# Model-Based RL I

## Basic Components of MBRL

Kurtland Chua

# This course so far:

# This course so far:

- Only needed ***samples*** from the transition dynamics.

# This course so far:

- Only needed ***samples*** from the transition dynamics.
  - Example: Policy gradients

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T-1} R_{t:T-1} \nabla_\theta \log \pi_\theta(s_t, a_t)$$

# This course so far:

- Only needed ***samples*** from the transition dynamics.
  - Example: Policy gradients

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T-1} R_{t:T-1} \nabla_\theta \log \pi_\theta(s_t, a_t)$$

- Wasting data! A lot of information from $(s, a, s')$ pairs that are independent of policy.

# This course so far:

- Only needed ***samples*** from the transition dynamics.
  - Example: Policy gradients

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T-1} R_{t:T-1} \nabla_\theta \log \pi_\theta(s_t, a_t)$$

- Wasting data! A lot of information from $(s, a, s')$ pairs that are independent of policy.

***Can we use*** $(s, a, s')$ ***data to model*** $p(s' \mid s, a)$***?***

# Today’s Goals

# Today's Goals

- What is model-based RL?

# Today’s Goals

- What is model-based RL?
- What are the pros and cons of these methods?

# Today's Goals

- What is model-based RL?
- What are the pros and cons of these methods?
- What kinds of models can I use?

# Today's Goals

- What is model-based RL?
- What are the pros and cons of these methods?
- What kinds of models can I use?
- How do I used a learned model to solve the RL problem?

# MBRL Pros and Cons

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.
  - Can reuse models when changing reward function.

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.
  - Can reuse models when changing reward function.
- **Pro:** Models can make more effective use of data.

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.
  - Can reuse models when changing reward function.
- **Pro:** Models can make more effective use of data.
  - Model could potentially generalize beyond data distribution.

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.
  - Can reuse models when changing reward function.
- **Pro:** Models can make more effective use of data.
  - Model could potentially generalize beyond data distribution.

- **Con:** Compounding model errors.

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.
  - Can reuse models when changing reward function.
- **Pro:** Models can make more effective use of data.
  - Model could potentially generalize beyond data distribution.

- **Con:** Compounding model errors.
  - Similar to the distribution shift problem of imitation learning.

# MBRL Pros and Cons

- **Pro:** Models are reward-function agnostic.
  - Can reuse models when changing reward function.
- **Pro:** Models can make more effective use of data.
  - Model could potentially generalize beyond data distribution.

- **Con:** Compounding model errors.
  - Similar to the distribution shift problem of imitation learning.
  - Big limiting factor!

# Part I: How to Train your Model

# How to Train your Model

# How to Train your Model

***Can we use*** $(s, a, s')$ ***data to model*** $p(s' \mid s, a)$***?***

# How to Train your Model

***Can we use*** $(s, a, s')$ ***data to model*** $p(s' \mid s, a)$***?***

- Sounds like regression? But your standard regression model only outputs a single prediction, not a distribution:

# How to Train your Model

***Can we use*** $(s, a, s')$ ***data to model*** $p(s' \mid s, a)$***?***

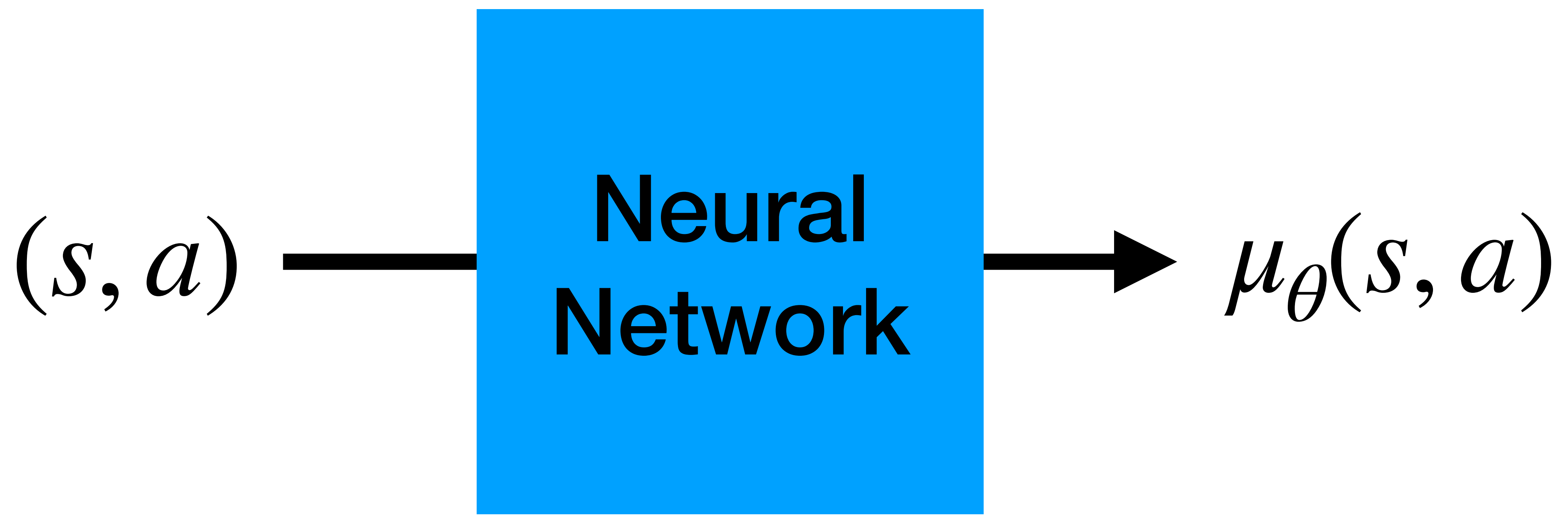
- Sounds like regression? But your standard regression model only outputs a single prediction, not a distribution:

$(s, a)$

# How to Train your Model

***Can we use*** $(s, a, s')$ ***data to model*** $p(s' \mid s, a)$***?***

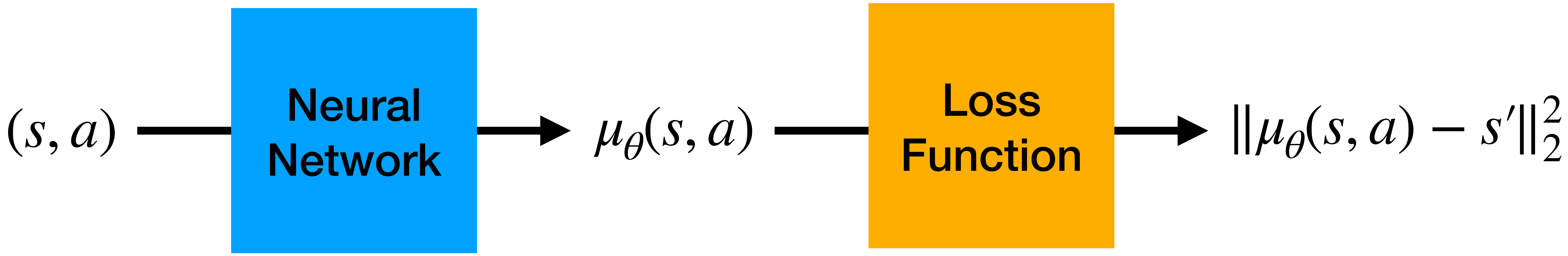
- Sounds like regression? But your standard regression model only outputs a single prediction, not a distribution:



# How to Train your Model

***Can we use*** $(s, a, s')$ ***data to model*** $p(s' \mid s, a)$***?***

- Sounds like regression? But your standard regression model only outputs a single prediction, not a distribution:



# Model Components

# Model Components

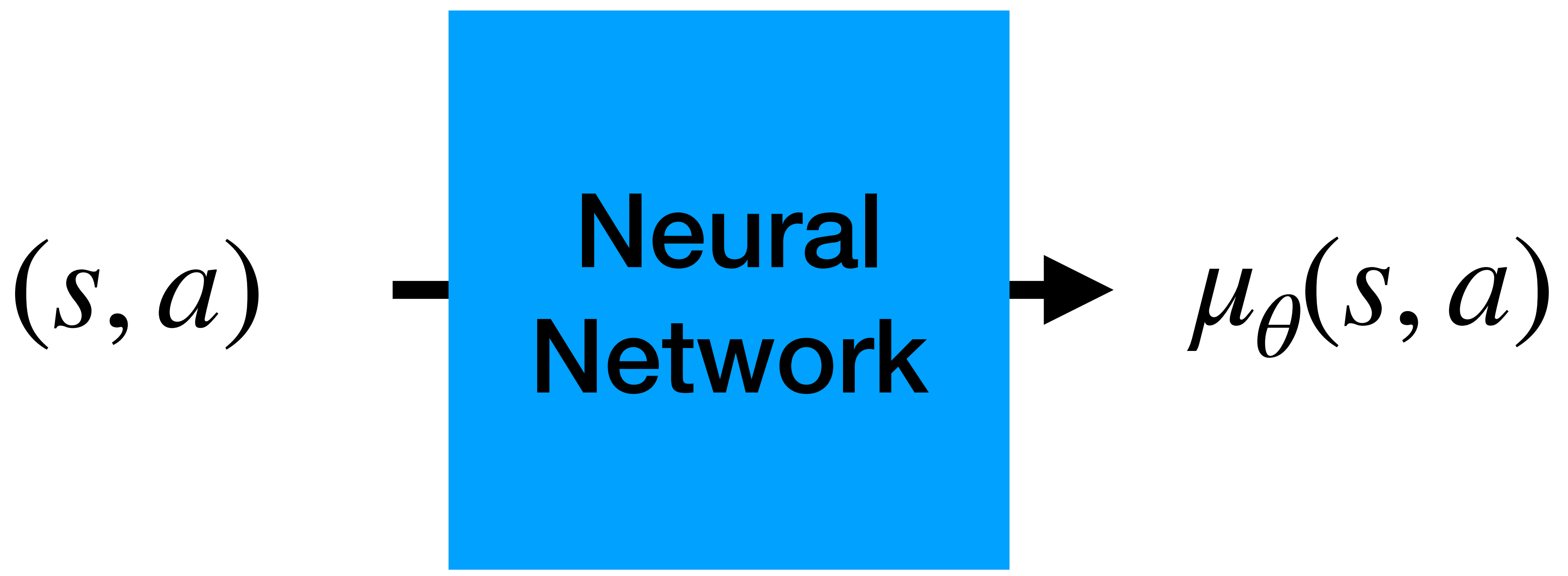
- How do we get a distributional output?

# Model Components

- How do we get a distributional output?
  - Simple idea: fix a constant $\sigma^2$, and consider a ***constant-variance Gaussian noise model:***

# Model Components

- How do we get a distributional output?
    - Simple idea: fix a constant $\sigma^2$, and consider a ***constant-variance Gaussian noise model:***



# Model Components

- How do we get a distributional output?
  - Simple idea: fix a constant $\sigma^2$, and consider a ***constant-variance Gaussian noise model:***

$(s, a)$ → **Neural Network** → $\mu_\theta(s, a)$ → **Noise Model** → $\mathcal{N}(\mu_\theta(s, a), \sigma^2)$

# Model Components

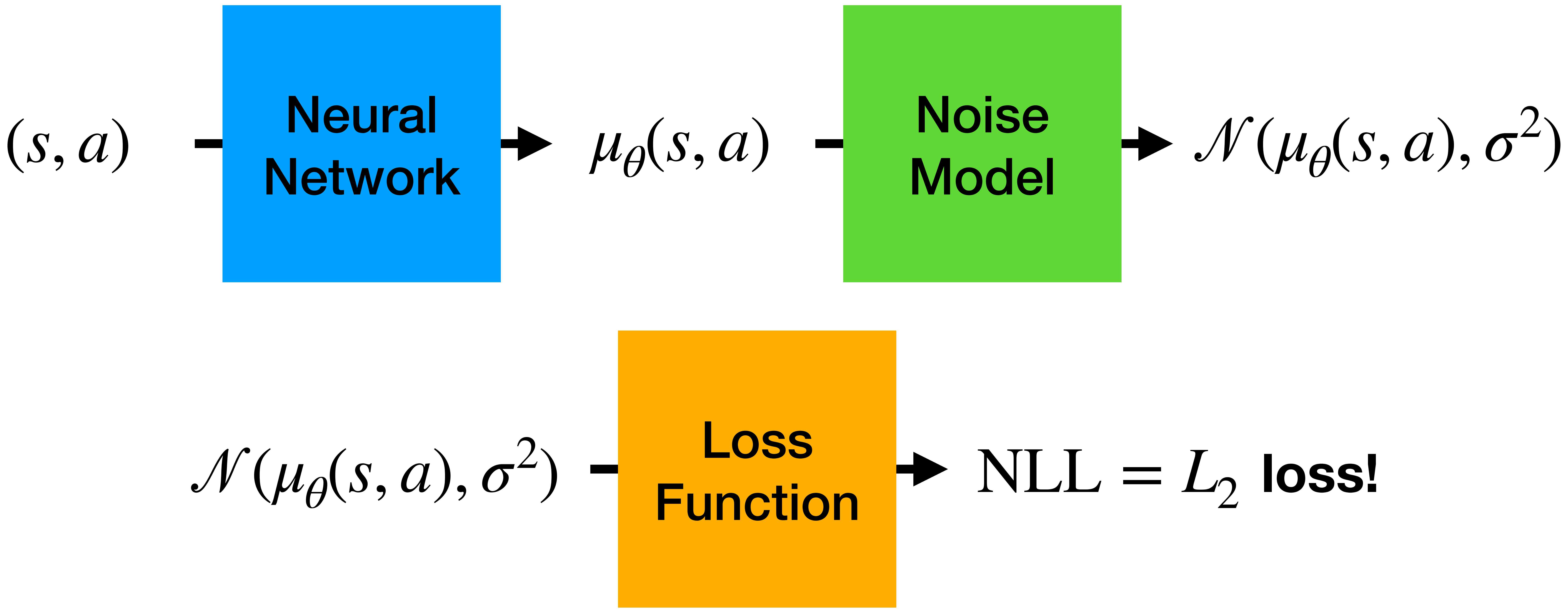
- How do we get a distributional output?
  - Simple idea: fix a constant $\sigma^2$, and consider a ***constant-variance Gaussian noise model:***

$$(s,a) \xrightarrow{\text{Neural Network}} \mu_\theta(s,a) \xrightarrow{\text{Noise Model}} \mathcal{N}(\mu_\theta(s,a), \sigma^2)$$

$$\mathcal{N}(\mu_\theta(s,a), \sigma^2)$$

# Model Components

- How do we get a distributional output?
  - Simple idea: fix a constant $\sigma^2$, and consider a ***constant-variance Gaussian noise model:***



# Model Components

# Model Components

- $L_2$ regression = MLE with a constant-variance noise model!

# Model Components

- $L_2$ regression = MLE with a constant-variance noise model!
- When modeling, need to consider three components explicitly:

# Model Components

- $L_2$ regression = MLE with a constant-variance noise model!
- When modeling, need to consider three components explicitly:

| Model Arch | Noise Model | Loss Function |
|---|---|---|
| | | |

# Model Components

- $L_2$ regression = MLE with a constant-variance noise model!
- When modeling, need to consider three components explicitly:

| **Model Arch** | **Noise Model** | **Loss Function** |
|---|---|---|
| Neural networks<br>Linear Predictors<br>Decision Trees<br>Kernel Linear Regression | | |

# Model Components

- $L_2$ regression = MLE with a constant-variance noise model!
- When modeling, need to consider three components explicitly:

| Model Arch | Noise Model | Loss Function |
|---|---|---|
| Neural networks<br>Linear Predictors<br>Decision Trees<br>Kernel Linear Regression | Constant-variance Gaussian noise model | |

# Model Components

- $L_2$ regression = MLE with a constant-variance noise model!
- When modeling, need to consider three components explicitly:

| **Model Arch** | **Noise Model** | **Loss Function** |
|---|---|---|
| Neural networks<br>Linear Predictors<br>Decision Trees<br>Kernel Linear Regression | Constant-variance Gaussian noise model | Maximum likelihood |

# Choosing Model Components, Example

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

| Model Arch | Noise Model | Loss Function |
|---|---|---|
| | | |

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

| Model Arch | Noise Model | Loss Function |
|---|---|---|
| Neural network | | |

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

| Model Arch | Noise Model | Loss Function |
|---|---|---|
| Neural network | ***Heteroskedastic Gaussian noise model*** | |

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

| Model Arch | Noise Model | Loss Function |
| --- | --- | --- |
| Neural network | ***Heteroskedastic Gaussian noise model*** | Maximum likelihood |

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

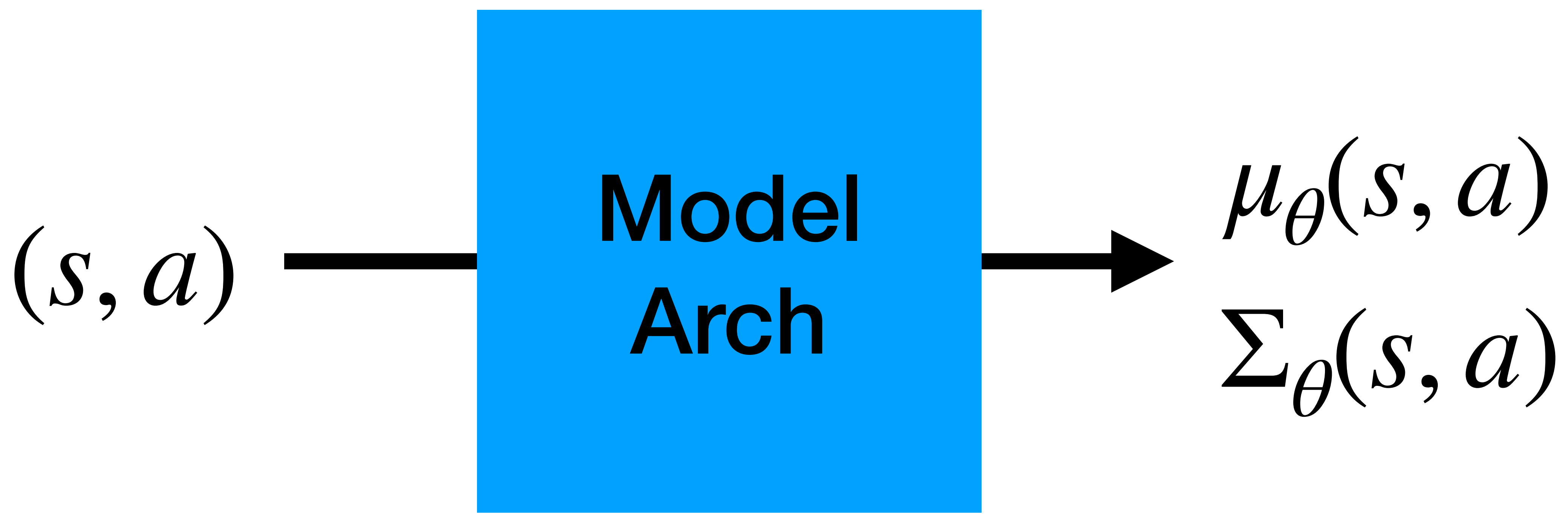
| Model Arch | Noise Model | Loss Function |
|---|---|---|
| Neural network | ***Heteroskedastic Gaussian noise model*** | Maximum likelihood |

$(s, a)$

# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

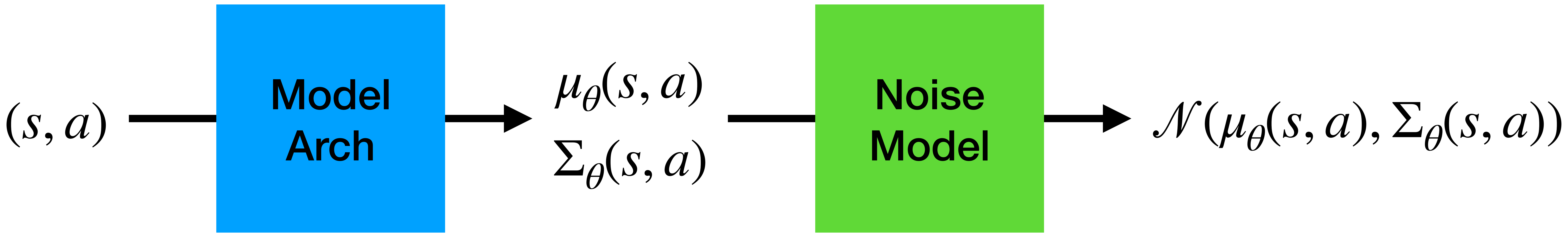
| Model Arch | Noise Model | Loss Function |
|---|---|---|
| Neural network | ***Heteroskedastic Gaussian noise model*** | Maximum likelihood |



# Choosing Model Components, Example

- Environment could be noisier in certain states/with certain actions.

| Model Arch | Noise Model | Loss Function |
|---|---|---|
| Neural network | ***Heteroskedastic Gaussian noise model*** | Maximum likelihood |



# Part II: Model Uncertainty

# Part II: Model Uncertainty

# Part II: Model Uncertainty

- Problem: Model subject to compounding errors.

# Part II: Model Uncertainty

- Problem: Model subject to compounding errors.
  - Would be nice if the model could tell us if it is “uncertain” about its prediction.

# Part II: Model Uncertainty

- Problem: Model subject to compounding errors.
  - Would be nice if the model could tell us if it is “uncertain” about its prediction.

- What is “uncertainty”?

# Part II: Model Uncertainty

- Problem: Model subject to compounding errors.
  - Would be nice if the model could tell us if it is “uncertain” about its prediction.

- What is “uncertainty”?
  - Leading question: is learning a noise model quantifying uncertainty?

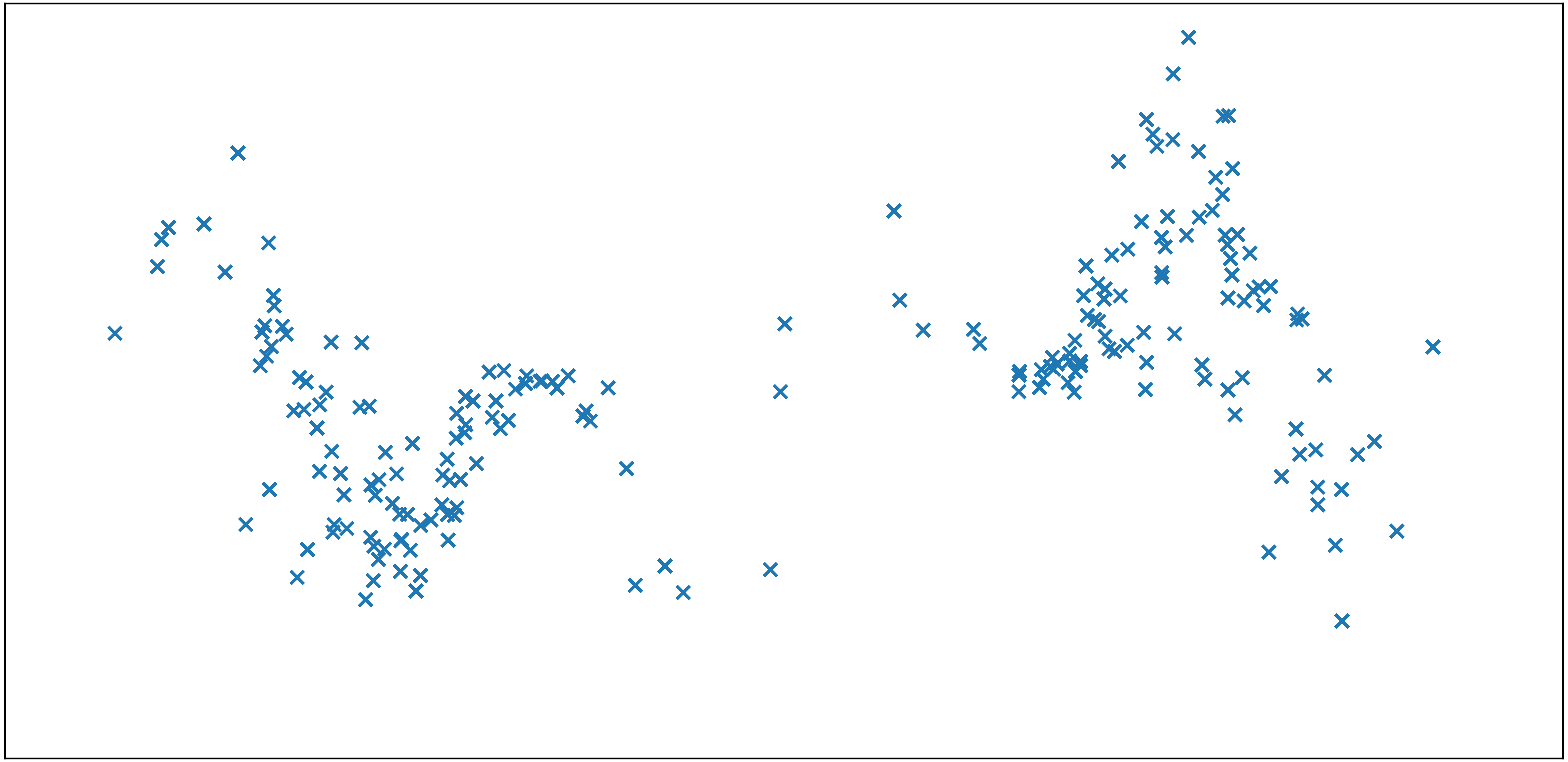
# Part II: Model Uncertainty

- Problem: Model subject to compounding errors.
  - Would be nice if the model could tell us if it is “uncertain” about its prediction.

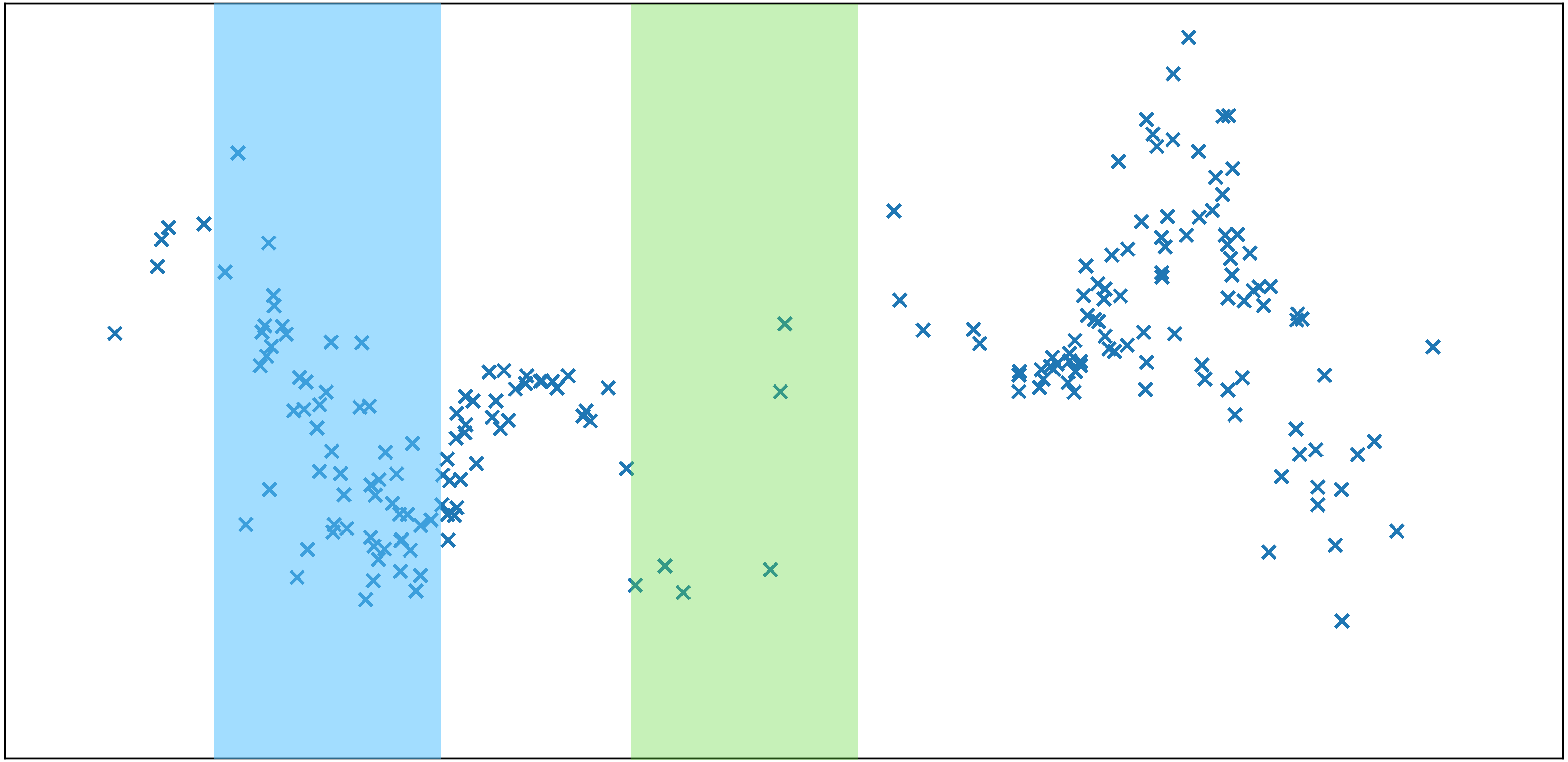
- What is “uncertainty”?
  - Leading question: is learning a noise model quantifying uncertainty?
    - Yes! **But not the one we necessarily care about!**

# Uncertainty Intuition Primer

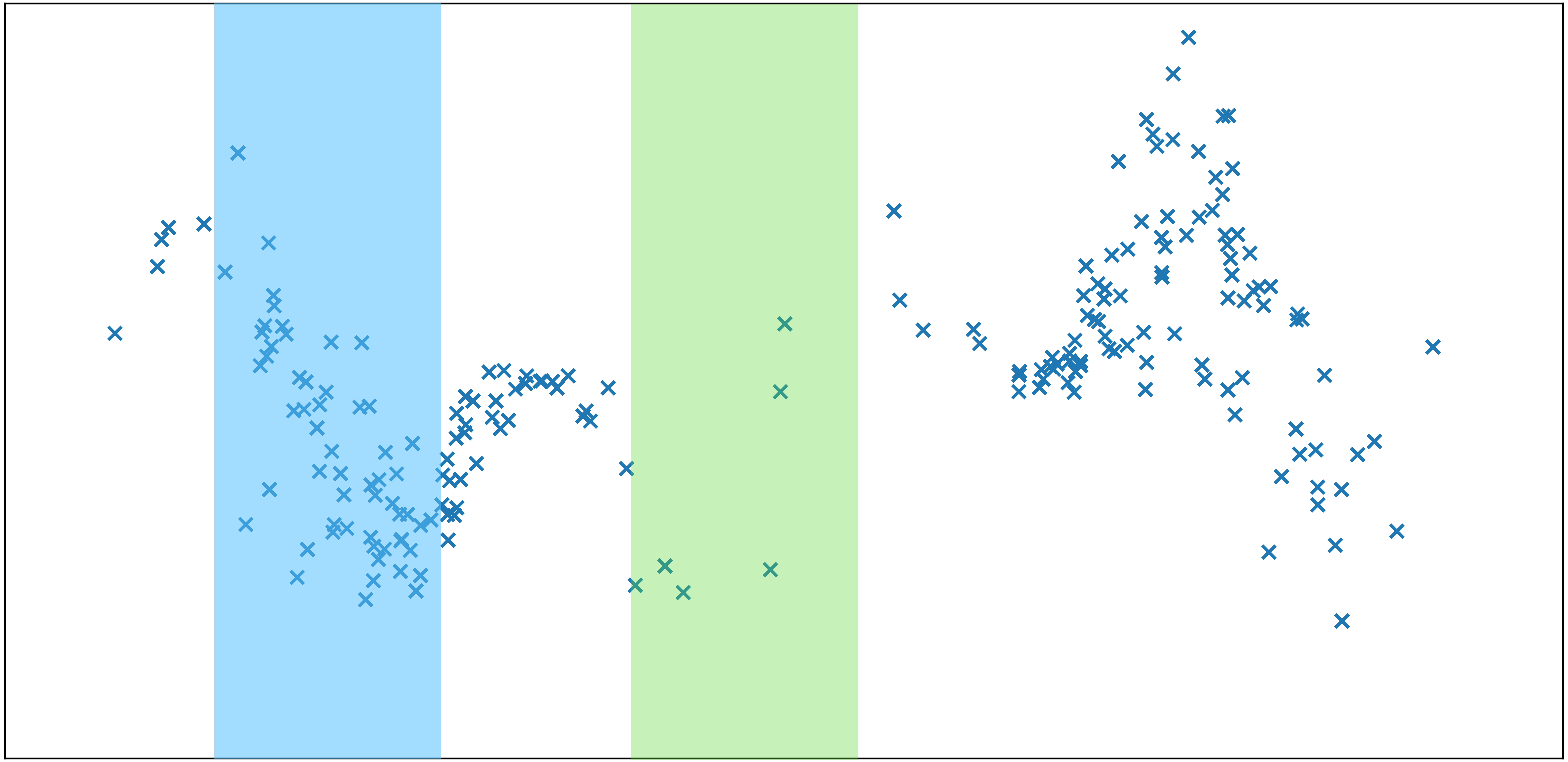
# Uncertainty Intuition Primer



# Uncertainty Intuition Primer



# Uncertainty Intuition Primer



Where do we need more data?

# Bias-Variance Decomposition Review

# Bias-Variance Decomposition Review

- Setting:

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathscr{D}}$

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathscr{D}}$
  - Want to evaluate error on a fixed test input $x$.

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathscr{D}}$
  - Want to evaluate error on a fixed test input $x$.
    - Label generated as $f^*(x) + \epsilon_x$ (noise distribution depends on $x$).

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathcal{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathcal{D}}$
  - Want to evaluate error on a fixed test input $x$.
    - Label generated as $f^*(x) + \epsilon_x$ (noise distribution depends on $x$).

- Bias-variance decomposition:

$$\mathbb{E}_{\mathcal{D},\epsilon_x}\left[(y - \hat{f}_{\mathcal{D}}(x))^2\right] = \left(f^*(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]\right)^2 + \mathrm{Var}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(x)\right] + \mathbb{E}[\epsilon_x^2]$$

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathscr{D}}$
  - Want to evaluate error on a fixed test input $x$.
    - Label generated as $f^*(x) + \epsilon_x$ (noise distribution depends on $x$).

- Bias-variance decomposition:

$$\mathbb{E}_{\mathscr{D},\epsilon_x}\left[(y - \hat{f}_{\mathscr{D}}(x))^2\right] = \underbrace{\left(f^*(x) - \mathbb{E}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]\right)^2}_{\text{Bias}} + \mathrm{Var}_{\mathscr{D}}\left[\hat{f}_{\mathscr{D}}(x)\right] + \mathbb{E}[\epsilon_x^2]$$

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathscr{D}}$
  - Want to evaluate error on a fixed test input $x$.
    - Label generated as $f^*(x) + \epsilon_x$ (noise distribution depends on $x$).

- Bias-variance decomposition:

$$\mathbb{E}_{\mathscr{D},\epsilon_x}\left[(y - \hat{f}_{\mathscr{D}}(x))^2\right] = \underbrace{\left(f^*(x) - \mathbb{E}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]\right)^2}_{\text{Bias}} + \underbrace{\mathrm{Var}_{\mathscr{D}}\left[\hat{f}_{\mathscr{D}}(x)\right]}_{\text{Variance}} + \mathbb{E}[\epsilon_x^2]$$

# Bias-Variance Decomposition Review

- Setting:
  - Randomly sampled dataset $\mathscr{D}$.
  - Some training procedure to get predictor $\hat{f}_{\mathscr{D}}$
  - Want to evaluate error on a fixed test input $x$.
    - Label generated as $f^*(x) + \epsilon_x$ (noise distribution depends on $x$).

- Bias-variance decomposition:

$$\mathbb{E}_{\mathscr{D},\epsilon_x}\left[(y-\hat{f}_{\mathscr{D}}(x))^2\right] = \underbrace{\left(f^*(x) - \mathbb{E}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]\right)^2}_{\text{Bias}} + \underbrace{\mathrm{Var}_{\mathscr{D}}\left[\hat{f}_{\mathscr{D}}(x)\right]}_{\text{Variance}} + \underbrace{\mathbb{E}[\epsilon_x^2]}_{\text{Irreducible}}$$

# Bias-Variance Decomposition in MBRL

$$\mathbb{E}_{\mathcal{D},\epsilon_x}\left[(y-\hat{f}_{\mathcal{D}}(x))^2\right] = \underbrace{\left(f^*(x)-\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]\right)^2}_{\text{Bias}} + \underbrace{\mathrm{Var}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(x)\right]}_{\text{Variance}} + \underbrace{\mathbb{E}[\epsilon_x^2]}_{\text{Irreducible}}$$

# Bias-Variance Decomposition in MBRL

$$\mathbb{E}_{\mathscr{D},\epsilon_x}\left[(y - \hat{f}_{\mathscr{D}}(x))^2\right] = \underbrace{\left(f^*(x) - \mathbb{E}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]\right)^2}_{\text{Bias}} + \underbrace{\mathrm{Var}_{\mathscr{D}}\left[\hat{f}_{\mathscr{D}}(x)\right]}_{\text{Variance}} + \underbrace{\mathbb{E}[\epsilon_x^2]}_{\text{Irreducible}}$$

- Bias: Error from choosing a model class that cannot capture the real dynamics.

# Bias-Variance Decomposition in MBRL

$$\mathbb{E}_{\mathcal{D},\epsilon_x}\left[(y - \hat{f}_{\mathcal{D}}(x))^2\right] = \underbrace{\left(f^*(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]\right)^2}_{\text{Bias}} + \underbrace{\mathrm{Var}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(x)\right]}_{\text{Variance}} + \underbrace{\mathbb{E}[\epsilon_x^2]}_{\text{Irreducible}}$$

- Bias: Error from choosing a model class that cannot capture the real dynamics.
- Variance: How much the prediction varies over different possible datasets.

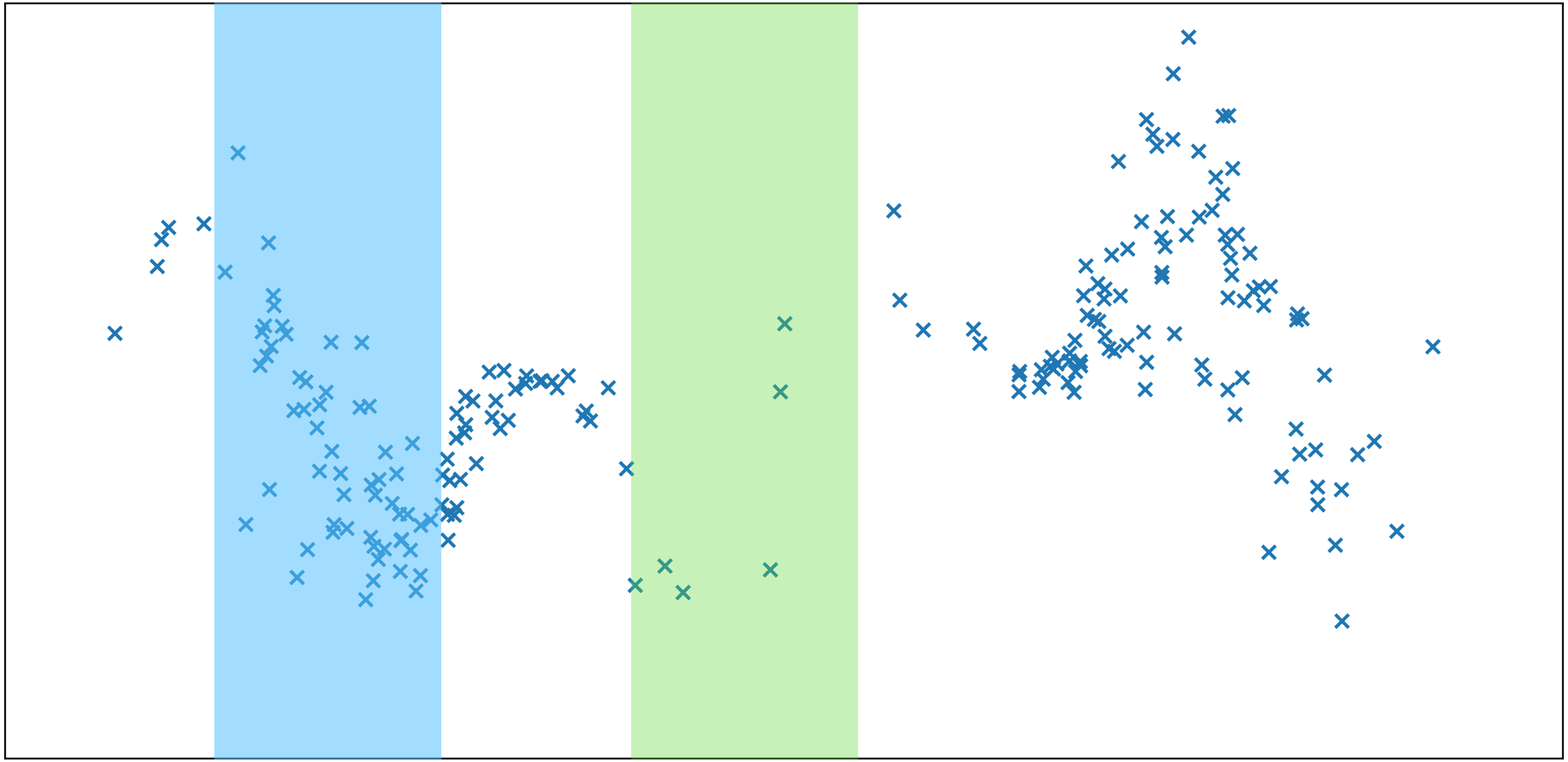
# Bias-Variance Decomposition in MBRL

$$\mathbb{E}_{\mathcal{D},\epsilon_x}\left[(y-\hat{f}_{\mathcal{D}}(x))^2\right] = \underbrace{\left(f^*(x)-\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]\right)^2}_{\text{Bias}} + \underbrace{\mathrm{Var}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(x)\right]}_{\text{Variance}} + \underbrace{\mathbb{E}[\epsilon_x^2]}_{\text{Irreducible}}$$

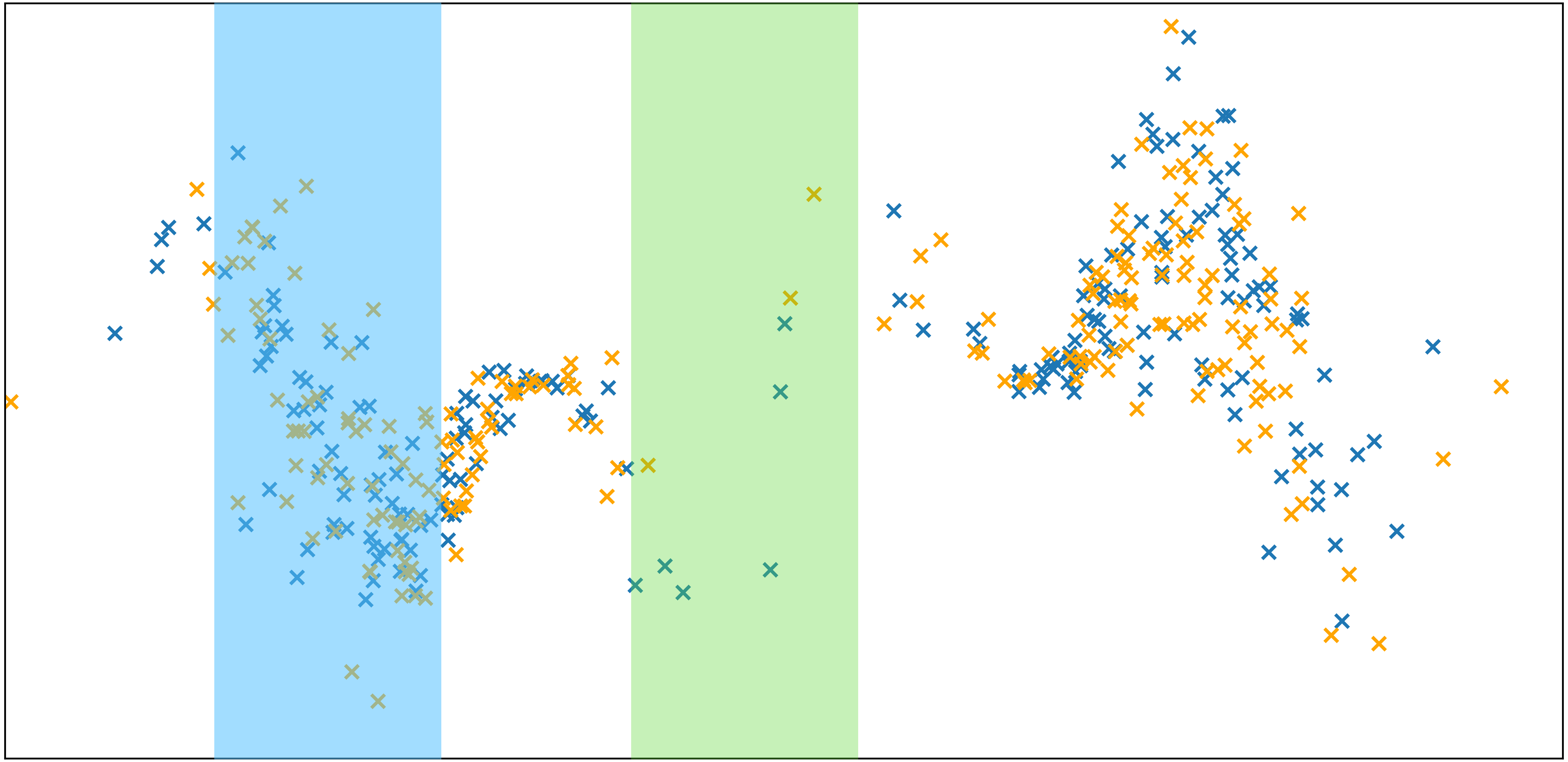
- Bias: Error from choosing a model class that cannot capture the real dynamics.
- Variance: How much the prediction varies over different possible datasets.
- Irreducible Error: Inherent environment noise.

# Variance = “Addressable” Uncertainty

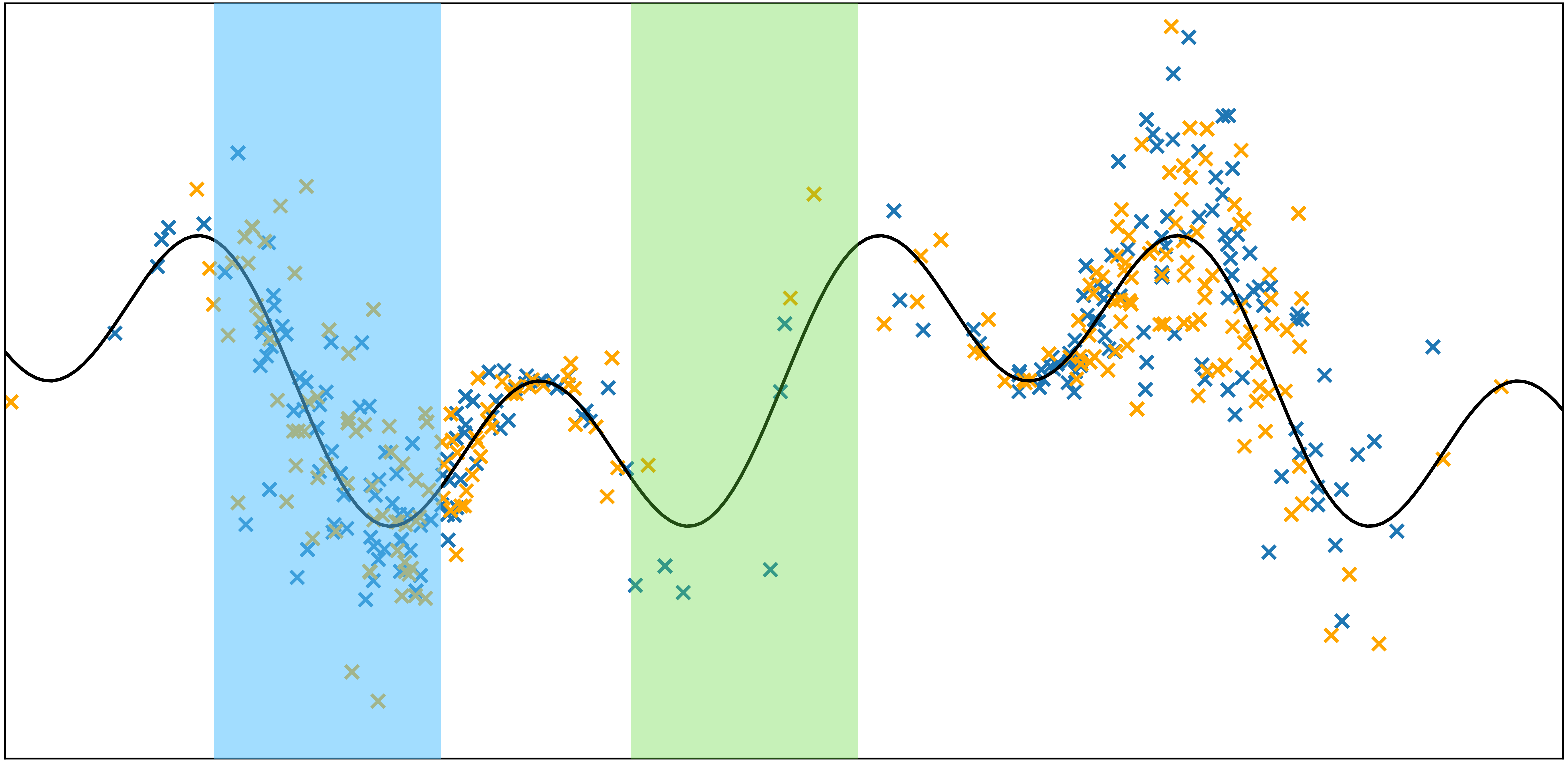
# Variance = “Addressable” Uncertainty



# Variance = “Addressable” Uncertainty



# Variance = “Addressable” Uncertainty



# Measuring the Variance

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(x)]$.

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]$.
  - How?? We only have one dataset.

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]$.
  - How?? We only have one dataset.
- Trick from classical statistics: pretend the uniform distribution over $\mathscr{D}$ is a good proxy for the real dataset-generating distribution!

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]$.
  - How?? We only have one dataset.
- Trick from classical statistics: pretend the uniform distribution over $\mathscr{D}$ is a good proxy for the real dataset-generating distribution!
  - Known as ***bootstrapping***.

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]$.
  - How?? We only have one dataset.
- Trick from classical statistics: pretend the uniform distribution over $\mathscr{D}$ is a good proxy for the real dataset-generating distribution!
  - Known as ***bootstrapping***.
    1. Sample datasets $\mathscr{D}_1, \ldots, \mathscr{D}_m$ from $\mathscr{D}$ with replacement.

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]$.
  - How?? We only have one dataset.
- Trick from classical statistics: pretend the uniform distribution over $\mathscr{D}$ is a good proxy for the real dataset-generating distribution!
  - Known as ***bootstrapping***.
    1. Sample datasets $\mathscr{D}_1, \ldots, \mathscr{D}_m$ from $\mathscr{D}$ with replacement.
    2. Train an ***ensemble of models*** $\hat{p}_i := \hat{p}_{\mathscr{D}_i}$ for $i = 1, \ldots, m$.

# Measuring the Variance

- We want to estimate $\mathrm{Var}_{\mathscr{D}}[\hat{f}_{\mathscr{D}}(x)]$.
  - How?? We only have one dataset.
- Trick from classical statistics: pretend the uniform distribution over $\mathscr{D}$ is a good proxy for the real dataset-generating distribution!
  - Known as ***bootstrapping***.
    1. Sample datasets $\mathscr{D}_1, \ldots, \mathscr{D}_m$ from $\mathscr{D}$ with replacement.
    2. Train an ***ensemble of models*** $\hat{p}_i := \hat{p}_{\mathscr{D}_i}$ for $i = 1, \ldots, m$.

**Key Takeaway: Train an ensemble of models to allow the agent to quantify "addressable" uncertainty when it needs to.**

# Part III: Trajectory Prediction

# Trajectory Prediction

# Trajectory Prediction

- RL algorithms all model the effect of applying a policy to an environment

# Trajectory Prediction

- RL algorithms all model the effect of applying a policy to an environment
  - e.g. Q-functions model the return of a particular policy.

# Trajectory Prediction

- RL algorithms all model the effect of applying a policy to an environment
  - e.g. Q-functions model the return of a particular policy.
  - With models, we can go beyond modeling returns to modeling whole trajectories.

# Trajectory Prediction

- RL algorithms all model the effect of applying a policy to an environment
  - e.g. Q-functions model the return of a particular policy.
  - With models, we can go beyond modeling returns to modeling whole trajectories.

- Problem: Given an action sequence, a given starting state, and a model, how do we predict the resulting trajectory?

# Trajectory Prediction

# Trajectory Prediction

- Problem: Given an action sequence, a given starting state, and a model, how do we predict the resulting trajectory?

# Trajectory Prediction

- Problem: Given an action sequence, a given starting state, and a model, how do we predict the resulting trajectory?

- Need to incorporate two aspects of our trained models:

# Trajectory Prediction

- Problem: Given an action sequence, a given starting state, and a model, how do we predict the resulting trajectory?

- Need to incorporate two aspects of our trained models:
  - We have an ensemble of models, each representing a distinct “belief” of the correct dynamics.

# Trajectory Prediction

- Problem: Given an action sequence, a given starting state, and a model, how do we predict the resulting trajectory?

- Need to incorporate two aspects of our trained models:
  - We have an ensemble of models, each representing a distinct “belief” of the correct dynamics.
  - Each model in the ensemble models inherent environment noise.

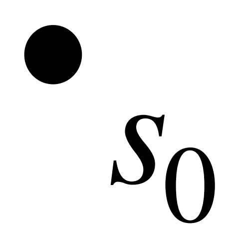
# Trajectory Prediction Approach

# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:

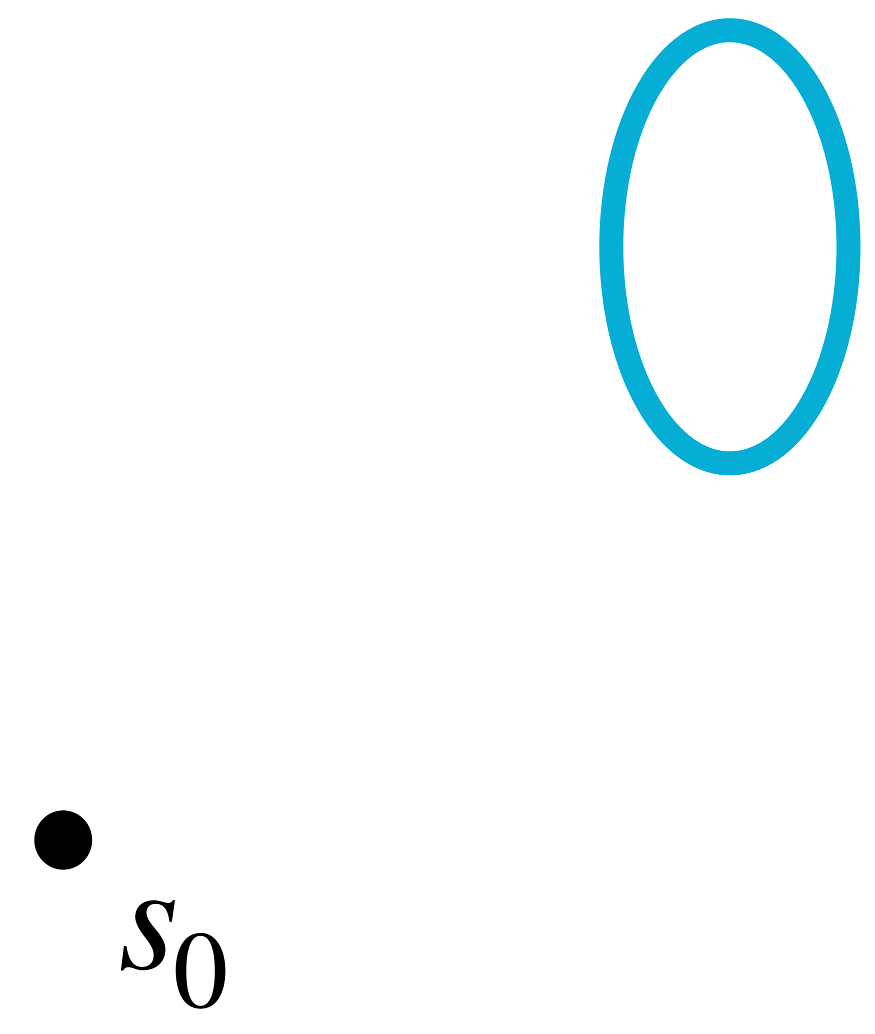
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1, \hat{p}_2, \hat{p}_3$:



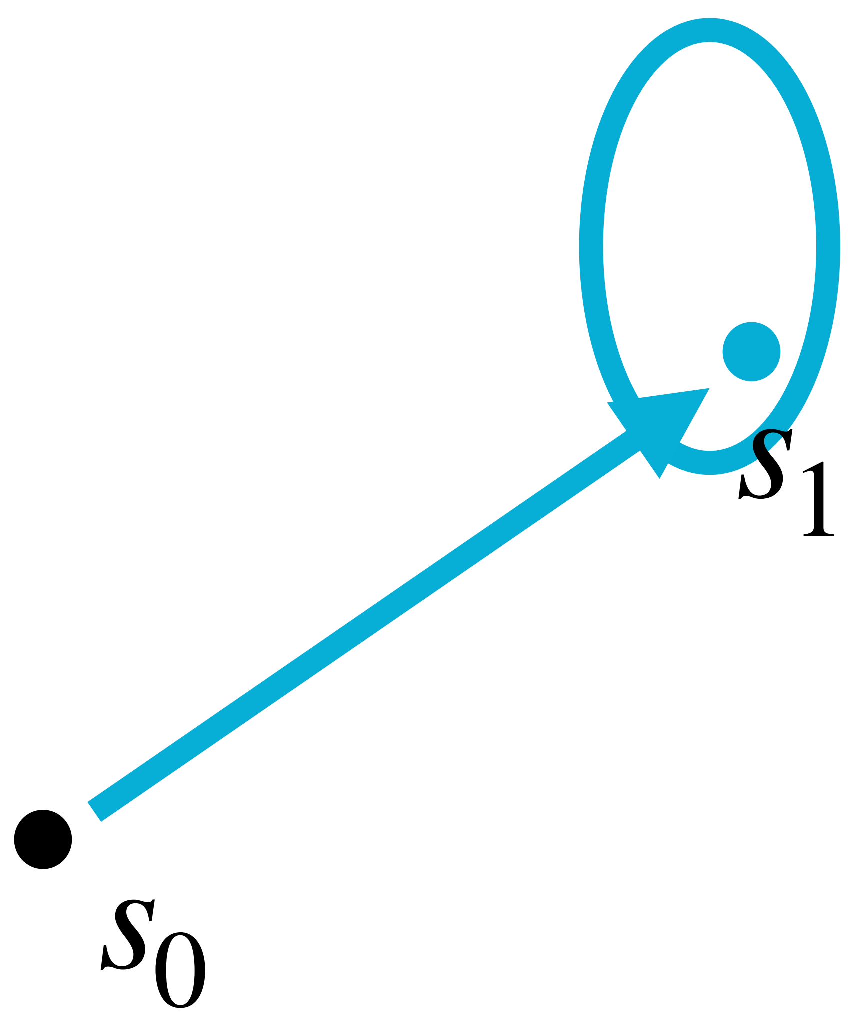
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



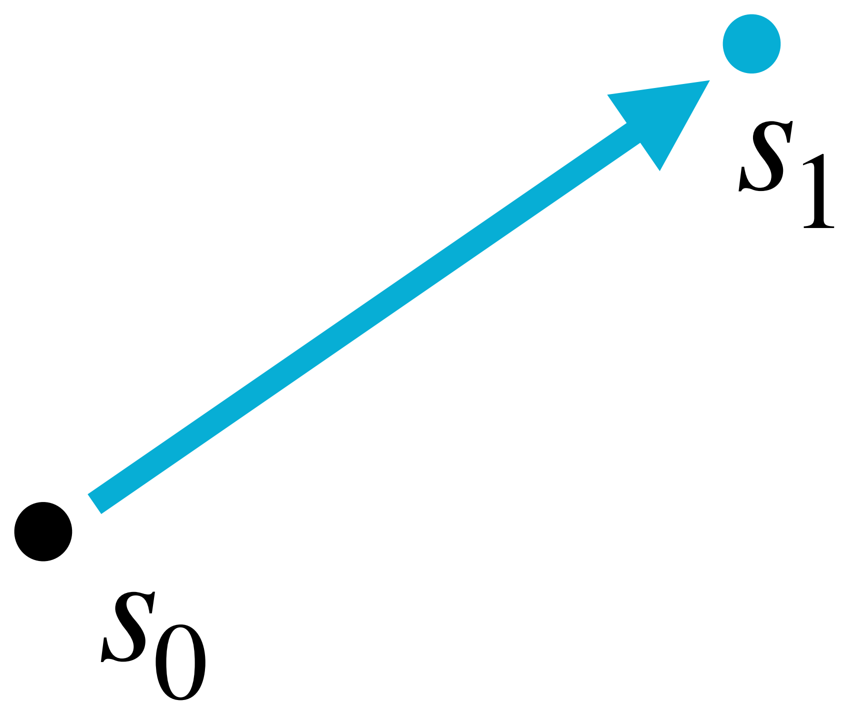
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



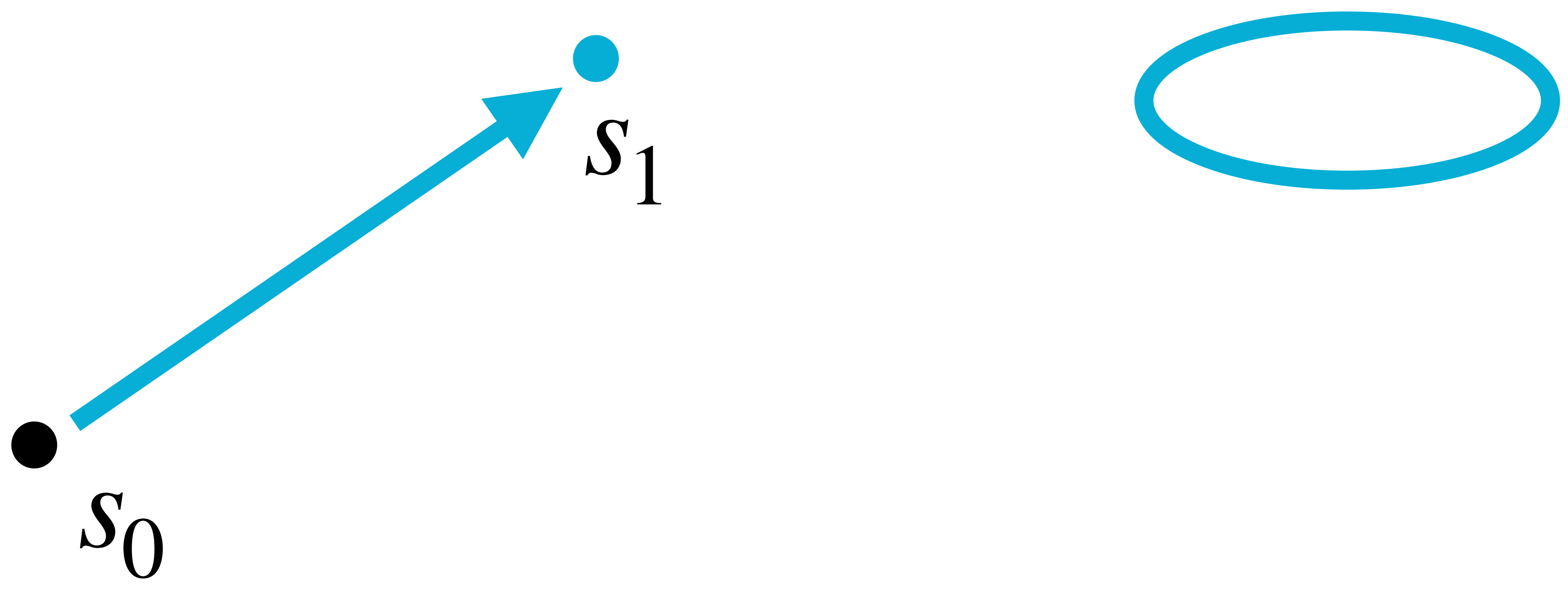
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



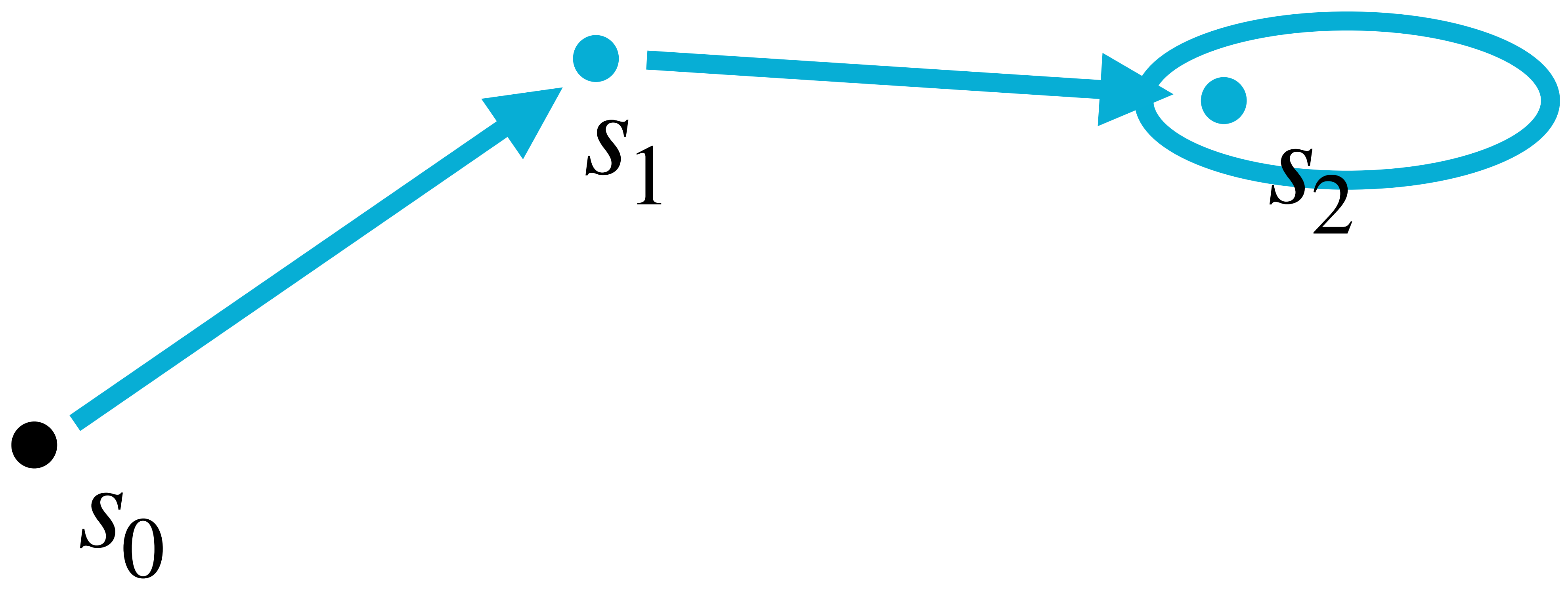
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1, \hat{p}_2, \hat{p}_3$:



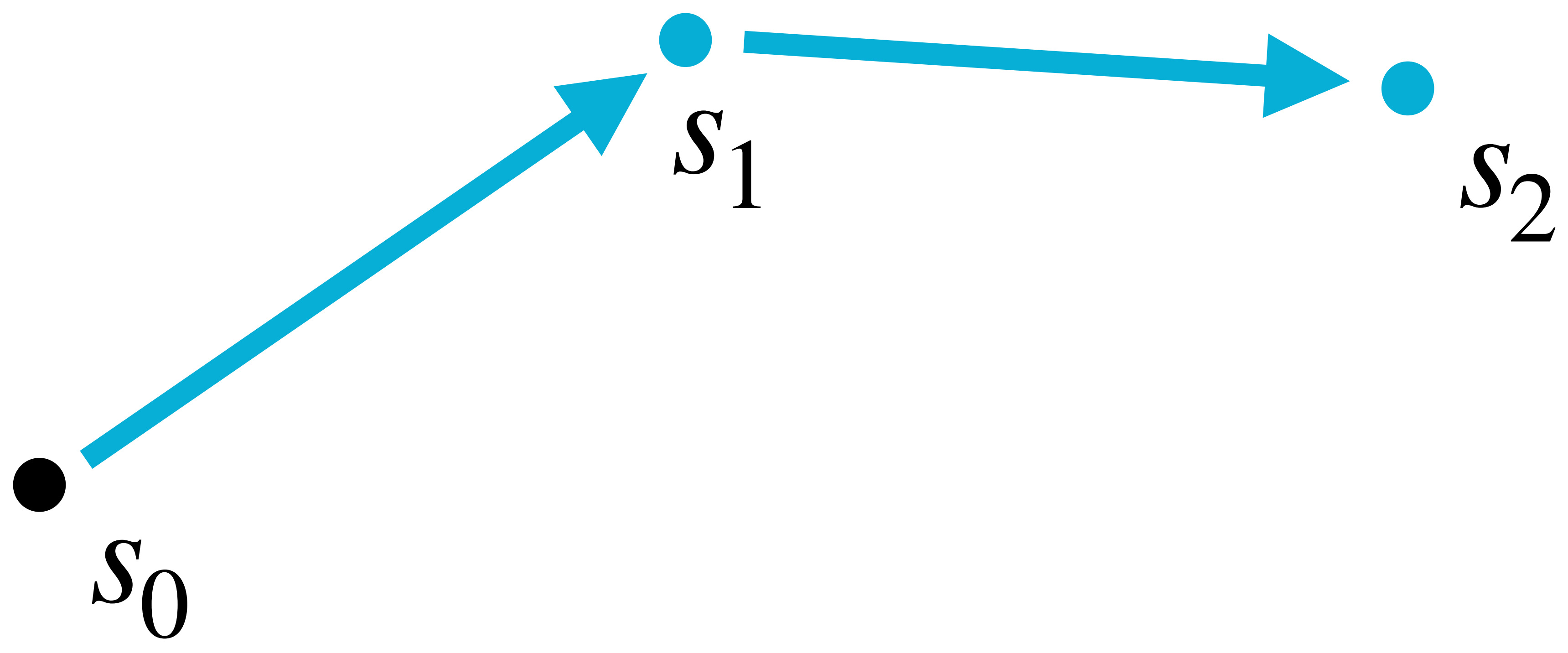
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



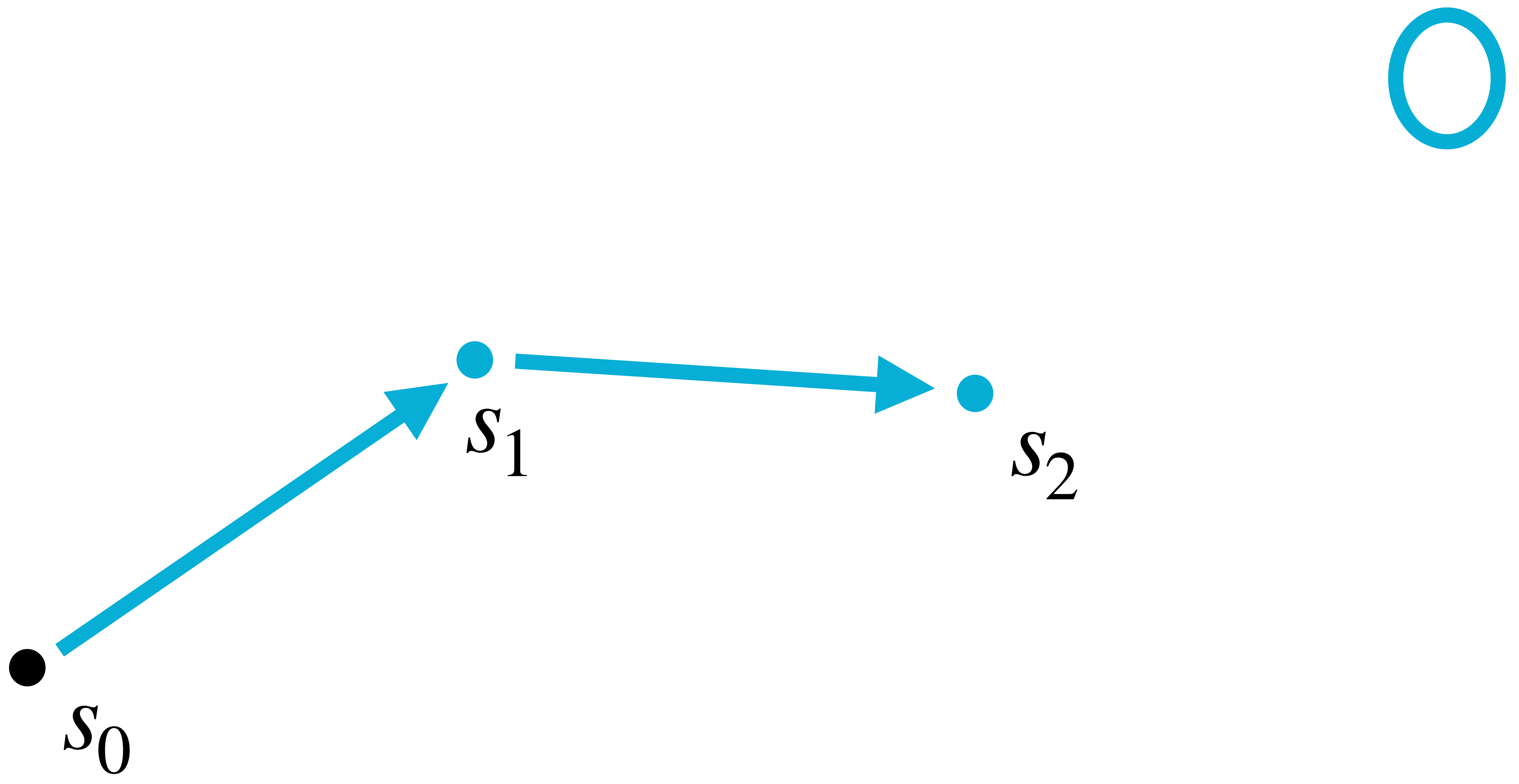
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



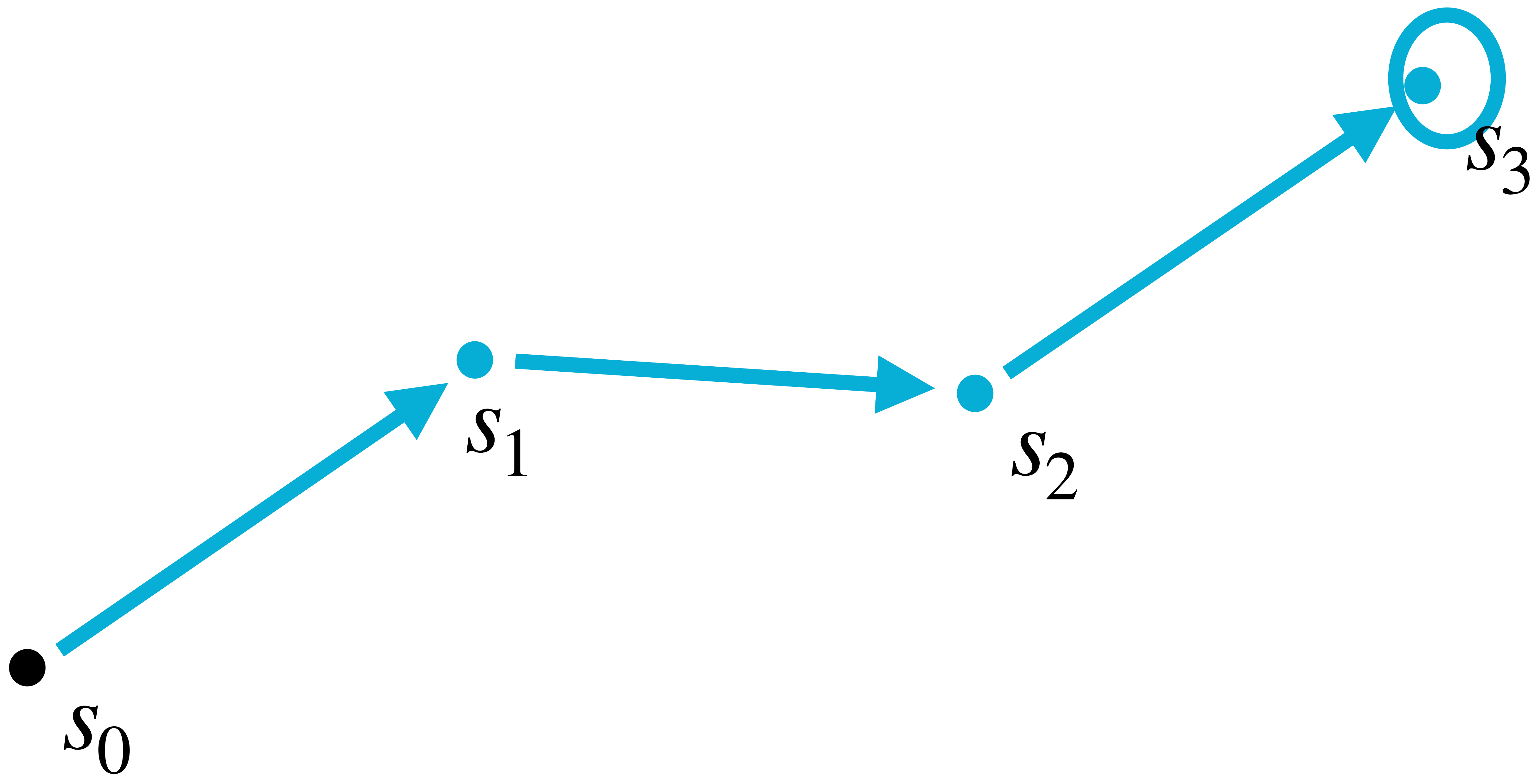
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



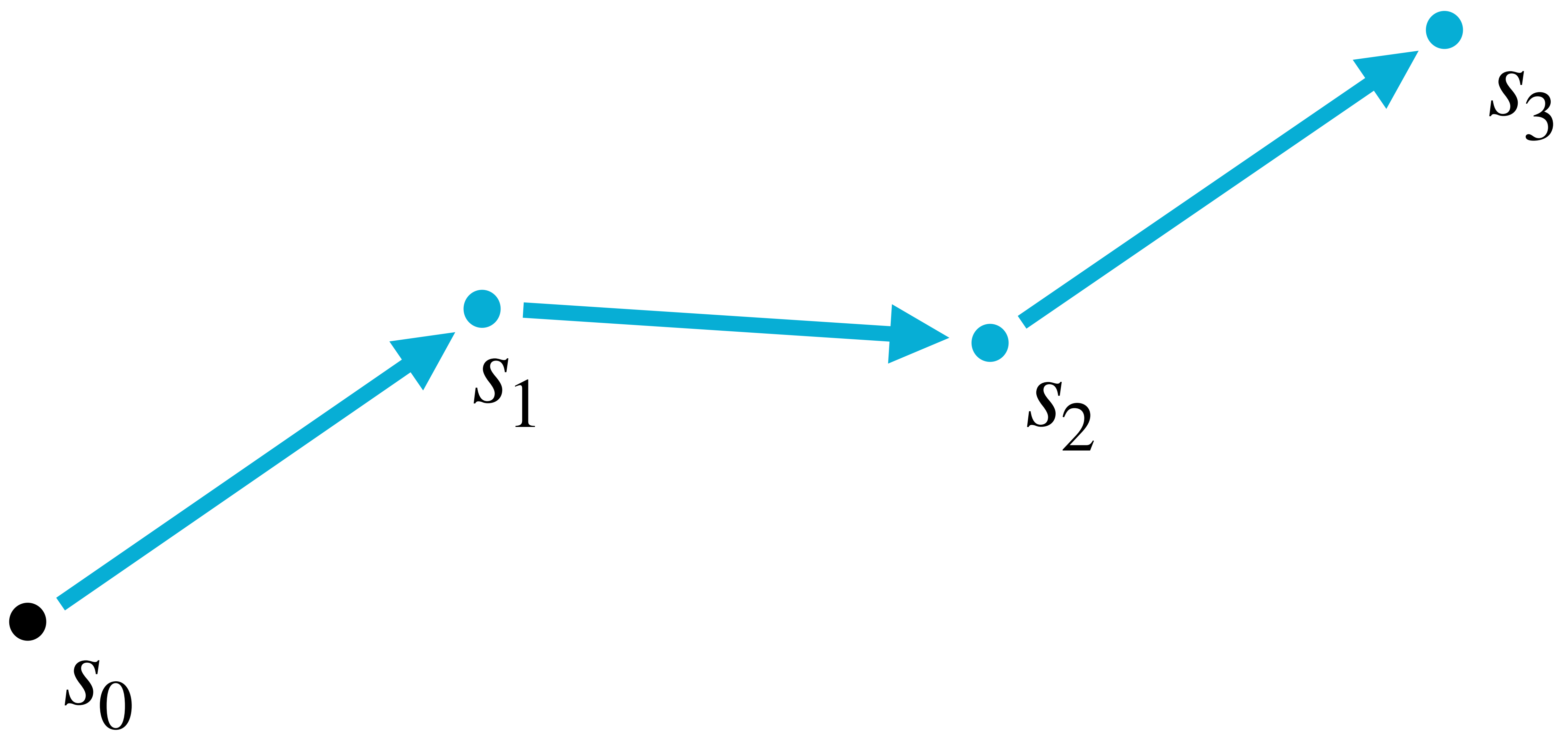
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



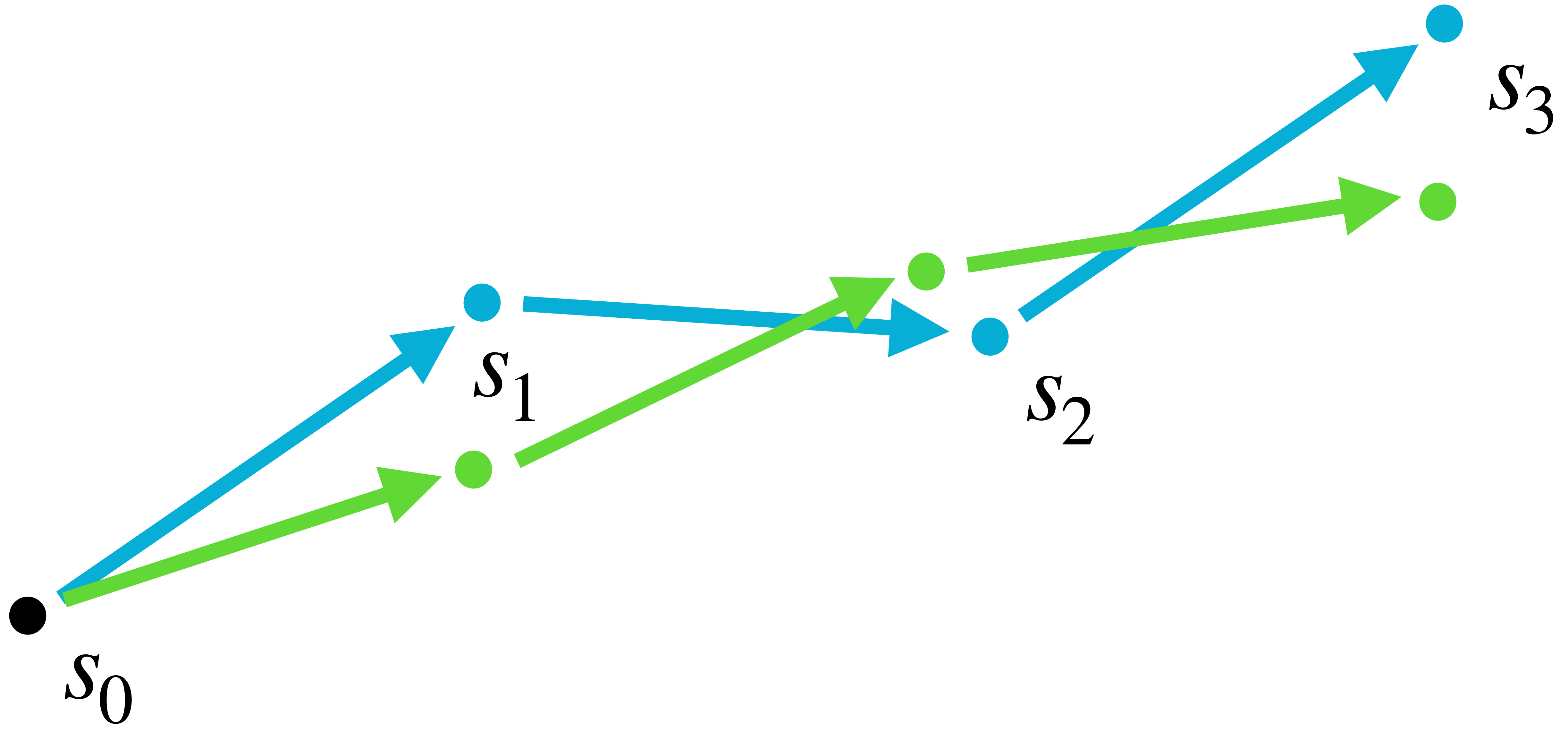
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



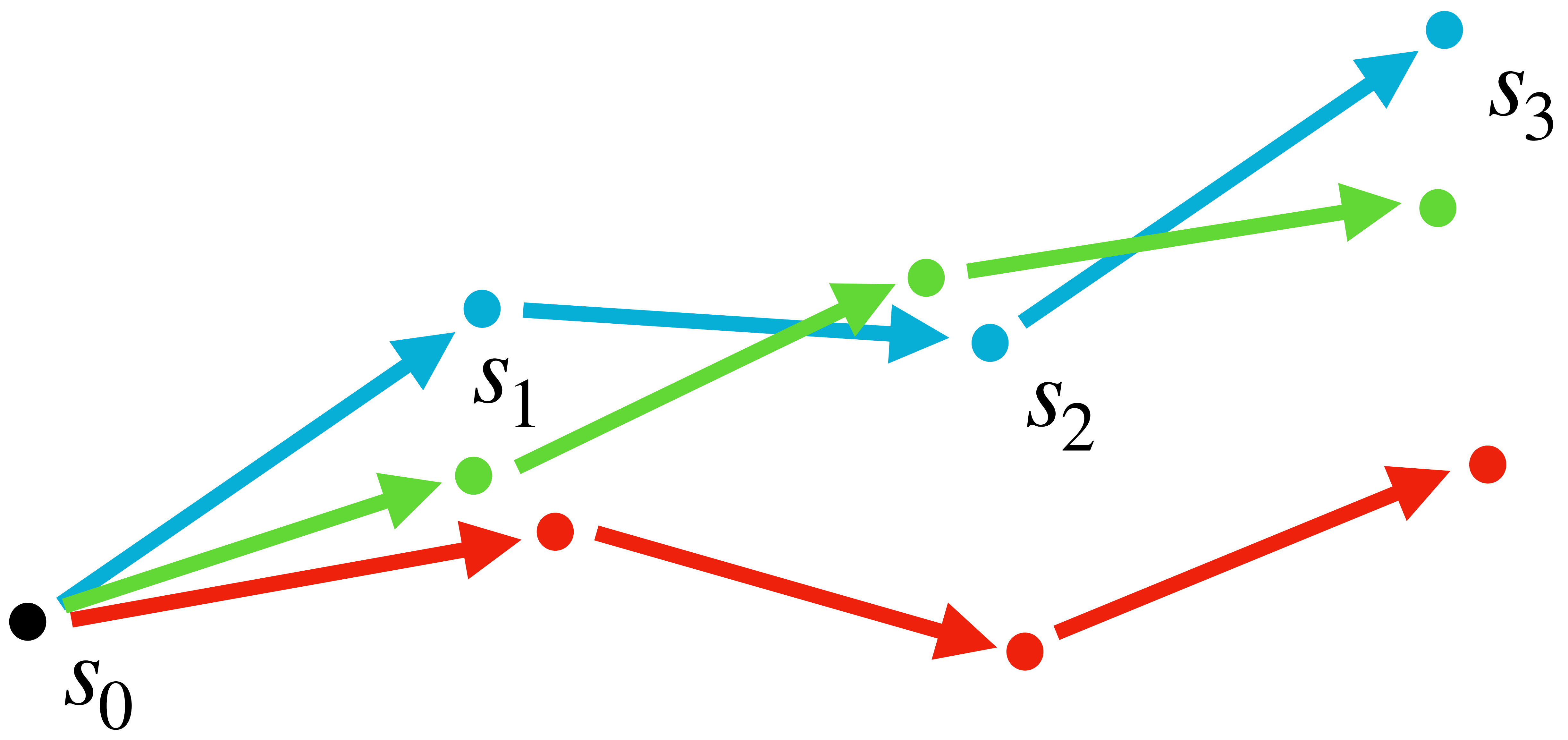
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



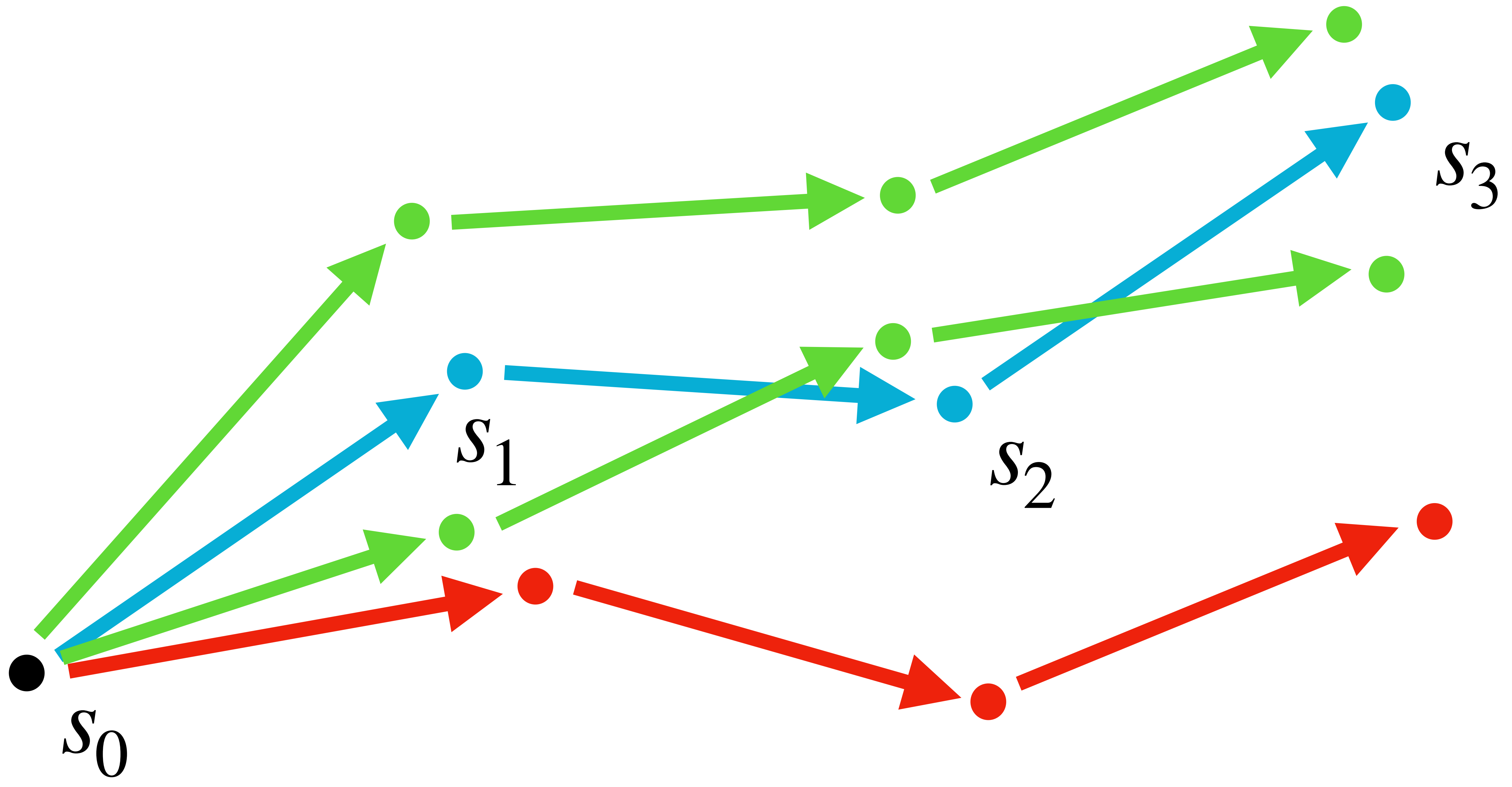
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



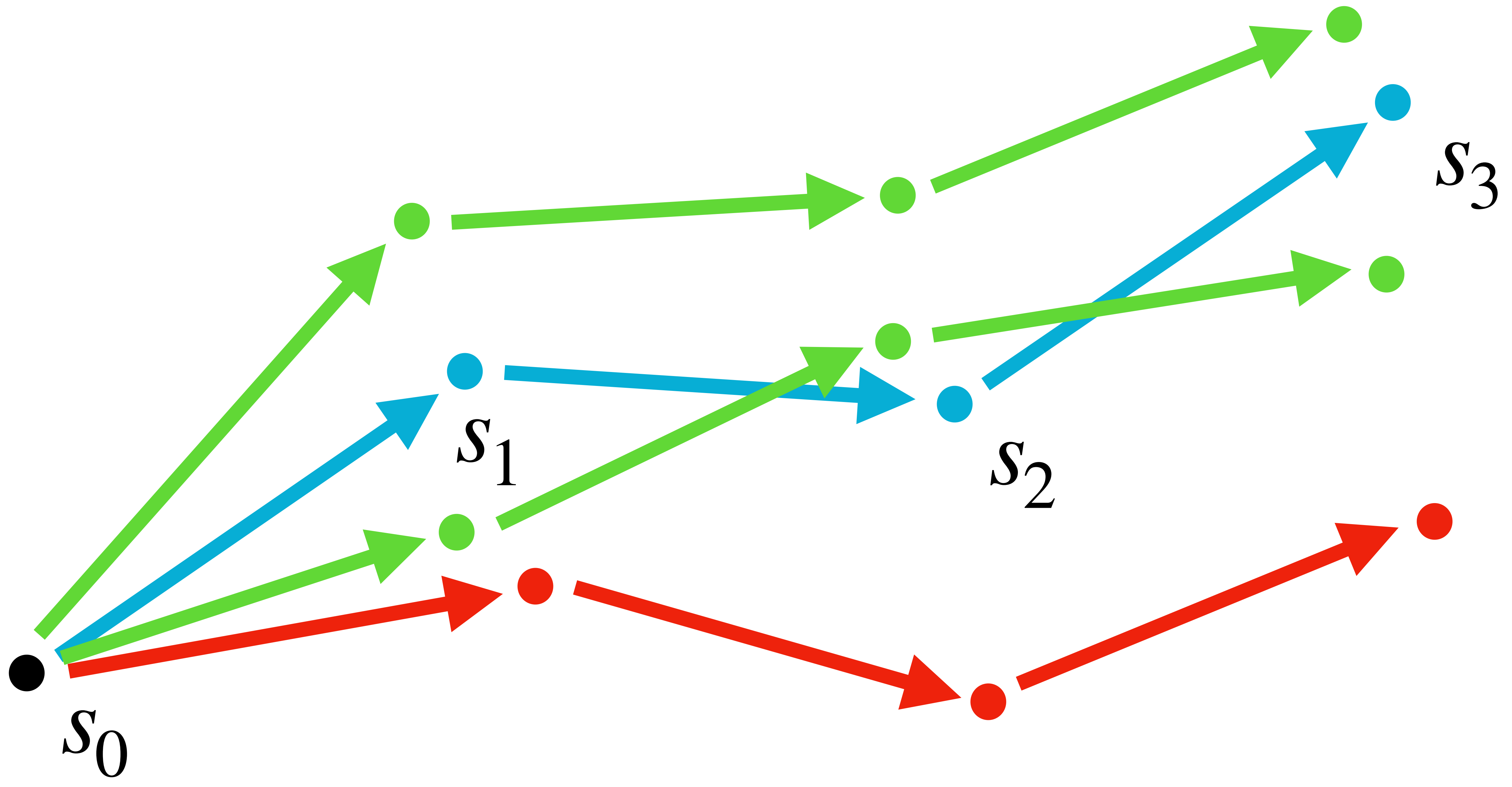
# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1$, $\hat{p}_2$, $\hat{p}_3$:



# Trajectory Prediction Approach

- Given starting state $s_0$, action sequence $(a_0, a_1, a_2)$, models $\hat{p}_1, \hat{p}_2, \hat{p}_3$:




Known as ***particle-based sampling***

# Part IV: From Prediction to Control

# Simple Idea: Open-Loop Control

# Simple Idea: Open-Loop Control

- Sample a bunch of action sequences, check which one does the best, and use that sequence for the entire episode!

# Simple Idea: Open-Loop Control

- Sample a bunch of action sequences, check which one does the best, and use that sequence for the entire episode!

---

**Algorithm 2** Open-Loop Control

---

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: Sample i.i.d. action sequences $\left\{(a_0^{(i)}, a_1^{(i)}, \ldots, a_{T-1}^{(i)})\right\}_{i=1,\ldots,m}$ from $\mu$.

2: **for** $i = 1, \ldots, m$ **do**

3: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{t=0}^{T-1} r(s_t, a_t^{(i)})\right].$$

4: $i^* \leftarrow \operatorname{argmax}_i R_i$.

5: Apply action sequence $(a_0^{(i^*)}, a_1^{(i^*)}, \ldots, a_{T-1}^{(i^*)})$ to the environment.

---

# Simple Idea: Open-Loop Control

- Sample a bunch of action sequences, check which one does the best, and use that sequence for the entire episode!

**Algorithm 2** Open-Loop Control

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: Sample i.i.d. action sequences $\left\{(a_0^{(i)}, a_1^{(i)}, \ldots, a_{T-1}^{(i)})\right\}_{i=1,\ldots,m}$ from $\mu$.

2: **for** $i = 1, \ldots, m$ **do**

3: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{t=0}^{T-1} r(s_t, a_t^{(i)})\right].$$

4: $i^* \leftarrow \operatorname{argmax}_i R_i$.

5: Apply action sequence $(a_0^{(i^*)}, a_1^{(i^*)}, \ldots, a_{T-1}^{(i^*)})$ to the environment.

"Open loop": not adapting response to observed environment state.

# When does open-loop control work?

# When does open-loop control work?

- When would choosing a fixed sequence of actions be near-optimal?

# When does open-loop control work?

- When would choosing a fixed sequence of actions be near-optimal?
    1. Dynamics have minimal stochasticity.

# When does open-loop control work?

- When would choosing a fixed sequence of actions be near-optimal?
  1. Dynamics have minimal stochasticity.
     - If there is a lot of stochasticity, might want to replan actions if realized noise is unfavorable.

# When does open-loop control work?

- When would choosing a fixed sequence of actions be near-optimal?
    1. Dynamics have minimal stochasticity.
        - If there is a lot of stochasticity, might want to replan actions if realized noise is unfavorable.
    2. Using a very accurate model.

# When does open-loop control work?

- When would choosing a fixed sequence of actions be near-optimal?
  1. Dynamics have minimal stochasticity.
     - If there is a lot of stochasticity, might want to replan actions if realized noise is unfavorable.
  2. Using a very accurate model.
     - If action sequence is not well-modeled, may want to replan actions based on new states seen.

# Closing the loop: Model Predictive Control

# Closing the loop: Model Predictive Control

- Easy fix: Replan actions at every step!

# Closing the loop: Model Predictive Control

- Easy fix: Replan actions at every step!

---

**Algorithm 3** Model-Predictive Control

---

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: **for** every timestep $t$ **do**

2: Sample i.i.d. action sequences $\left\{(a_t^{(i)}, a_{t+1}^{(i)}, \dots, a_{t+T-1}^{(i)})\right\}_{i=1,\dots,m}$ from $\mu$.

3: **for** $i = 1, \dots, m$ **do**

4: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{s=0}^{T-1} r(s_{t+s}, a_{t+s}^{(i)}) \;\middle|\; s_t\right].$$

5: Apply the first action from the best sequence to the environment.

---

# Closing the loop: Model Predictive Control

- Easy fix: Replan actions at every step!

**Algorithm 3** Model-Predictive Control

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: **for** every timestep $t$ **do**

2: Sample i.i.d. action sequences $\left\{\left(a_t^{(i)}, a_{t+1}^{(i)}, \ldots, a_{t+T-1}^{(i)}\right)\right\}_{i=1,\ldots,m}$ from $\mu$.

3: **for** $i = 1, \ldots, m$ **do**

4: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{s=0}^{T-1} r\left(s_{t+s}, a_{t+s}^{(i)}\right) \;\middle|\; s_t\right].$$

5: Apply the first action from the best sequence to the environment.

# Is the loop closed?

**Algorithm 3** Model-Predictive Control

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: **for** every timestep $t$ **do**

2: Sample i.i.d. action sequences $\left\{(a_t^{(i)}, a_{t+1}^{(i)}, \ldots, a_{t+T-1}^{(i)})\right\}_{i=1,\ldots,m}$ from $\mu$.

3: **for** $i = 1, \ldots, m$ **do**

4: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{s=0}^{T-1} r(s_{t+s}, a_{t+s}^{(i)}) \,\middle|\, s_t\right].$$

5: Apply the first action from the best sequence to the environment.

# Is the loop closed?

- Is there something that looks weird here?

---

**Algorithm 3** Model-Predictive Control

---

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: **for** every timestep $t$ **do**

2: Sample i.i.d. action sequences $\left\{\left(a_t^{(i)}, a_{t+1}^{(i)}, \ldots, a_{t+T-1}^{(i)}\right)\right\}_{i=1,\ldots,m}$ from $\mu$.

3: **for** $i = 1, \ldots, m$ **do**

4: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{s=0}^{T-1} r(s_{t+s}, a_{t+s}^{(i)}) \;\middle|\; s_t\right].$$

5: Apply the first action from the best sequence to the environment.

---

# Is the loop closed?

- Is there something that looks weird here?

**Algorithm 3** Model-Predictive Control

**Require:** Number of sequences $m$, action sequence proposal distribution $\mu$, dynamics model $\hat{p}$.

1: **for** every timestep $t$ **do**

2: Sample i.i.d. action sequences $\left\{(a_t^{(i)}, a_{t+1}^{(i)}, \ldots, a_{t+T-1}^{(i)})\right\}_{i=1,\ldots,m}$ from $\mu$.

3: **for** $i = 1, \ldots, m$ **do**

4: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{s=0}^{T-1} r(s_{t+s}, a_{t+s}^{(i)}) \;\middle|\; s_t\right].$$

5: Apply the first action from the best sequence to the environment.

# Closed loops only!

# Closed loops only!

- How to specify closed-loop behaviors? Policies!

# Closed loops only!

- How to specify closed-loop behaviors? Policies!

---

**Algorithm 4** Model-Predictive Control with Policies

---

**Require:** Number of policy candidates $m$, policy proposal distribution $\mu$, dynamics model $\hat{p}$.

1: **for** every timestep $t$ **do**
2: Sample $m$ policy candidates $\pi_{\theta_1}, \ldots, \pi_{\theta_m}$ from $\mu$.
3: **for** $i = 1, \ldots, m$ **do**
4: Sample from the model $\hat{p}$ to perform Monte Carlo estimation of

$$R_i = \mathbb{E}\left[\sum_{s=0}^{T-1} r(s_{t+s}, a_{t+s}) \;\middle|\; s_t\right], \quad a_{t+s} \sim \pi_{\theta_i}(s_{t+s}).$$

5: Apply the action from the best policy.

---

# Summary of Control Policies

# Summary of Control Policies

| Control Policy | Improvement |
| --- | --- |
| | |
| | |
| | |

# Summary of Control Policies

| Control Policy | Improvement |
| --- | --- |
| Open-loop control | |
| | |
| | |

# Summary of Control Policies

| **Control Policy** | **Improvement** |
|---|---|
| Open-loop control | |
| (Closed-loop) Model Predictive Control | |
| | |

# Summary of Control Policies

| Control Policy | Improvement |
|---|---|
| Open-loop control | |
| (Closed-loop) Model Predictive Control | Replanning at every step to take into account current state. |
| | |

# Summary of Control Policies

| Control Policy | Improvement |
|---|---|
| Open-loop control | |
| (Closed-loop) Model Predictive Control | Replanning at every step to take into account current state. |
| MPC with (Adaptive) Policies | |

# Summary of Control Policies

| Control Policy | Improvement |
|---|---|
| Open-loop control | |
| (Closed-loop) Model Predictive Control | Replanning at every step to take into account current state. |
| MPC with (Adaptive) Policies | Fixes mismatch between evaluated policies (open-loop) and the overall policy (closed-loop). |

# Part V: Putting Everything Together Suddenly (PETS)

# Probabilistic Ensembles with Trajectory Sampling (PETS)

# Probabilistic Ensembles with Trajectory Sampling (PETS)

**Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models**

# Probabilistic Ensembles with Trajectory Sampling (PETS)

## Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models

**Kurtland Chua** **Roberto Calandra** **Rowan McAllister** **Sergey Levine**
Berkeley Artificial Intelligence Research
University of California, Berkeley
{kchua, roberto.calandra, rmcallister, svlevine}@berkeley.edu

# PETS components

# PETS components

| Component | PETS choice |
| --- | --- |
| | |
| | |
| | |

# PETS components

| Component | PETS choice |
|---|---|
| Model | |
| | |
| | |

# PETS components

| **Component** | **PETS choice** |
| --- | --- |
| Model | Ensemble of neural networks<br>Heteroskedastic noise model<br>MLE |
| | |
| | |

# PETS components

| **Component** | **PETS choice** |
|---|---|
| Model | Ensemble of neural networks<br>Heteroskedastic noise model<br>MLE |
| Controller | |
| | |

# PETS components

| Component | PETS choice |
|---|---|
| Model | Ensemble of neural networks<br>Heteroskedastic noise model<br>MLE |
| Controller | Basic MPC |
| | |

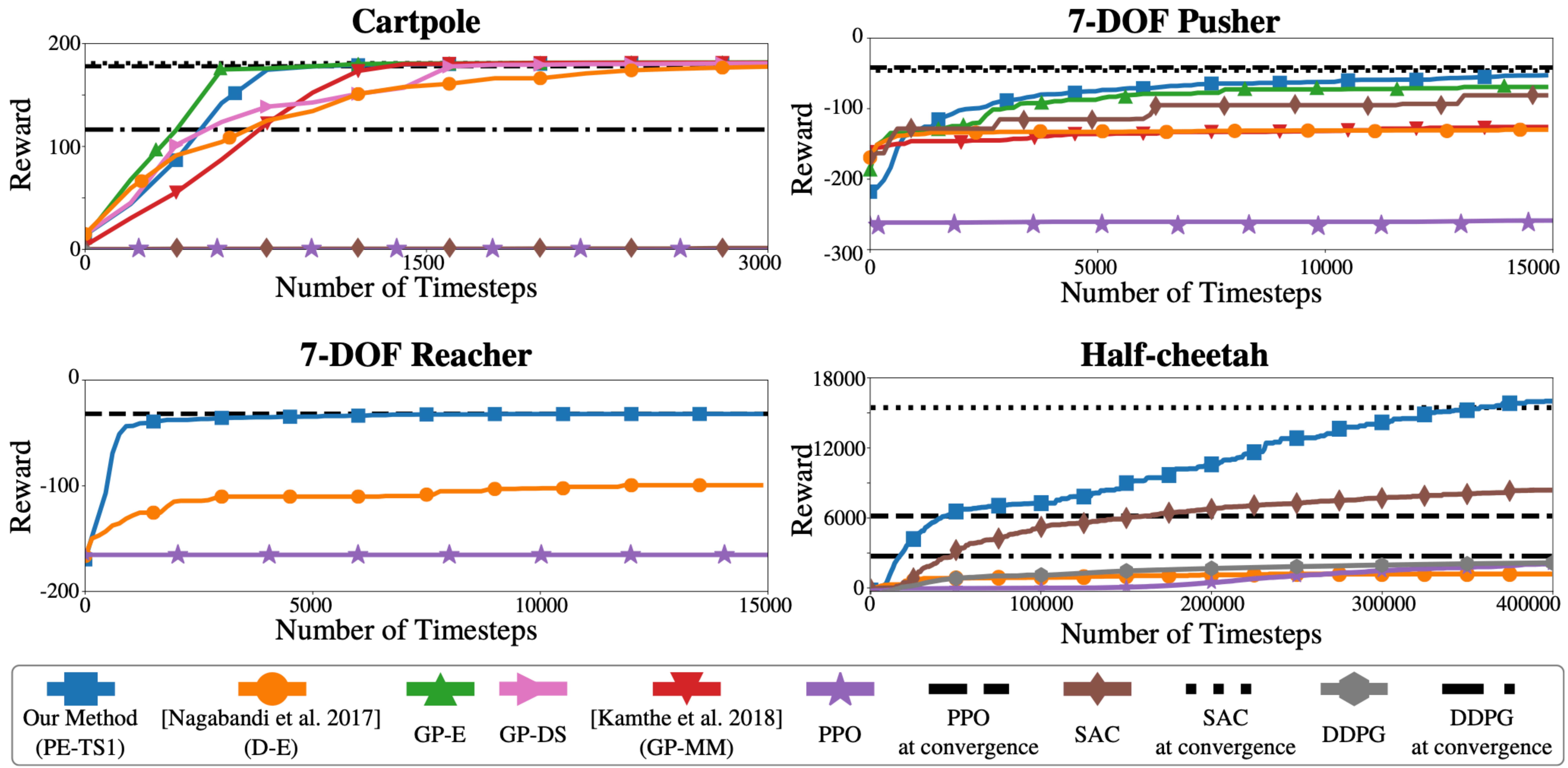
# PETS components

| Component | PETS choice |
| --- | --- |
| Model | Ensemble of neural networks<br>Heteroskedastic noise model<br>MLE |
| Controller | Basic MPC |
| Extras/Useful Tricks | |

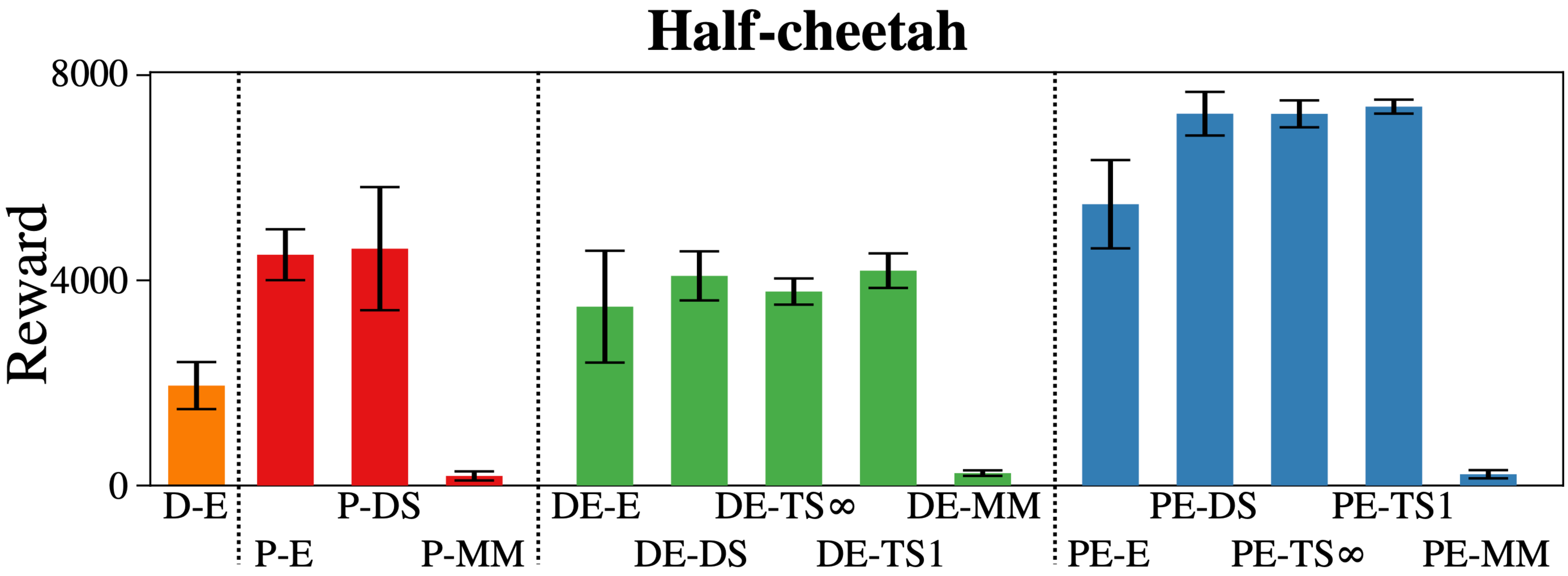
# PETS components

| Component | PETS choice |
|---|---|
| Model | Ensemble of neural networks<br>Heteroskedastic noise model<br>MLE |
| Controller | Basic MPC |
| Extras/Useful Tricks | Cross-entropy method-based optimization;<br>Model predicts state change;<br>Warm-start MPC |

# Some experimental results



# Some experimental results



# PETS on HalfCheetah

# PETS on HalfCheetah

You may not like it, but this is what peak performance looks like