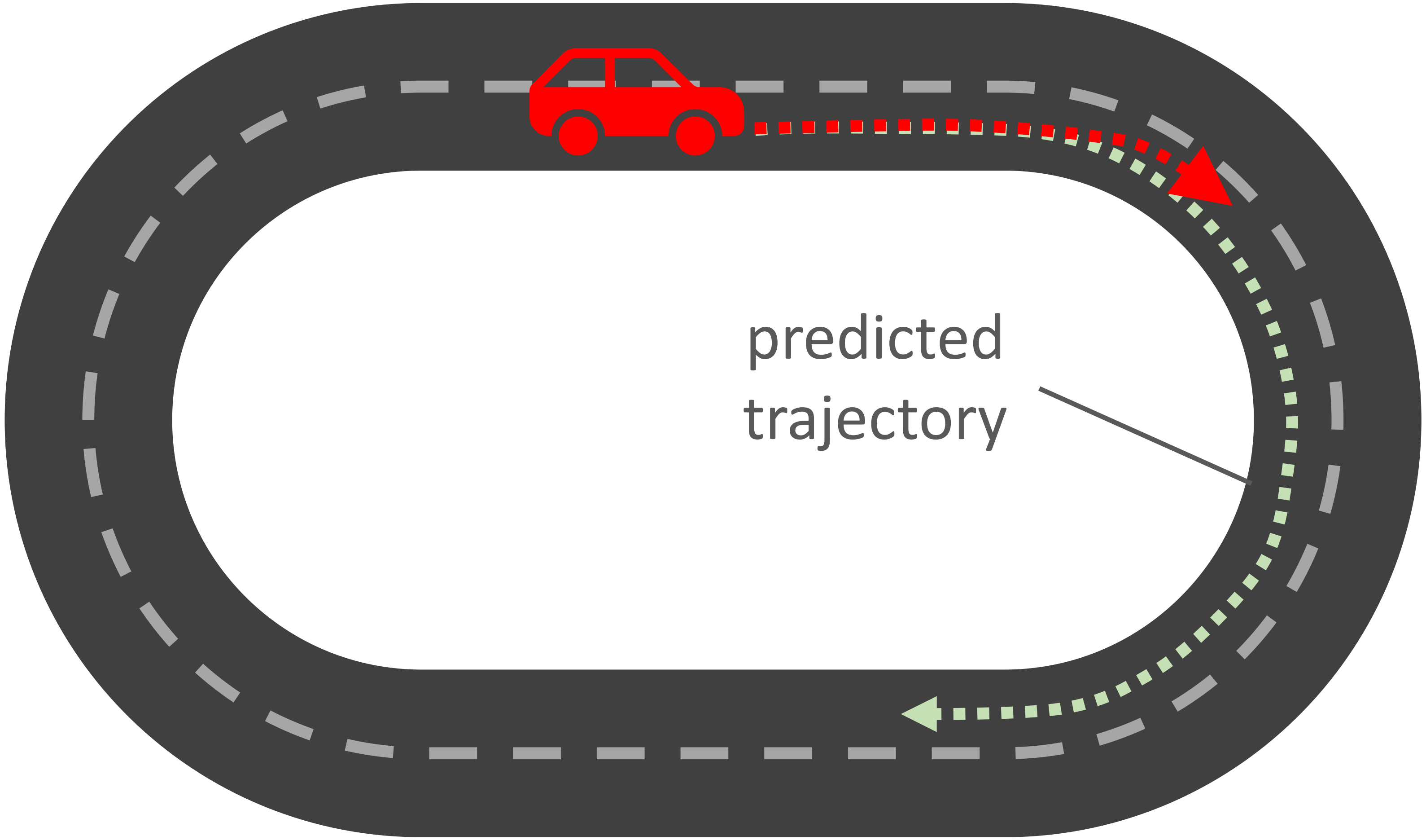
Drift



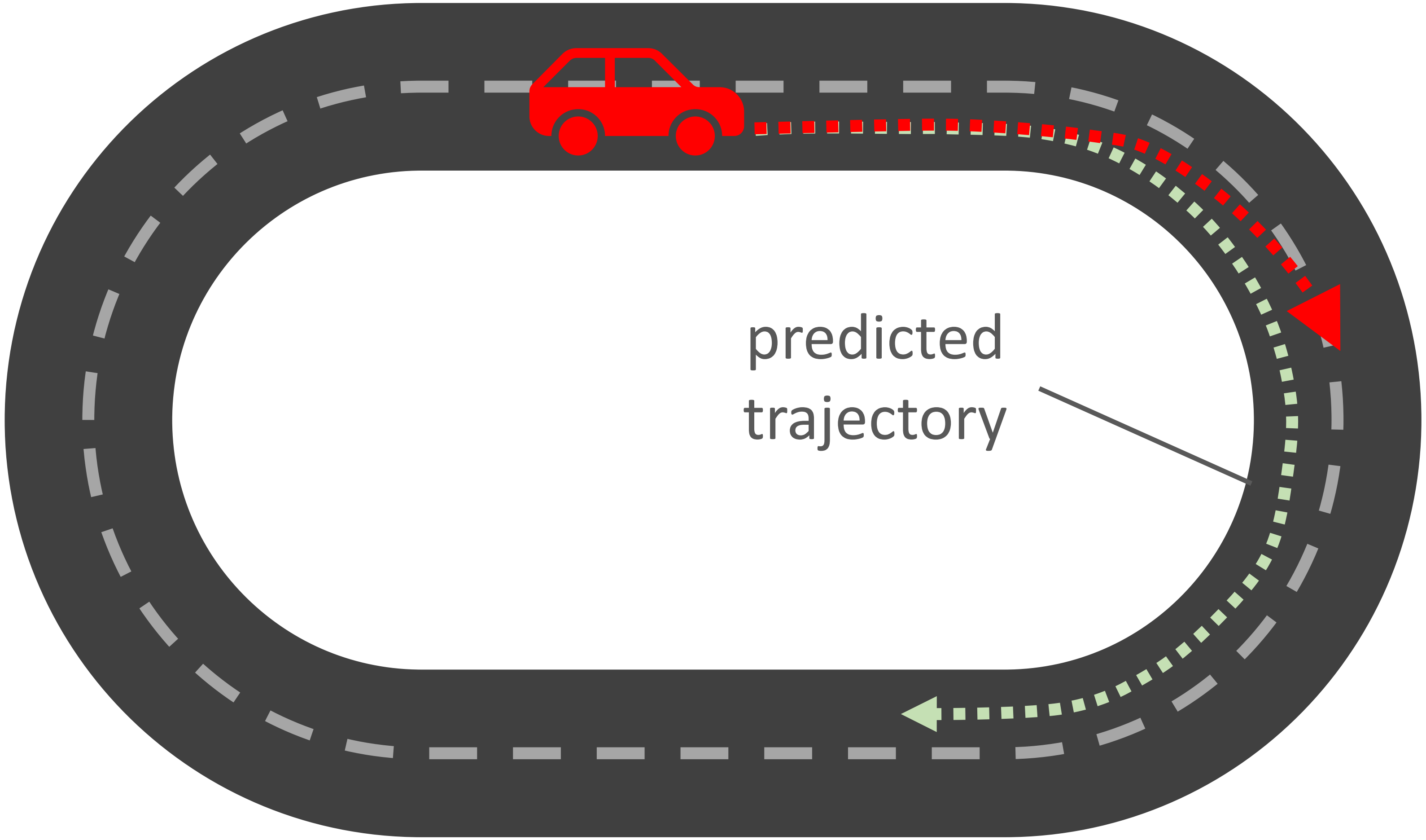
Drift



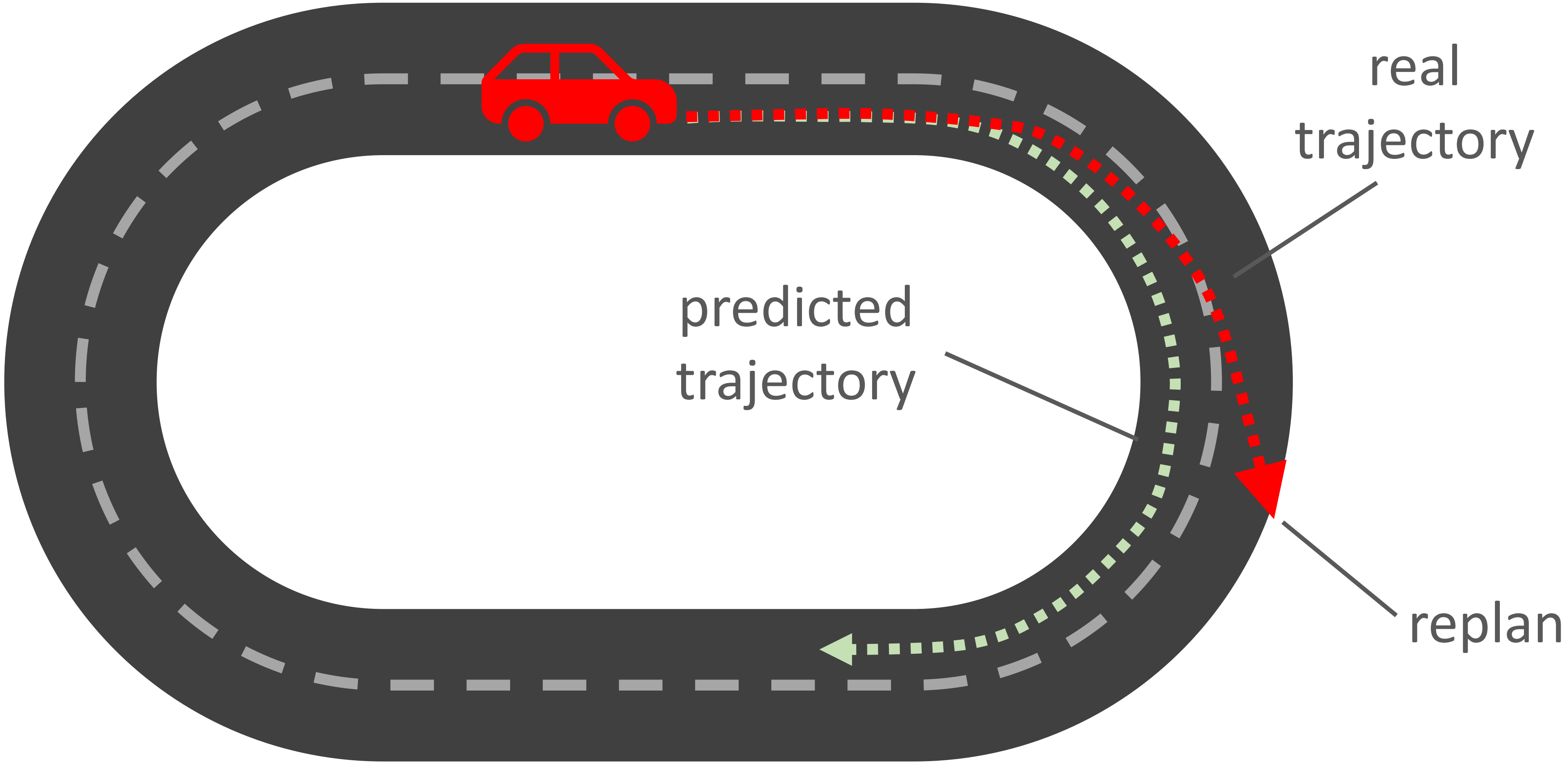
Drift



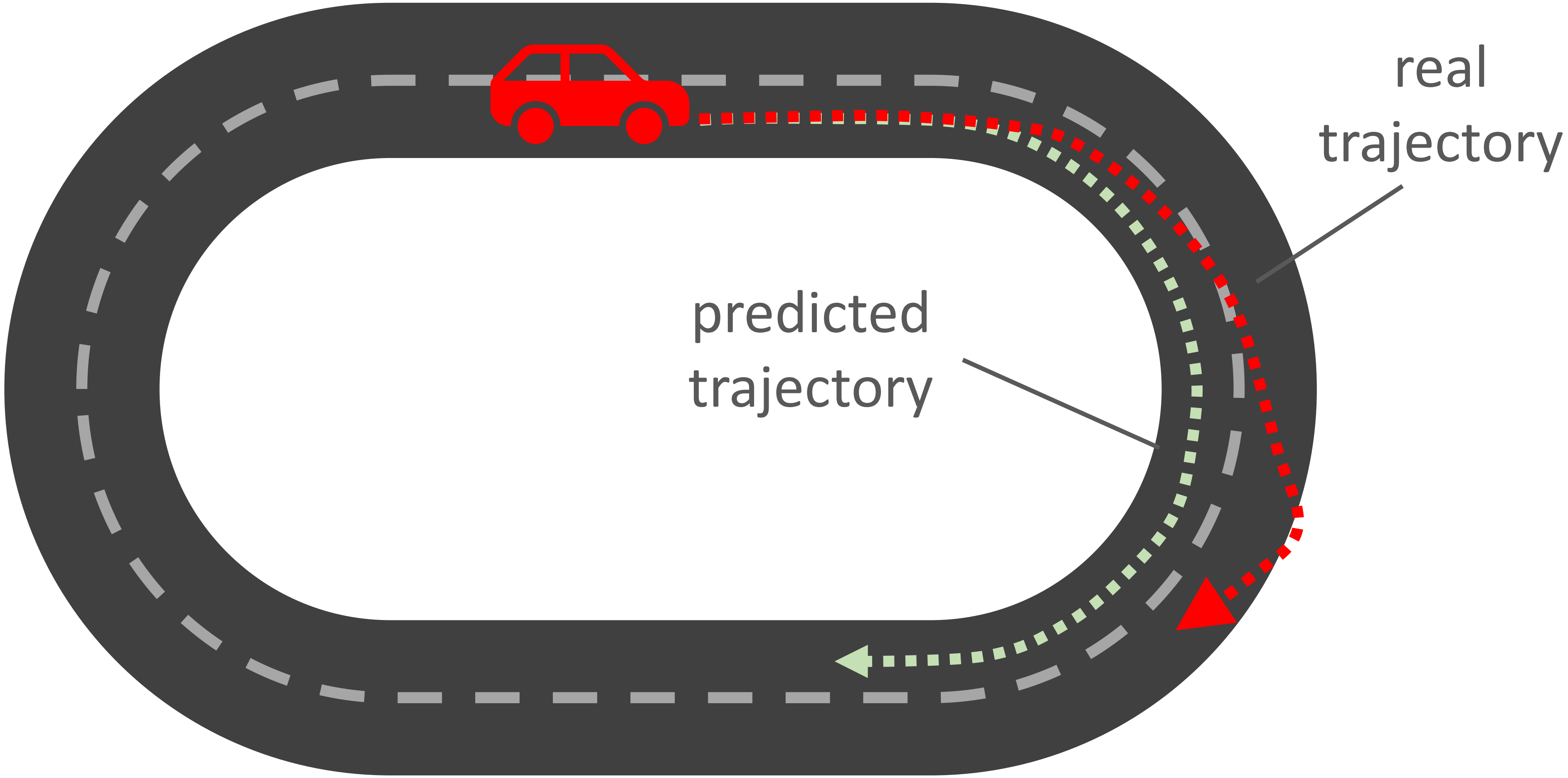
Drift



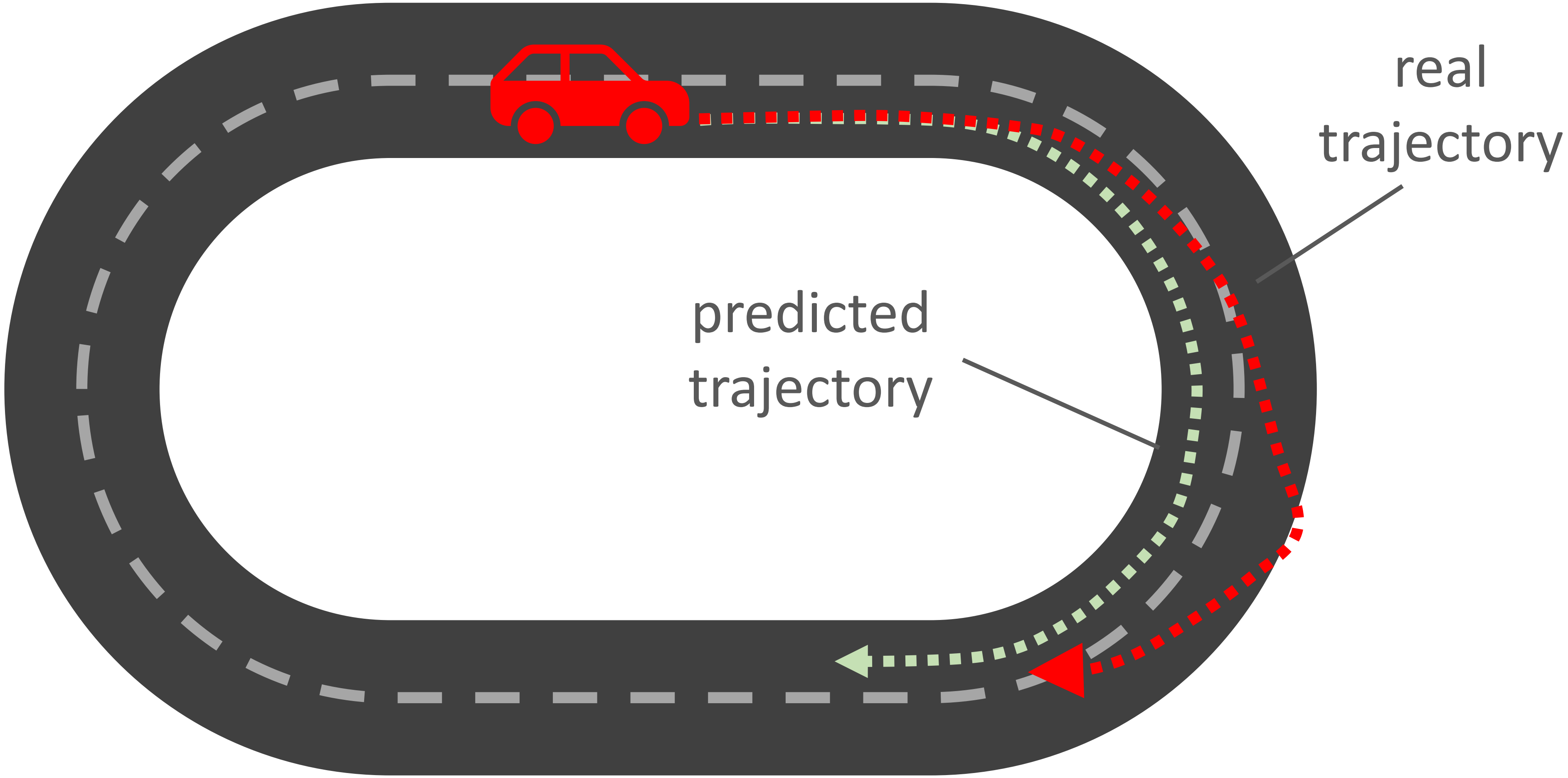
Drift



Drift



Drift



(Closed-Loop Control)

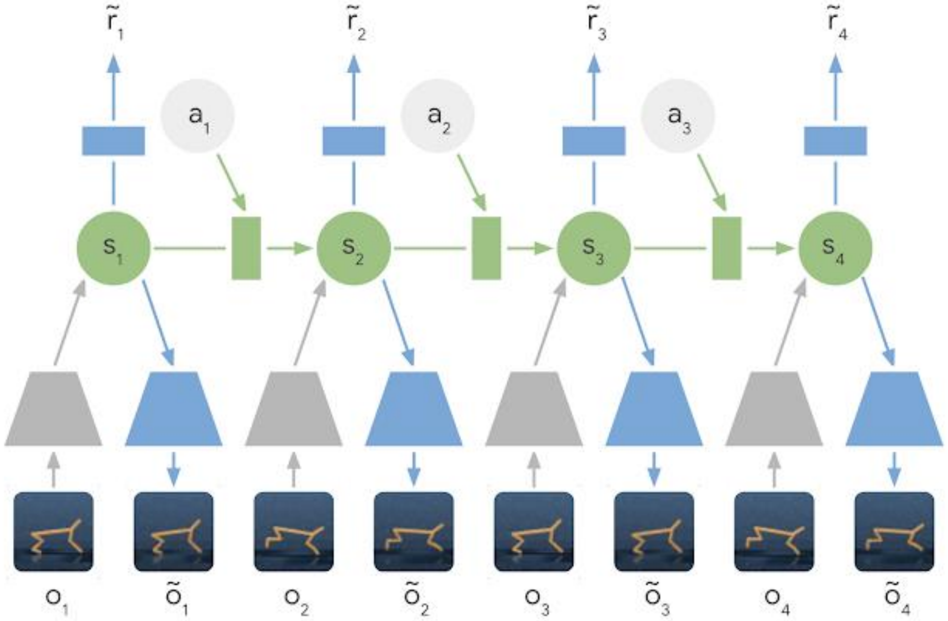
MPC

- How to solve optimization problem every timestep?

$$\arg\max_{\mathbf{a}_{0:k}} \mathbb{E}_{\tau \sim f(\tau | \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_{0:k})} [R(\tau)]$$

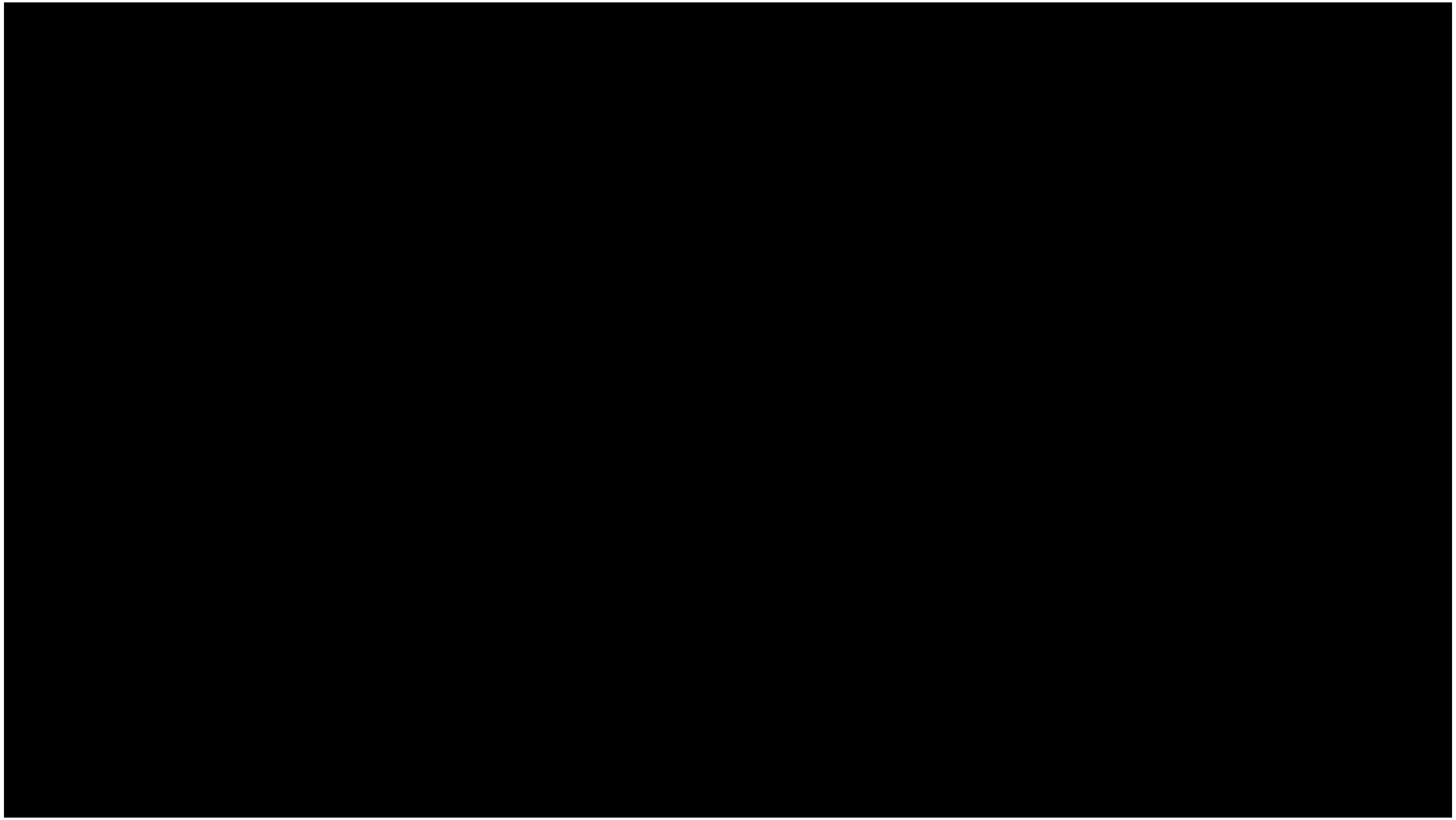
- Black-Box Optimization
    - CEM, random shooting, etc.
- If differentiable model and reward function, use gradient ascent
- Can incorporate other model-based RL improvements
    - Uncertainty estimation, ensembles, etc.

MPC



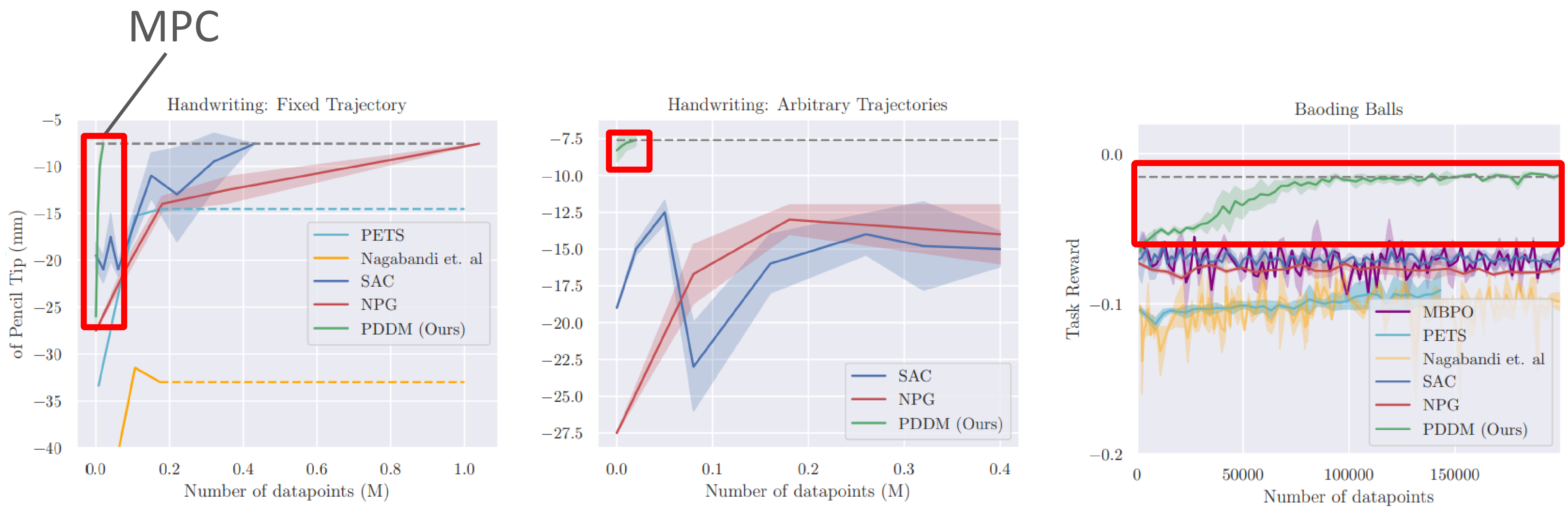
Learning Latent Dynamics for Planning from Pixels
[Hafner et al. 2019]

MPC



Deep Dynamics Models for Learning Dexterous Manipulation
[Nagabandi et al. 2019]

MPC



Deep Dynamics Models for Learning Dexterous Manipulation
[Nagabandi et al. 2019]

Model-Based RL

Policy Learning

- Learn model + policy
- Runtime policy inference is fast
- Policy is task-specific
- Typically better asymptotic performance

Online Planning

- Learn model
- Runtime planning can be slow
- Model can be task-agnostic
- May need many samples during online planning to find good plans

Summary

- Model-Based RL
- DYNA
- Model Representations
- Uncertainty Estimation
- MPC