人工智能实践
Artificial Intelligence Practice

DCS3015 Autumn 2022

Chao Yu (余超)

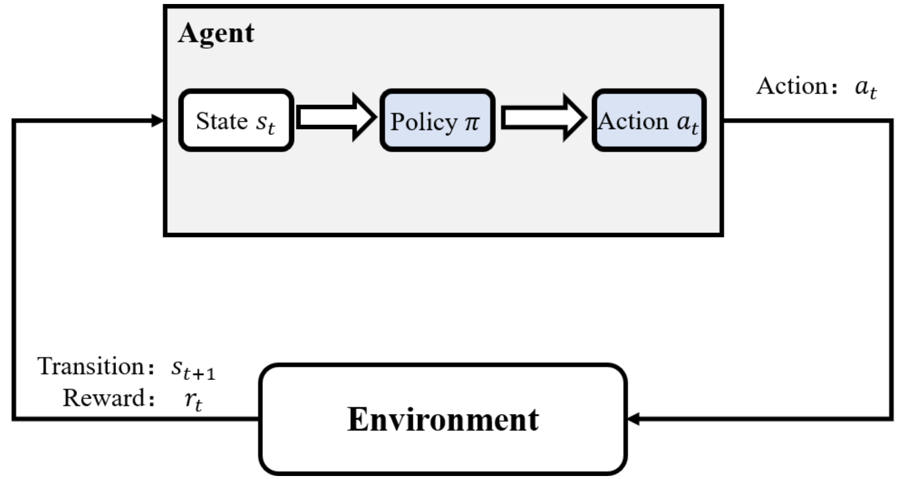
School of Computer Science and Engineering
Sun Yat-Sen University

Lecture 8： Model-based RL

15th December 2022

Quick Recap

- States $S \in \mathbb{R}^s$
- Actions $A \in \mathbb{R}^a$
- Reward Function $R: S \times A \rightarrow \mathbb{R}$
- Transition Function $T: S \times A \rightarrow S$
- Discounted Factor $\gamma \in (0,1)$
- Policy $\pi: S \rightarrow A$

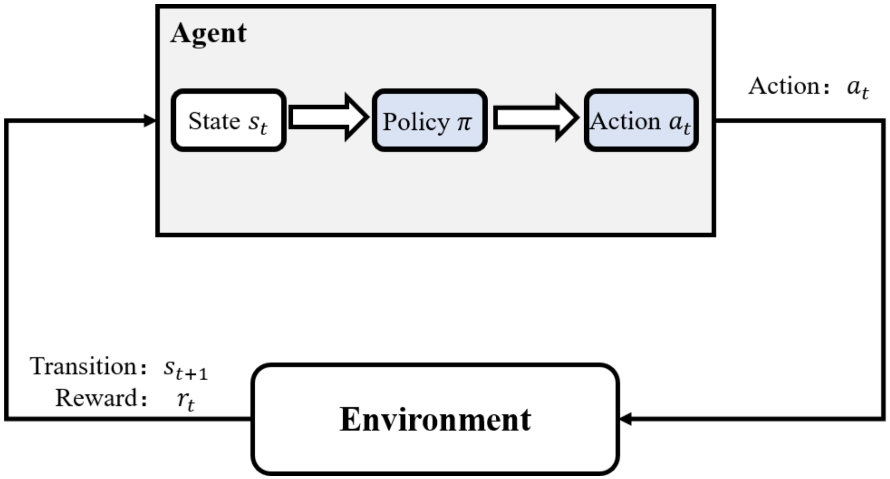


$$\max_{\pi} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

Subject to $a_t = \pi(s_t)$, $s_{t+1} = T(s_t, a_t)$

Quick Recap

- States $S \in \mathbb{R}^s$
- Actions $A \in \mathbb{R}^a$
- Reward Function $R: S \times A \rightarrow \mathbb{R}$
- Transition Function $T: S \times A \rightarrow S$
- Discounted Factor $\gamma \in (0,1)$
- Policy $\pi: S \rightarrow A$



$$\max_{\pi} \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

Subject to $a_t = \pi(s_t)$, $s_{t+1} = T(s_t, a_t)$

Model-free RL vs. Model-based RL

Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

Model-free: learn policy directly from data

$$\mathcal{D} \rightarrow \pi$$

e.g. Q-learning, policy gradient

Model-based: learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

Model-free RL vs. Model-based RL

Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

Model-free: learn policy directly from data

$$\mathcal{D} \rightarrow \pi$$ *e.g. Q-learning, policy gradient*

Model-based: learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

Why Model-based ?

	Model-free	Model-based
Asymptotic Performance	+	- / +
Computation	+	-
Sample Efficiency	-	+
Exploration	-	+

Model-free RL vs. Model-based RL

Collect data

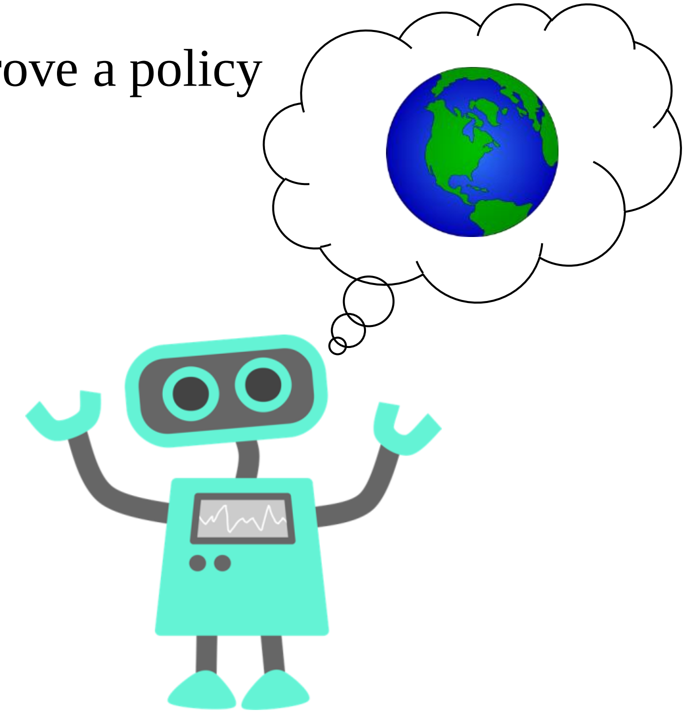
$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

Model-free: learn policy directly from data

$$\mathcal{D} \rightarrow \pi$$ *e.g. Q-learning, policy gradient*

Model-based: learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$



□ What is a Model ?

What is a Model ?

Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

Model-free: learn policy directly from data

$$\mathcal{D} \to \pi$$ *e.g. Q-learning, policy gradient*

Model-based: learn model, then use it to learn or improve a policy

$$\mathcal{D} \to f \to \pi$$

☐ What is a Model ?

Definition: a model is a representation that **explicitly** encodes knowledge about the structure of the environment and task. state_next, reward, terminate, infos = env.step(action)

A transition/dynamics model: $s_{t+1} = f_s(s_t, a_t)$

A rewards model: $r_{t+1} = f_r(s_t, a_t)$

A inverse transition/dynamics model: $a_t = f_s^{-1}(s_t, s_{t+1})$

What is a Model ?

Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

Model-free: learn policy directly from data

$\mathcal{D} \rightarrow \pi$ *e.g. Q-learning, policy gradient*

Model-based: learn model, then use it to learn or improve a policy

$$\mathcal{D} \rightarrow f \rightarrow \pi$$

☐ What is a Model ?

Definition: a model is a representation that **explicitly** encodes knowledge about the structure of the environment and task.

state_next, **reward**, terminate, infos = env.step(action)

A transition/dynamics model: $s_{t+1} = f_s(s_t, a_t)$

A rewards model: $r_{t+1} = f_r(s_t, a_t)$

Typically what is meant by the model in model-based RL

A inverse transition/dynamics model: $a_t = f_s^{-1}(s_t, s_{t+1})$

What is a Model ?

Collect data

$$\mathcal{D} = \{s_t, a_t, r_{t+1}, s_{t+1}\}_{t=0}^{T}$$

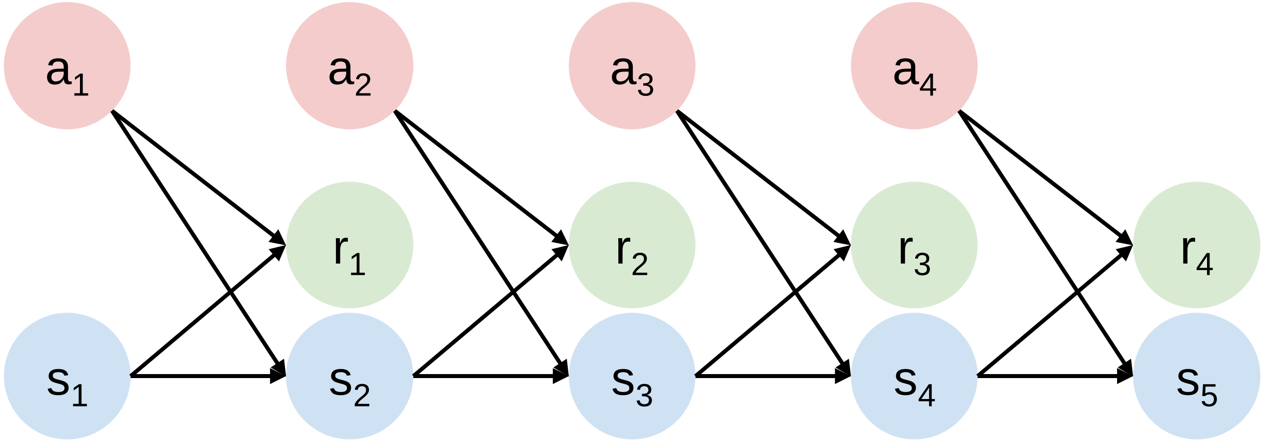
Model-free: learn policy directly from data

$$\mathcal{D} \to \pi$$ *e.g. Q-learning, policy gradient*

Model-based: learn model, then use it to learn or improve a policy

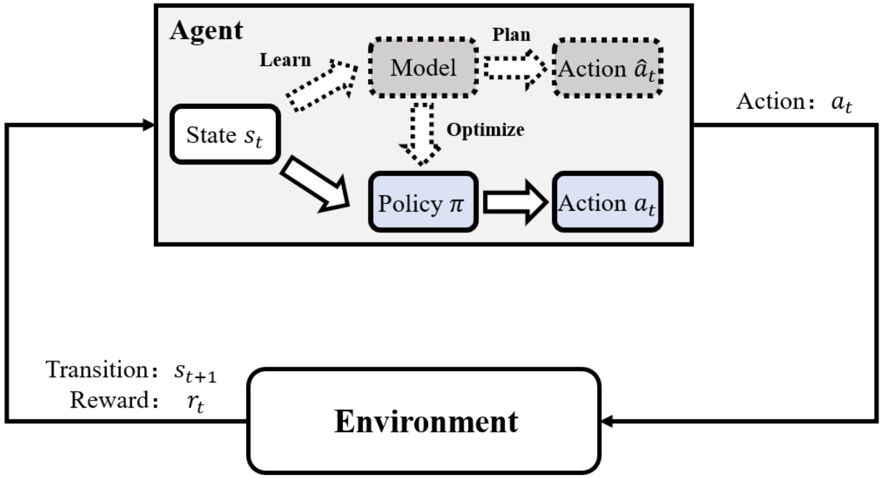
$$\mathcal{D} \to f \to \pi$$

❑ What is a Model ?



Model-based RL

Model-based RL Process:



Run base policy $\pi_0(a_t|s_t)$ to collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

For _ do

Learn transition/dynamics model $f(s, a)$

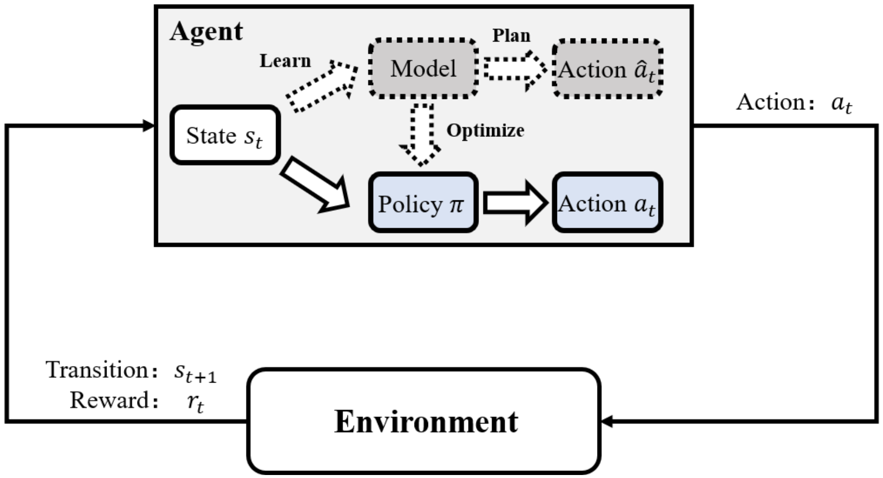
Use model $f(s, a)$ to assist agent to update policy $\pi(a_t|s_t)$

Execute $\pi(a_t|s_t)$ and add the resulting data $\{(s_t, a_t, s_{t+1})_j\}$ to $\mathcal{D}$

End For

Model-based RL

Model-based RL Process:



Run base policy $\pi_0(a_t|s_t)$ to collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

For _ do

Learn transition/dynamics model $f(s, a)$ **Model Building**

Use model $f(s, a)$ to assist agent to update policy $\pi(a_t|s_t)$ **Model Utilizing**

Execute $\pi(a_t|s_t)$ and add the resulting data $\{(s_t, a_t, s_{t+1})_j\}$ to $\mathcal{D}$

End For

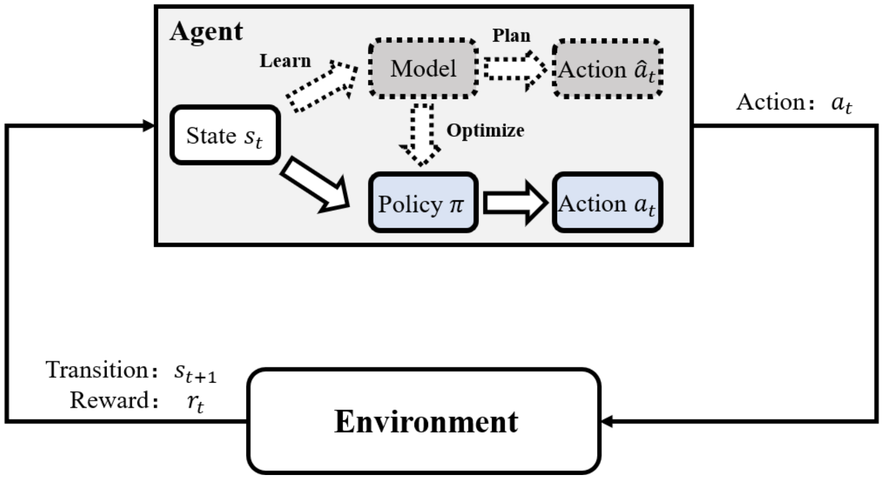
Model-based RL

◻ Model Building

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

$\Downarrow$

Deterministic Model or **Probabilistic Model**

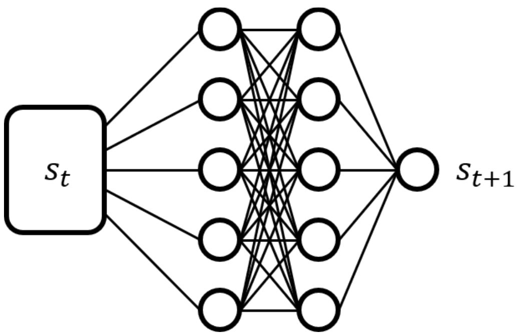


Model-based RL

Model Building

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

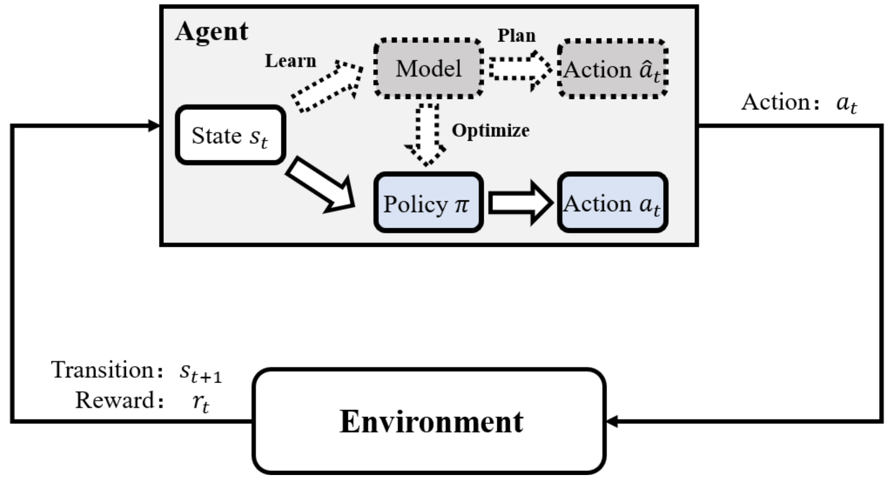
Deterministic Model or Probabilistic Model



$$\hat{s}_{t+1} \sim f(s_t, a_t)$$

Incremental Predict:

$$\hat{s}_{t+1} \sim s_t + f(s_t, a_t)$$



Learn transition model $f(s, a)$ to minimize **Mean Squared Error (MSE):**

$$\mathcal{L}_{MSE} = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left[f\left(s_t, a_t\right) - s_{t+1} \right]^2 .$$

Model-based RL

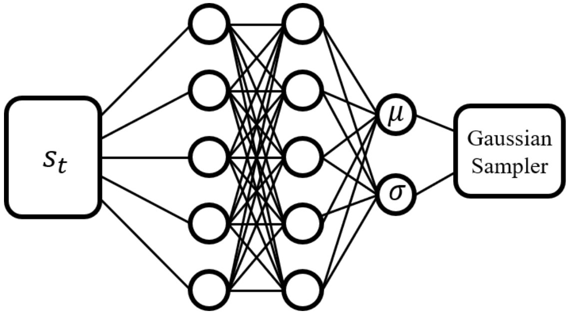
Model Building

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

$\Downarrow$

Deterministic Model or **Probabilistic Model**

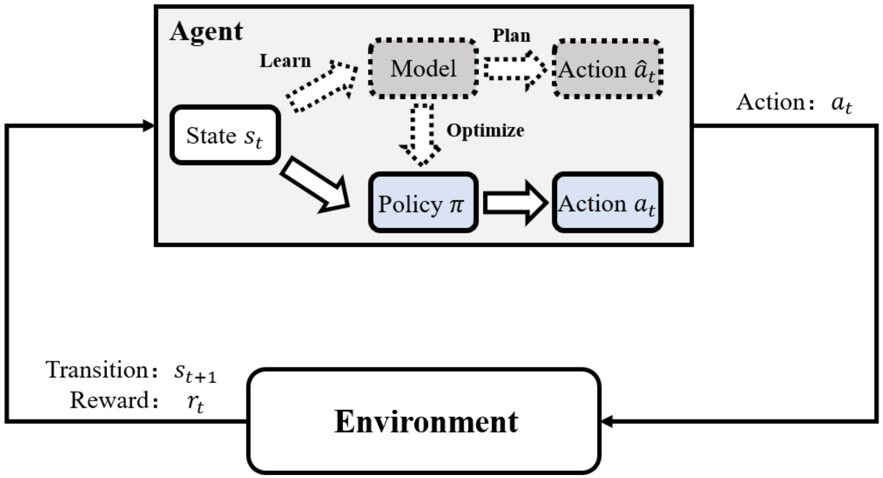
$\Downarrow$



$$\hat{s}_{t+1} \sim \mathcal{N}\left(\mu(s_t, a_t), \sigma(s_t, a_t)\right)$$

Learn transition model $f(s, a)$ to optimize **Negative Log Likelihood (NLL)**

$$\mathcal{L}_{NLL} = \mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{D}} \left(\frac{\left[\mu(s_t, a_t) - s_{t+1}\right]^2}{\sigma(s_t, a_t)} + \log \sigma(s_t, a_t) \right)$$



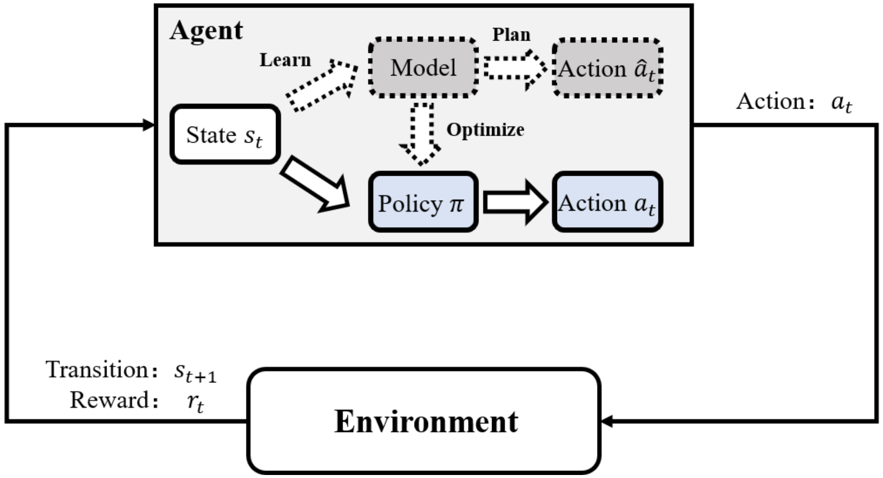
Model-based RL

Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

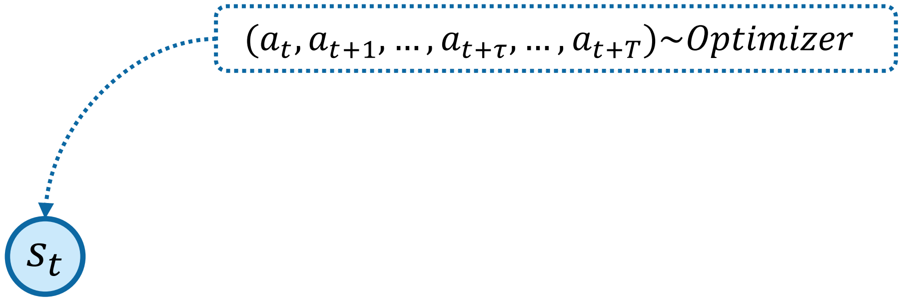
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



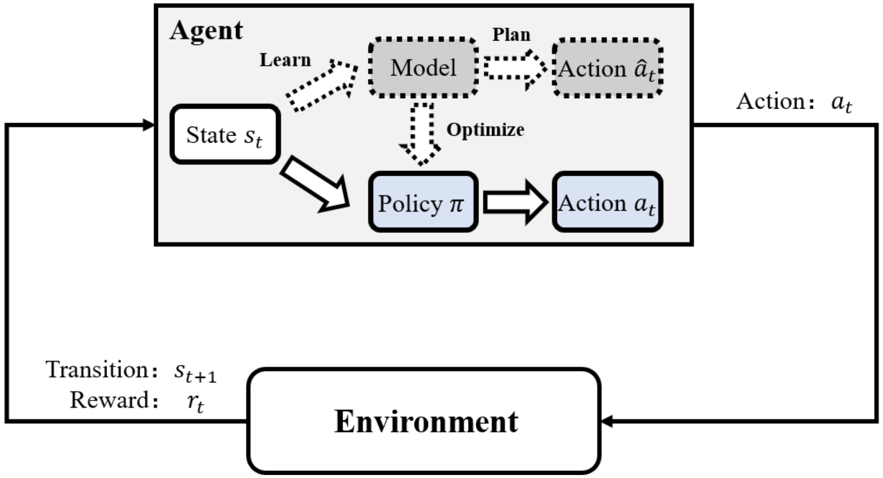
Model-based RL

Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

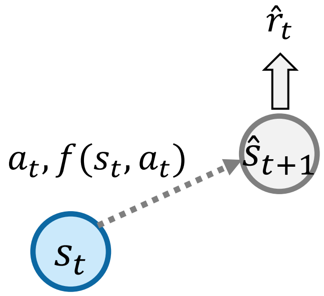
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



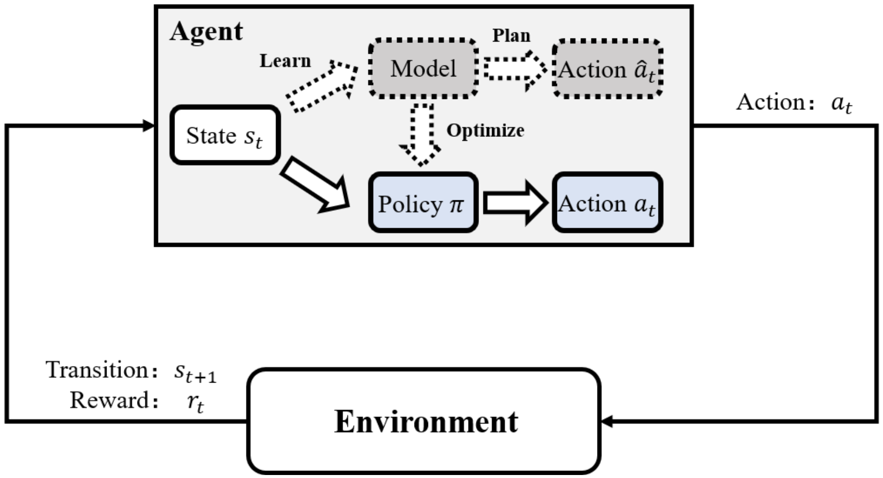
Model-based RL

Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

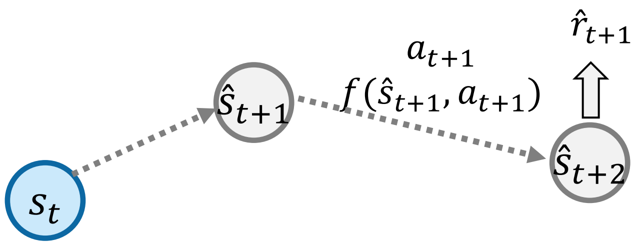
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



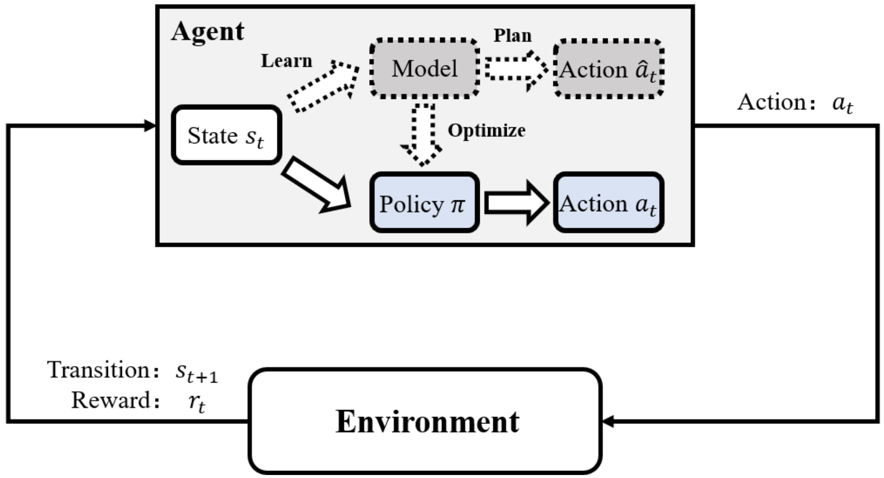
Model-based RL

◻ Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

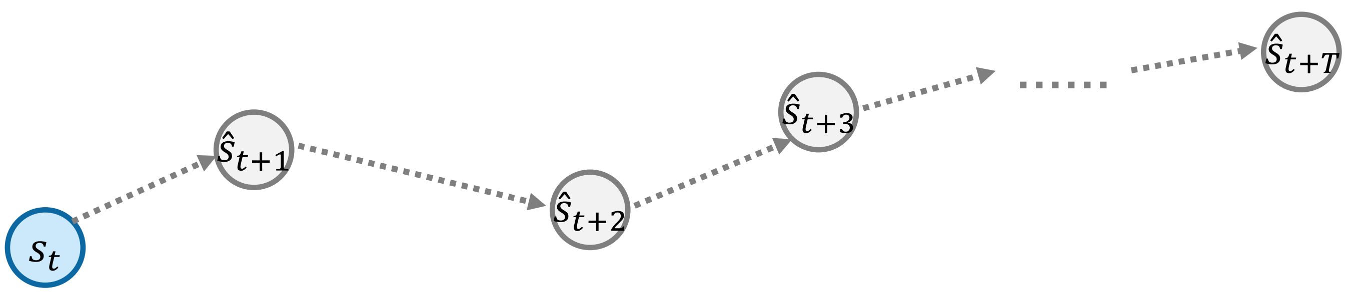
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



◻ Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



$$\Longrightarrow \sum_{\tau=t}^{t+T-1} \hat{r}(s_\tau, a_\tau)$$

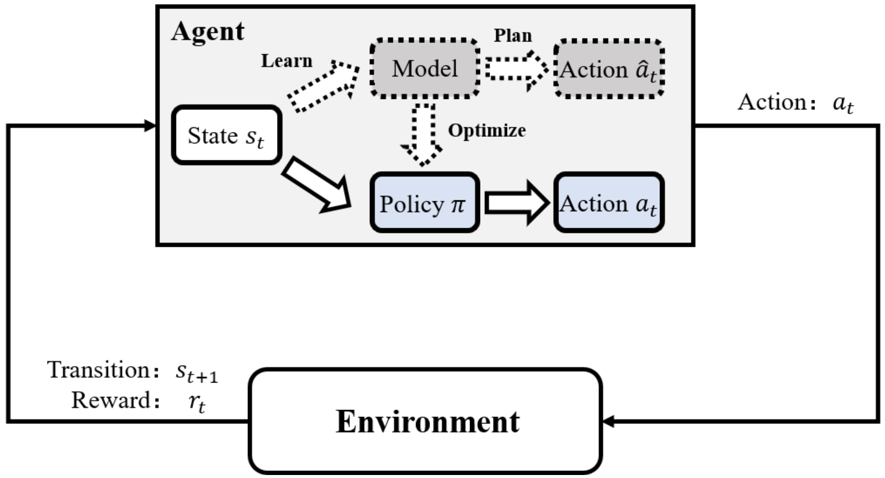
Model-based RL

Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

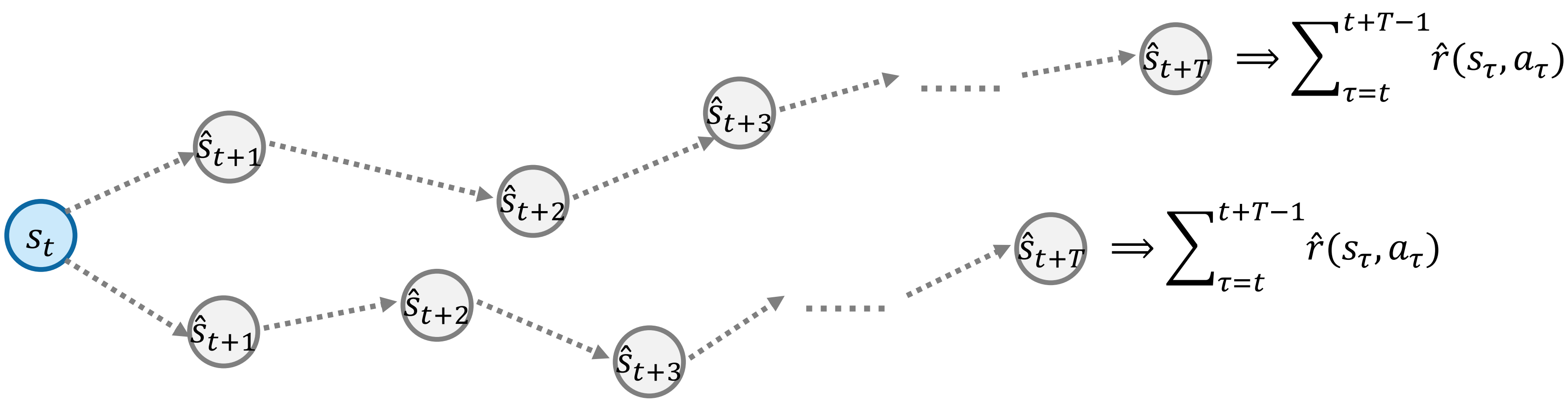
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



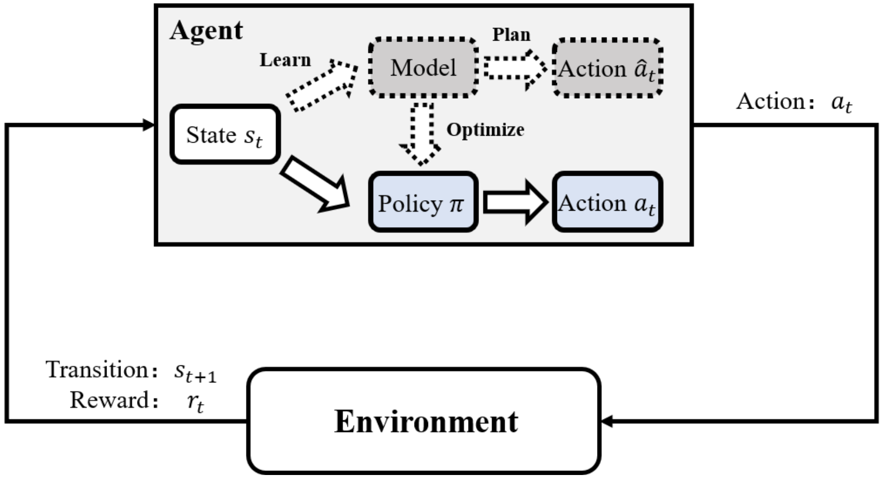
Model-based RL

Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

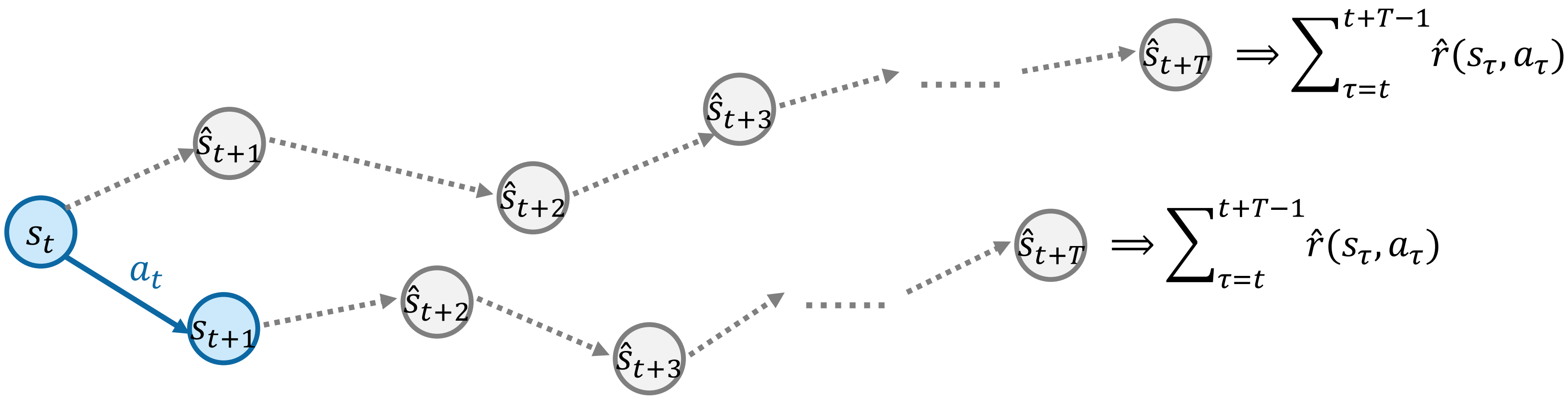
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



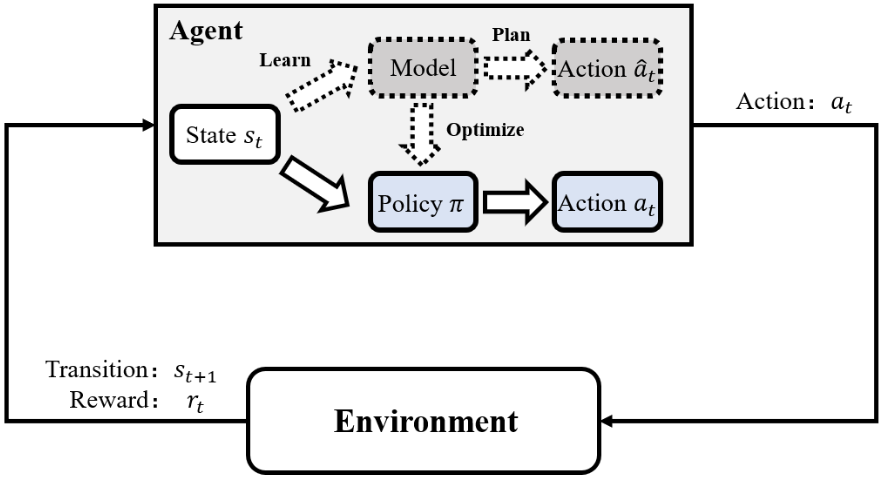
Model-based RL

Model Building:

Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

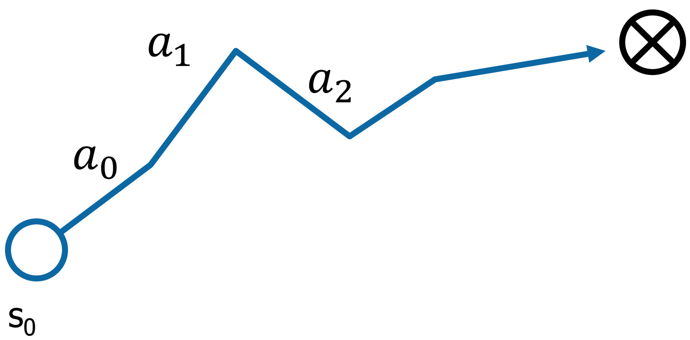
Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Decision-Time Method (Plan):

Model Predictive Control (MPC) without Policy Function π



Trajectory Optimization with Action Sequence:

$$s_t \rightarrow a_t \rightarrow s_{t+1} \rightarrow a_{t+1} \rightarrow \cdots$$

$$J(a_0, \ldots, a_H) = \sum_{t=0}^{H} \gamma^t r_t$$

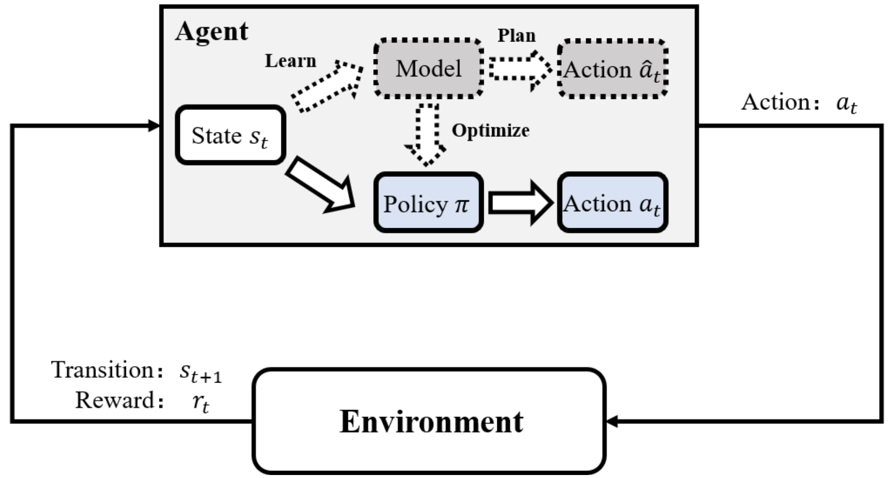
Model-based RL

Model Building:

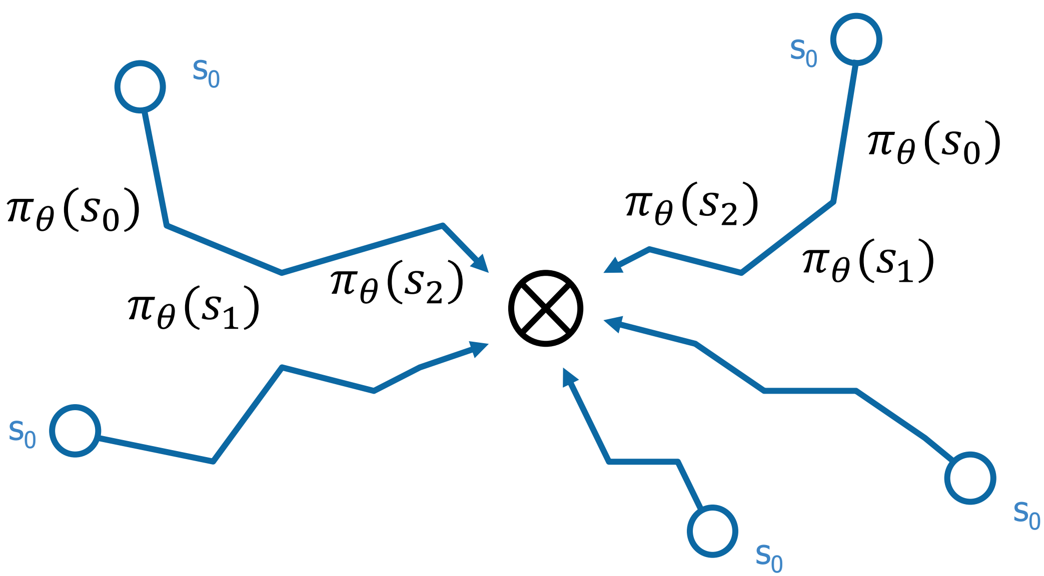
Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

Learn transition model $f(s, a)$ to minimize

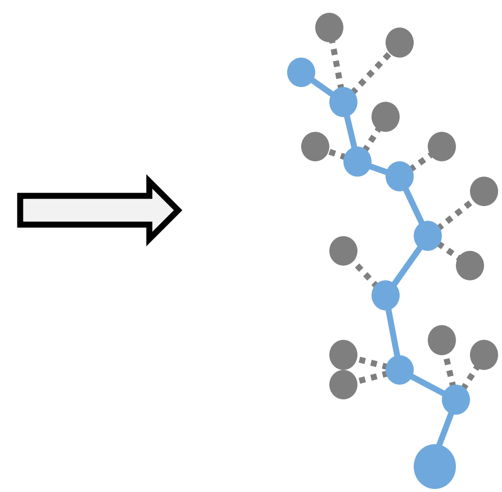
$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Background Method (Optimize):



Mix real and model-generated experience and apply additional policy updates.



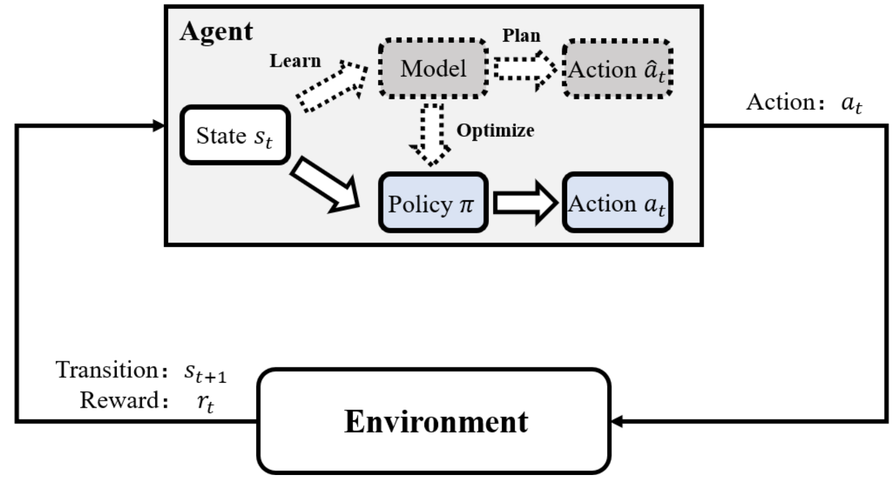
Model-based RL

Model Building:

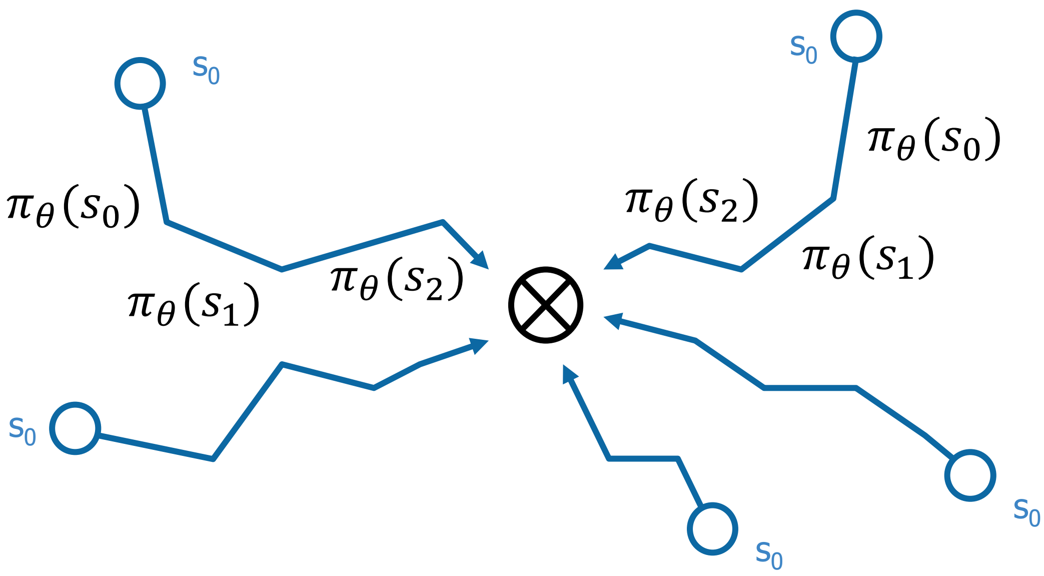
Collect data $\mathcal{D} = \{(s_t, a_t, s_{t+1})_i\}$

Learn transition model $f(s, a)$ to minimize

$$\sum_i \|f(s_t, a_t) - s_{t+1}\|^2$$



Background Method (Optimize):



Policy Optimization with parameter θ :

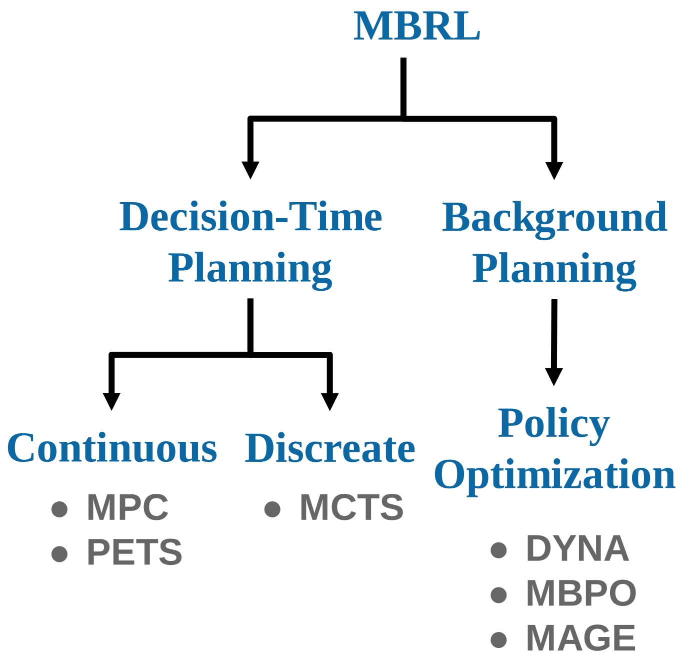
$$s_t \rightarrow \pi_\theta(s_t) \rightarrow s_{t+1} \rightarrow \pi_\theta(s_{t+1}) \rightarrow \cdots$$

$$J(\theta) = \mathbb{E}_{s_0}\left[\sum_{t=0}^{H} \gamma^t r_t\right], \quad a_t = \pi_\theta(s_t)$$

Model-based RL

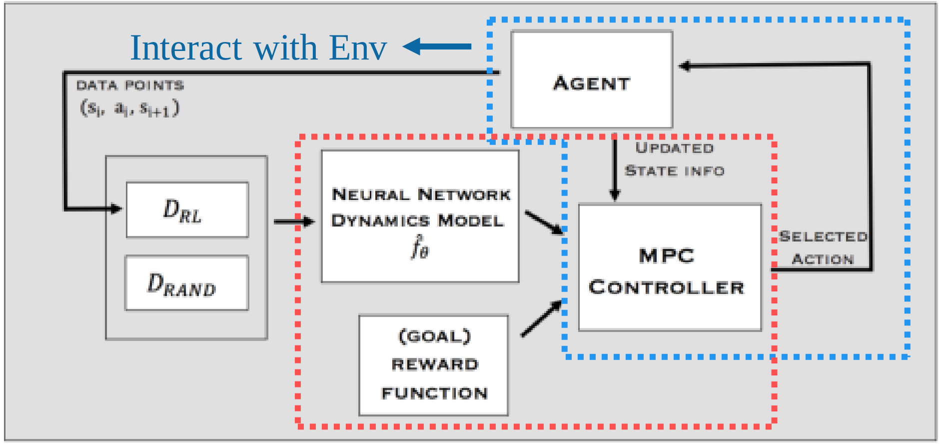
Background vs. Decision-Time

	Background	Decision-Time
Act without Learning	-	+
Unfamiliar situations	-	+
Computation	+	-
Predictability Coherence	+	-
Discrete and Continuous	+	-



MPC with Neural Network Model

- $\mathcal{D}_{RL} = \{(s_t, a_t, s_{t+1})_i\}$ to train neural network dynamic model.
- $\mathcal{D}_{RAND} = \{(a_1, a_2, \dots, a_T)\}$ to store the sampled actions sequence.
- (Goal) Reward Function



1. Planning based on Model
2. Agent action selection

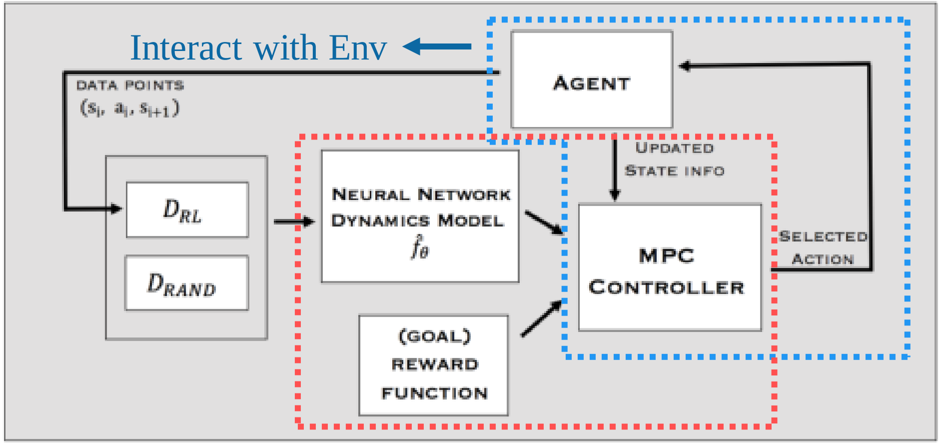
Nagabandi, Anusha, et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning.", ICRA. 2018.

Decision-Time

MPC with Neural Network Model

Algorithm 1 Model-based Reinforcement Learning

gather dataset $\mathcal{D}_{\text{RAND}}$ of random trajectories
initialize empty dataset $\mathcal{D}_{\text{RL}}$, and randomly initialize $\hat{f}_\theta$
for iter=1 **to** max_iter **do**
  train $\hat{f}_\theta(\mathbf{s}, \mathbf{a})$ by performing gradient descent on Eqn. 2, using $\mathcal{D}_{\text{RAND}}$ and $\mathcal{D}_{\text{RL}}$
  for $t = 1$ **to** T **do**
    get agent's current state $\mathbf{s}_t$
    use $\hat{f}_\theta$ to estimate optimal action sequence $\mathbf{A}_t^{(H)}$ (Eqn. 4)
    execute first action $\mathbf{a}_t$ from selected action sequence $\mathbf{A}_t^{(H)}$
    add $(\mathbf{s}_t, \mathbf{a}_t)$ to $\mathcal{D}_{\text{RL}}$
  end for
end for



1. Planning based on Model
2. Agent action selection

$$\hat{\mathbf{s}}_{t+1} = \mathbf{s}_t + \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)$$

$$\mathcal{E}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}} \frac{1}{2} \|(\mathbf{s}_{t+1} - \mathbf{s}_t) - \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)\|^2 \quad (1)$$

$$\mathbf{A}_t^{(H)} = \arg\max_{\mathbf{A}_t^{(H)}} \sum_{t'=t}^{t+H-1} r(\hat{\mathbf{s}}_{t'}, \mathbf{a}_{t'}) \quad :$$
$$\hat{\mathbf{s}}_t = \mathbf{s}_t, \hat{\mathbf{s}}_{t'+1} = \hat{\mathbf{s}}_{t'} + \hat{f}_\theta(\hat{\mathbf{s}}_{t'}, \mathbf{a}_{t'}). \qquad (3)$$

Random Shooting (RS):

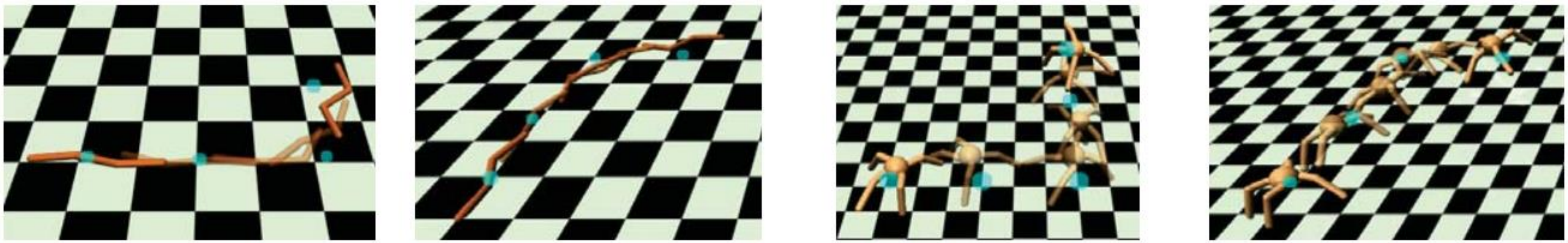
For $t = 1$ to T do:

$$random\ (a_t) \in A \ \rightarrow \mathcal{D}_{RAND}$$

Nagabandi, Anusha, et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning.", ICRA. 2018.

Decision-Time

MPC with Neural Network Model



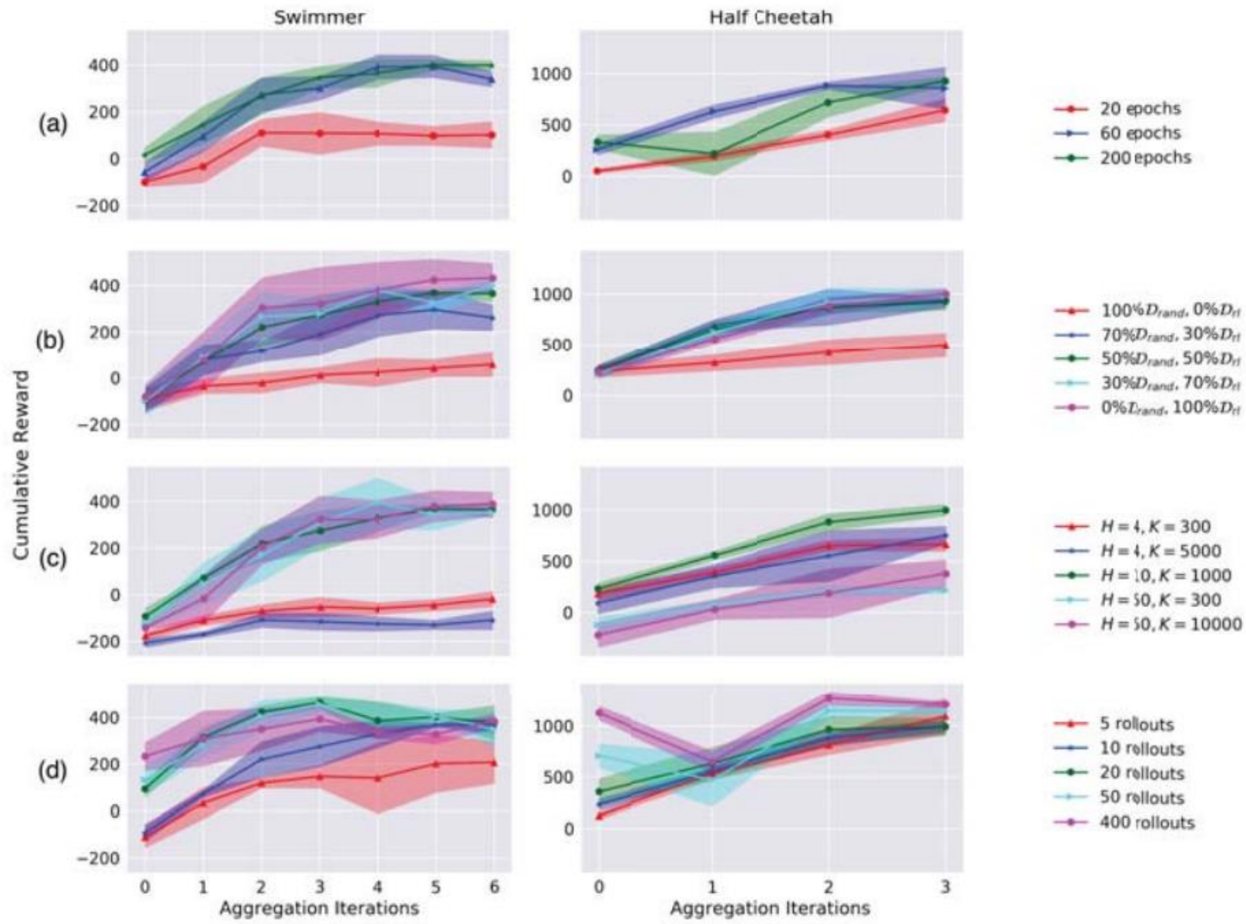
(a) Swimmer left turn (b) Swimmer right turn (c) Ant left turn (d) Ant right turn

(a) Swimmer (b) Cheetah (c) Ant (d) Hopper

Nagabandi, Anusha, et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning.", ICRA. 2018.

Decision-Time

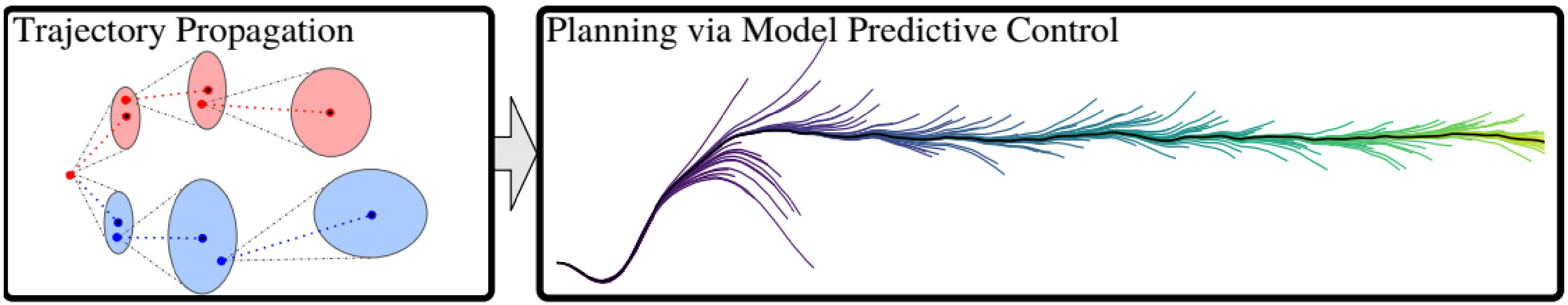
MPC with Neural Network Model



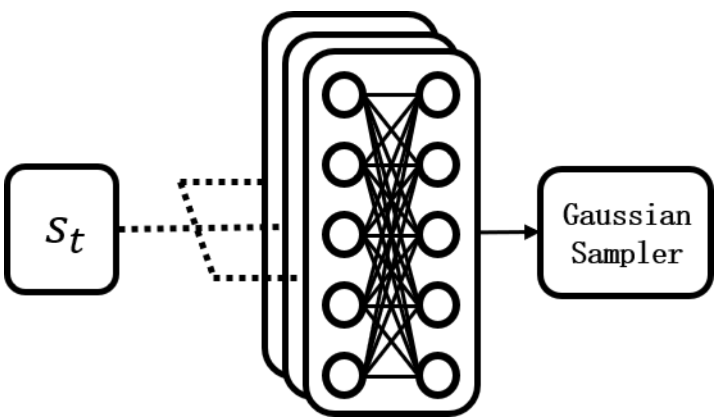
Nagabandi, Anusha, et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning.", ICRA. 2018.

Decision-Time

Probabilistic Ensemble with Trajectory Sampling (PETS)



- Probabilistic Model Networks to capture **aleatoric uncertainty**.
- Model Ensembles to capture **epistemic uncertainty**.



Model Ensemble Output:

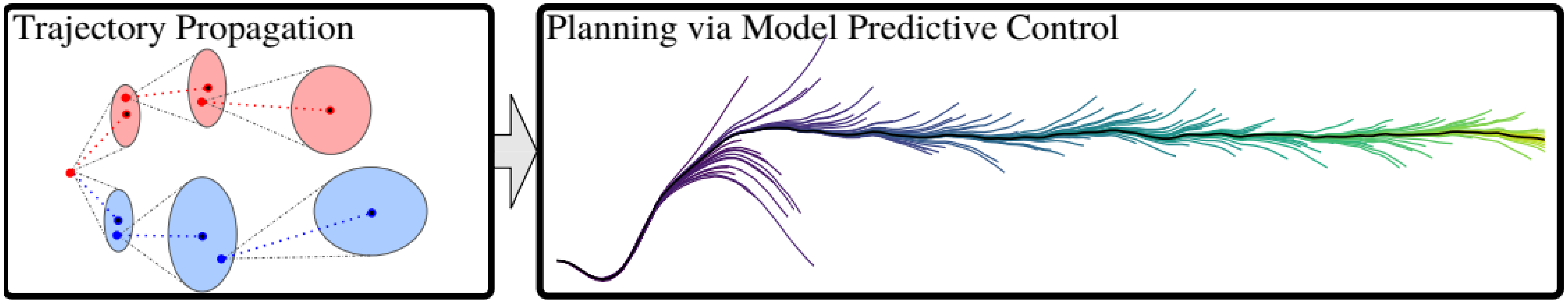
$$\hat{s}_{t+1} = random\left(f_1(s_t, a_t), f_2(s_t, a_t), \ldots, f_M(s_t, a_t)\right)$$

$$\hat{s}_{t+1} = \frac{1}{M}\sum_{j=1}^{M} f_j(s_t, a_t).$$

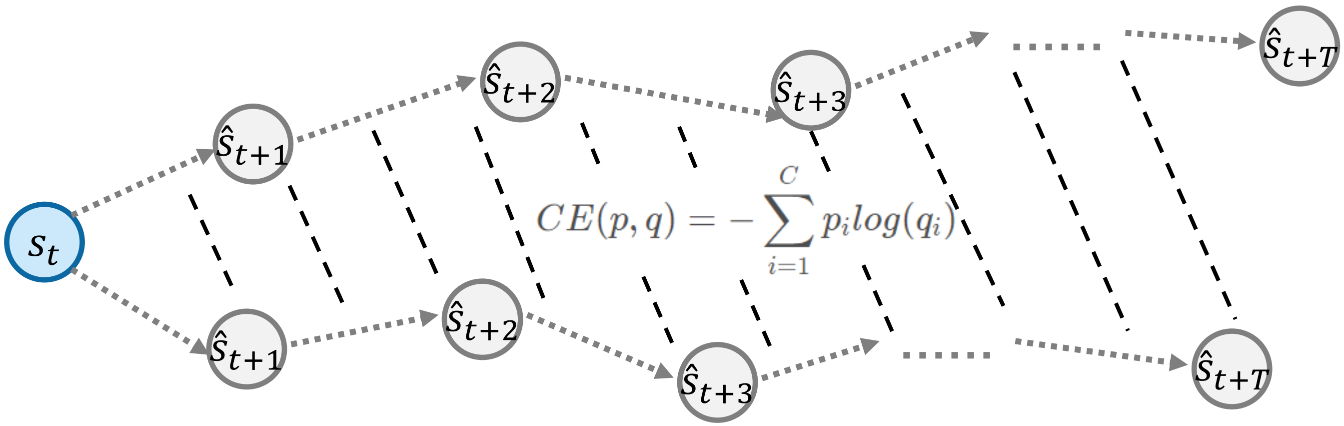
Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." , NIPS, 2018.

Decision-Time

Probabilistic Ensemble with Trajectory Sampling (PETS)



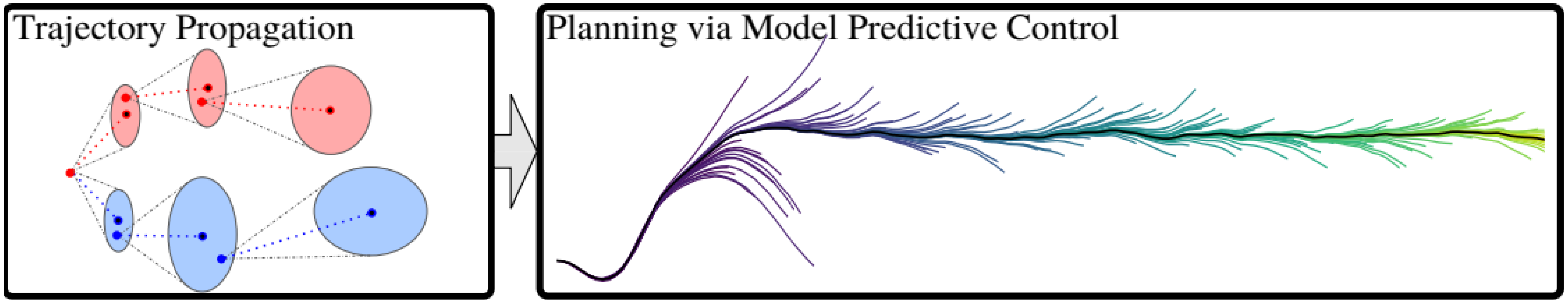
Cross Entropy Method (CEM) instead of Random Shooting (RS)



Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." , NIPS, 2018.

Decision-Time

Probabilistic Ensemble with Trajectory Sampling (PETS)



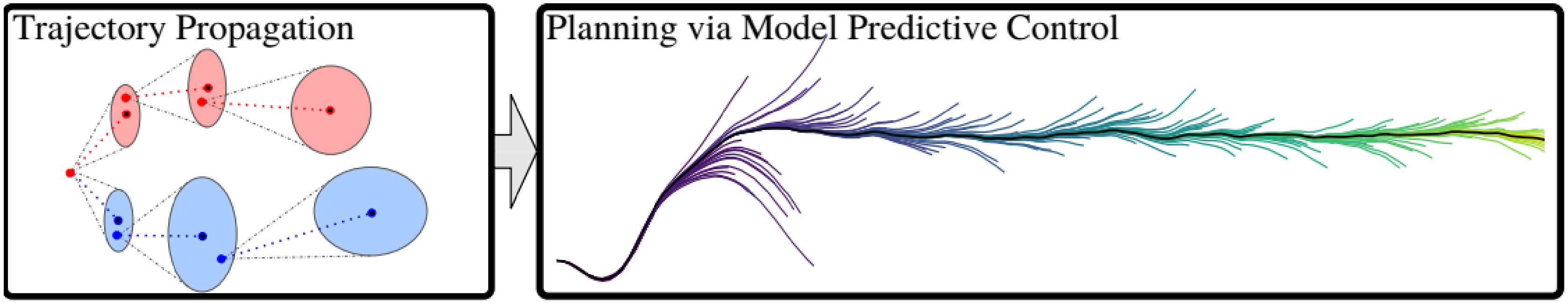
Algorithm 1 Our model-based MPC algorithm '*PETS*':

Initialize data $\mathbb{D}$ with a random controller for one trial.
for Trial $k = 1$ to K **do**
  Train a *PE* dynamics model $\widetilde{f}$ given $\mathbb{D}$.
  for Time $t = 0$ to TaskHorizon **do**
    for Actions sampled $\boldsymbol{a}_{t:t+T} \sim \text{CEM}(\cdot)$, 1 to NSamples **do**
      Propagate state particles $\boldsymbol{s}_\tau^p$ using *TS* and $\widetilde{f}|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
      Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_\tau^p, \boldsymbol{a}_\tau)$
      Update CEM$(\cdot)$ distribution.
    Execute first action $\boldsymbol{a}_t^*$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^*$.
    Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_t, \boldsymbol{a}_t^*, \boldsymbol{s}_{t+1}\}$.

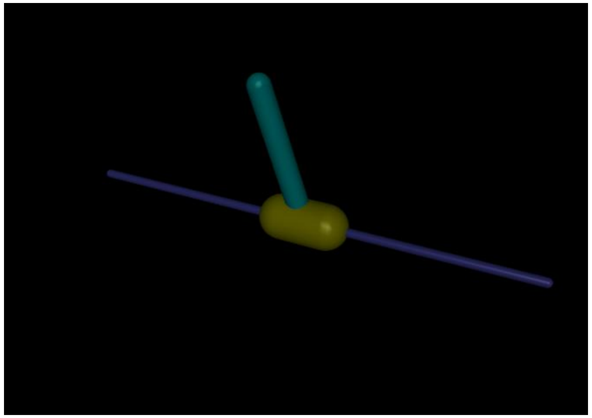
Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." , NIPS, 2018.

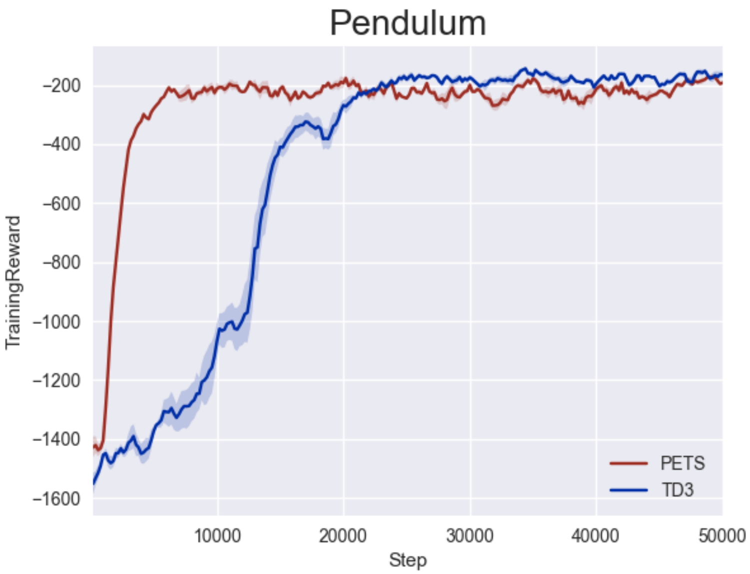
Decision-Time

Probabilistic Ensemble with Trajectory Sampling (PETS)



Experiments



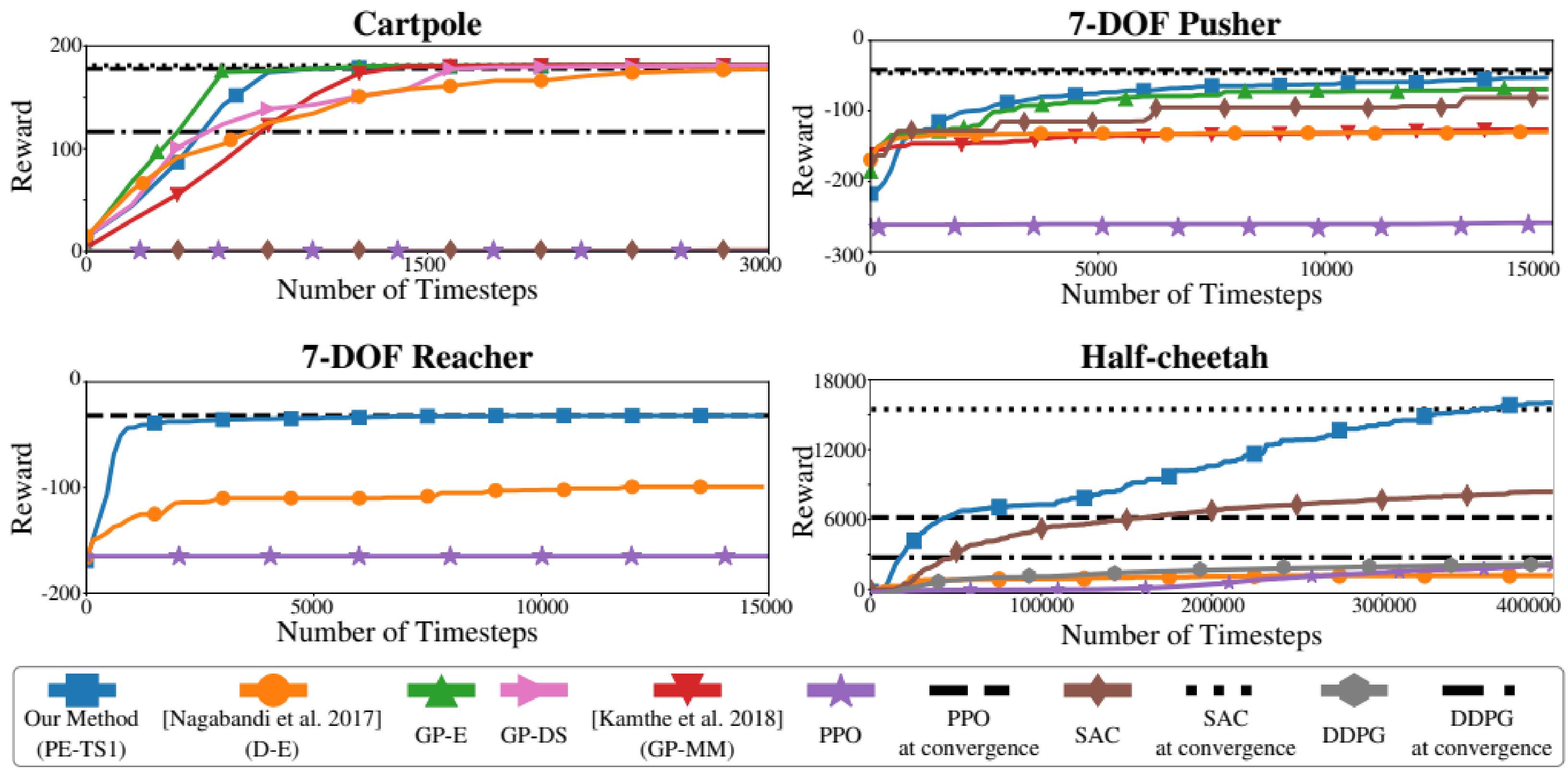


https://github.com/ShAw7ock/PETS-MuJoCo

Decision-Time

Probabilistic Ensemble with Trajectory Sampling (PETS)

Experiments

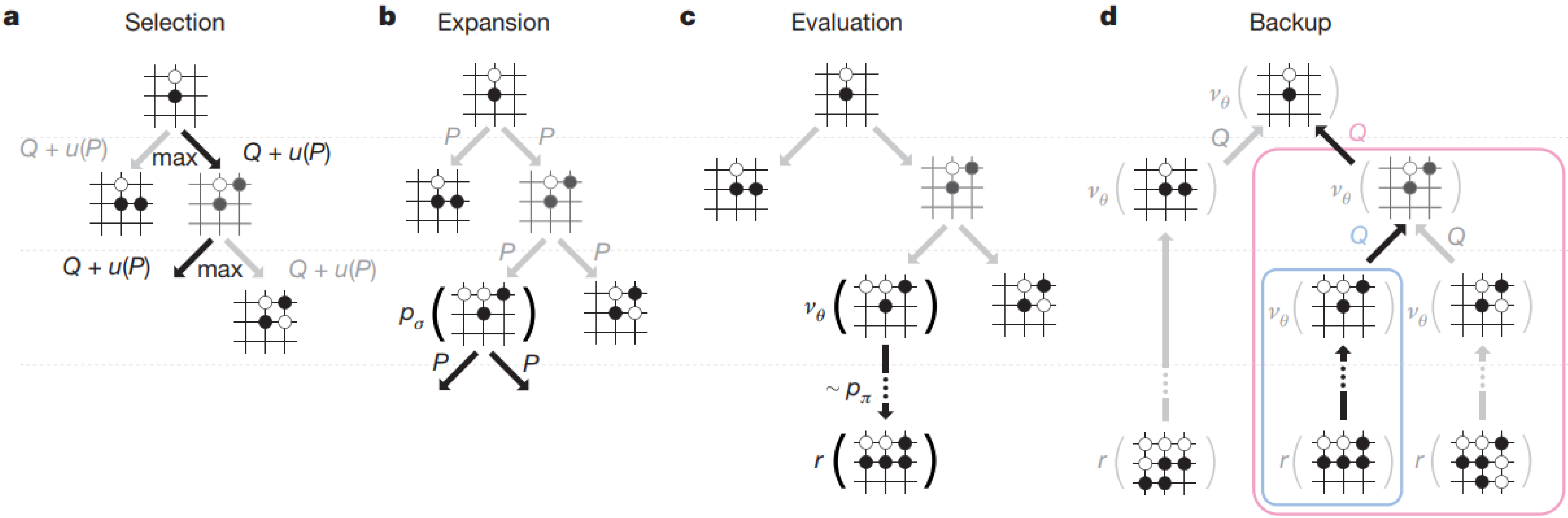


https://github.com/ShAw7ock/PETS-MuJoCo

Decision-Time

MuZero: Mastering Atari, Go and Chess

Monte-Carlo Tree Search (MCTS) with Dynamics Model

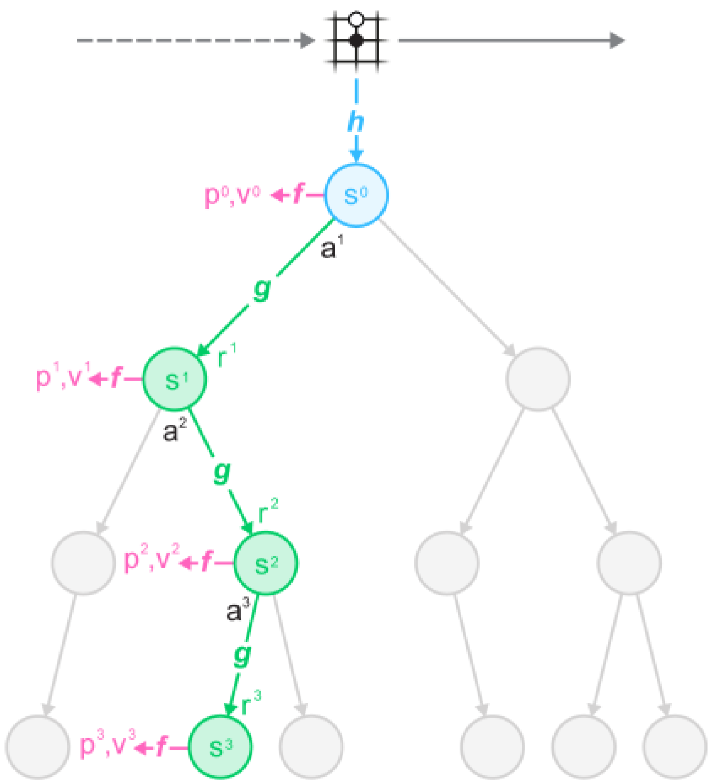


Schrittwieser, Julian, et al. "Mastering atari, go, chess and shogi by planning with a learned model." Nature. 2020.

Decision-Time

MuZero: Mastering Atari, Go and Chess

Monte-Carlo Tree Search (MCTS) with Dynamics Model



- Representation Function: $s^0 = h_\theta(o_1, \ldots, o_n)$ to encode past observations.
- Dynamic Function: $r^k, s^k = g_\theta(s^{k-1}, a^k)$
- Prediction Function: $\boldsymbol{p}^k, v^k = f_\theta(s^k)$

MCTS Backup:

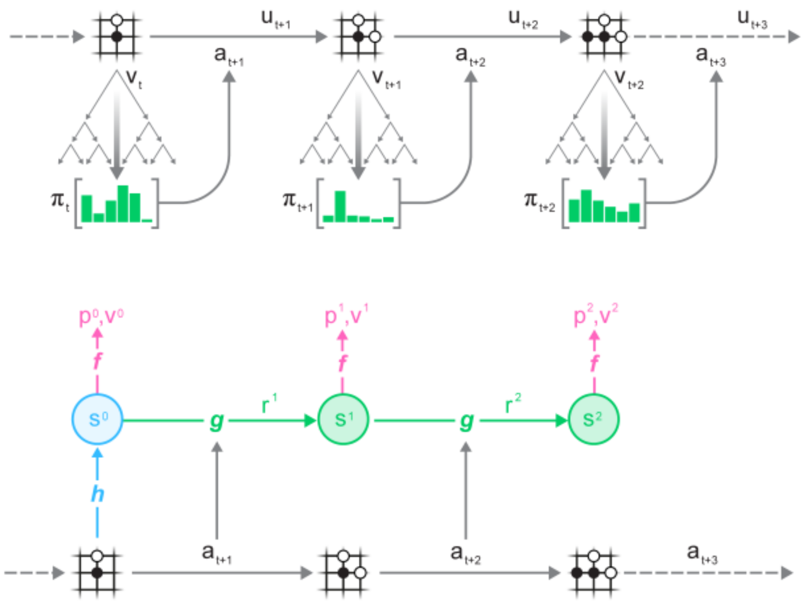
$$Q_{k+1}(s,a) = \frac{Q_k(s,a)\cdot N_k(s,a) + R}{N_k(s,a)+1}$$

$$N_{k+1}(s,a) = N_k(s,a) + 1$$

Schrittwieser, Julian, et al. "Mastering atari, go, chess and shogi by planning with a learned model." Nature. 2020.

MuZero: Mastering Atari, Go and Chess

Monte-Carlo Tree Search (MCTS) with Dynamics Model



- Representation Function: $s^0 = h_\theta(o_1, \dots, o_n)$ to encode past observations.
- Dynamic Function: $r^k, s^k = g_\theta(s^{k-1}, a^k)$
- Prediction Function: $\boldsymbol{p}^k, v^k = f_\theta(s^k)$

MCTS Backup:

$$Q_{k+1}(s,a) = \frac{Q_k(s,a) \cdot N_k(s,a) + R}{N_k(s,a) + 1}$$

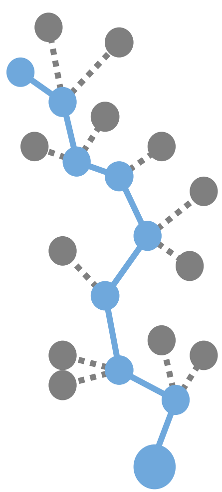
$$N_{k+1}(s,a) = N_k(s,a) + 1$$

Schrittwieser, Julian, et al. "Mastering atari, go, chess and shogi by planning with a learned model." Nature. 2020.

Background

Dyna-Q

- **Mix real and model-generated experiences and apply for additional policy updates.**



Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
(a) $S \leftarrow$ current (nonterminal) state
(b) $A \leftarrow \varepsilon$-greedy(S,Q)
(c) Take action A; observe resultant reward, R, and state, S'
(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
(f) Loop repeat n times:
$S \leftarrow$ random previously observed state
$A \leftarrow$ random action previously taken in S
$R, S' \leftarrow Model(S,A)$
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Update Q values with real transitions

Update Q values with model transitions

Sutton, Richard S. "Dyna, an integrated architecture for learning, planning, and reacting." ACM Sigart Bulletin, 1991.

Background

Model-Based Policy Optimization (MBPO)

- **Dyna-Q updates the policy with both real and model-generated transitions.**
- **How to guarantee the policy improvement if we only use the model-generated transitions ?**

Definition: Errors in the model can be exploited during policy optimization, resulting in large discrepancies between the predicted returns of the policy under the model and under the true dynamics.

$$\eta[\pi] \geq \hat{\eta}[\pi] - C.$$

$\eta[\pi]$: the returns of the policy in true MDP; $\hat{\eta}[\pi]$: the returns under learned model.

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Background

Model-Based Policy Optimization (MBPO)

- **Dyna-Q updates the policy with both real and model-generated transitions.**
- **How to guarantee the policy improvement if we only use the model-generated transitions ?**

Definition: Errors in the model can be exploited during policy optimization, resulting in large discrepancies between the predicted returns of the policy under the model and under the true dynamics.

$$\eta[\pi] \geq \hat{\eta}[\pi] - C.$$

The gap between true and model returns can be expressed in:

- **Generalization Error**: $\epsilon_m = \mathbb{E}_{s,a\sim D_{env}}[D_{TV}(p(s'|s,a)||p_\theta(s'|s,a))]$

- **Policy Distribution**: $\epsilon_\pi = D_{TV}(\pi_D\ (s)||\pi(s))$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$\begin{aligned}
|\eta_1 - \eta_2| &= |\sum_{s,a}(p_1(s,a) - p_2(s,a))r(s,a)| \\
&= |\sum_{s,a}(\sum_t \gamma^t p_1^t(s,a) - p_2^t(s,a))r(s,a)| \\
&= |\sum_t \sum_{s,a} \gamma^t (p_1^t(s,a) - p_2^t(s,a))r(s,a)| \\
&\leq \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)| r(s,a) \\
&\leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)|
\end{aligned}$$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Background

Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$|\eta_1 - \eta_2| \leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)|$$

Lemma:

Lemma B.1 (TVD of Joint Distributions). *Suppose we have two distributions $p_1(x,y) = p_1(x)p_1(y|x)$ and $p_2(x,y) = p_2(x)p_2(y|x)$. We can bound the total variation distance of the joint as:*

$$D_{TV}(p_1(x,y)||p_2(x,y)) \leq D_{TV}(p_1(x)||p_2(x)) + E_{x\sim p_1}[D_{TV}(p_1(y|x)||p_2(y|x))]$$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Background

Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$|\eta_1 - \eta_2| \leq r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)|$$

$$\epsilon_m = \max_t E_{s\sim\pi_{D,t}} \left[D_{TV} \left(p(s', r|s,a) \| p_\theta (s', r|s,a) \right) \right]$$

Lemma:

$$\max_s D_{TV}(\pi \| \pi_D) \leq \epsilon_\pi$$

Lemma B.2 (Markov chain TVD bound, time-varying). *Suppose the expected KL-divergence between two transition distributions is bounded as* $\max_t E_{s\sim p_1^t(s)} D_{KL}(p_1(s'|s) \| p_2(s'|s)) \leq \delta$, *and the initial state distributions are the same –* $p_1^{t=0}(s) = p_2^{t=0}(s)$. *Then the distance in the state marginal is bounded as:*

$$D_{TV}(p_1^t(s) \| p_2^t(s)) \leq t\delta$$

$$\delta = \epsilon_m + \epsilon_\pi$$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Background

Model-Based Policy Optimization (MBPO)

- Dyna-Q updates the policy with both real and model-generated transitions.
- How to guarantee the policy improvement if we only use the model-generated transitions ?

The gap between:

$$
\begin{aligned}
|\eta_1 - \eta_2| &\le r_{\max} \sum_t \sum_{s,a} \gamma^t |p_1^t(s,a) - p_2^t(s,a)| \\
&\le 2r_{\max} \sum_t \gamma^t t(\epsilon_m + \epsilon_\pi) + \epsilon_\pi \\
&\le 2r_{\max}\left(\frac{\gamma(\epsilon_\pi + \epsilon_m)}{(1-\gamma)^2} + \frac{\epsilon_\pi}{1-\gamma}\right)
\end{aligned}
$$

$$D_{TV}(p_1^t(s)||p_2^t(s)) \le t(\epsilon_m + \epsilon_\pi)$$

$$D_{TV}(\pi_1(a|s)||\pi_2(a|s)) \le \epsilon_\pi$$

The returns and model returns of the policy are bounded as:

$$\eta[\pi] \ge \hat{\eta}[\pi] - \underbrace{\left[\frac{2\gamma r_{\max}(\epsilon_m + 2\epsilon_\pi)}{(1-\gamma)^2} + \frac{4r_{\max}\epsilon_\pi}{(1-\gamma)}\right]}_{C(\epsilon_m,\epsilon_\pi)}$$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Model-Based Policy Optimization (MBPO)

- **Dyna-Q updates the policy with both real and model-generated transitions.**
- **How to guarantee the policy improvement if we only use the model-generated transitions ?**
- **H-steps Q-target objective:**

Model-based Critic Loss Function:

$$Loss_{critic} = \frac{1}{2}\left(y - Q(s_t, a_t)\right)^2$$

Dyna-Q one-step Q-target:

$$y = \hat{r}(s_t, a_t) + \gamma Q(\hat{s}_{t+1}, \hat{a}_{t+1})$$

Multi-steps Q-target:

$$y = \sum_{k=t}^{H-1} \gamma^{k-t} \hat{r}_k + \gamma^H Q(\hat{s}_H, \hat{a}_H)$$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Background

Model-Based Policy Optimization (MBPO)

- **Dyna-Q updates the policy with both real and model-generated transitions.**
- **How to guarantee the policy improvement if we only use the model-generated transitions ?**
- **H-steps Q-target objective**
- **MBPO Algorithm:**

Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

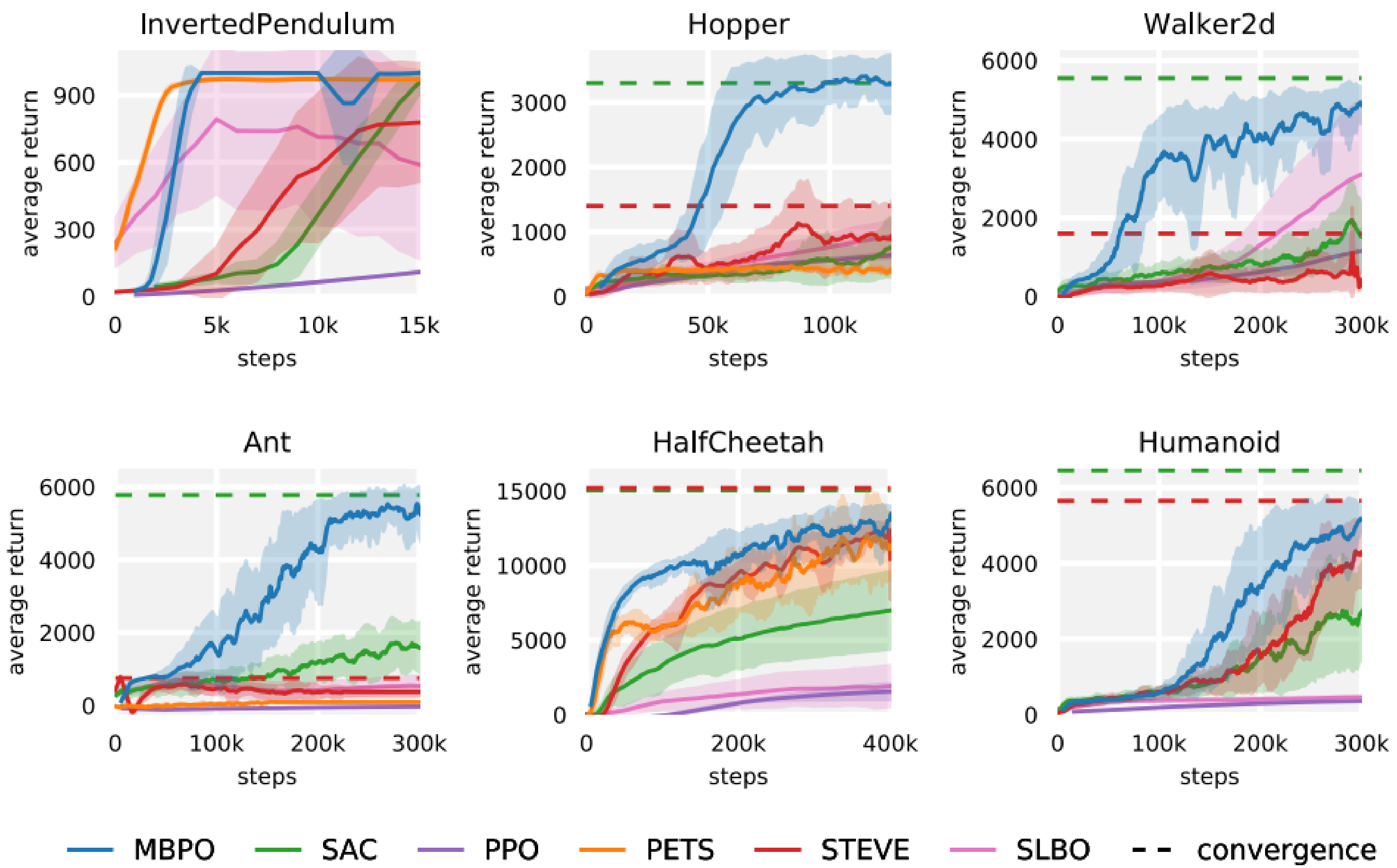
Initialize policy π_ϕ, predictive model p_θ, environment dataset $\mathcal{D}_{\text{env}}$, model dataset $\mathcal{D}_{\text{model}}$
for N epochs **do**
  Train model p_θ on $\mathcal{D}_{\text{env}}$ via maximum likelihood
  for E steps **do**
    Take action in environment according to π_ϕ; add to $\mathcal{D}_{\text{env}}$
    for M model rollouts **do**
      Sample s_t uniformly from $\mathcal{D}_{\text{env}}$
      Perform k-step model rollout starting from s_t using policy π_ϕ; add to $\mathcal{D}_{\text{model}}$
    for G gradient updates **do**
      Update policy parameters on model data: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$

Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Background

Model-Based Policy Optimization (MBPO)

Experiments



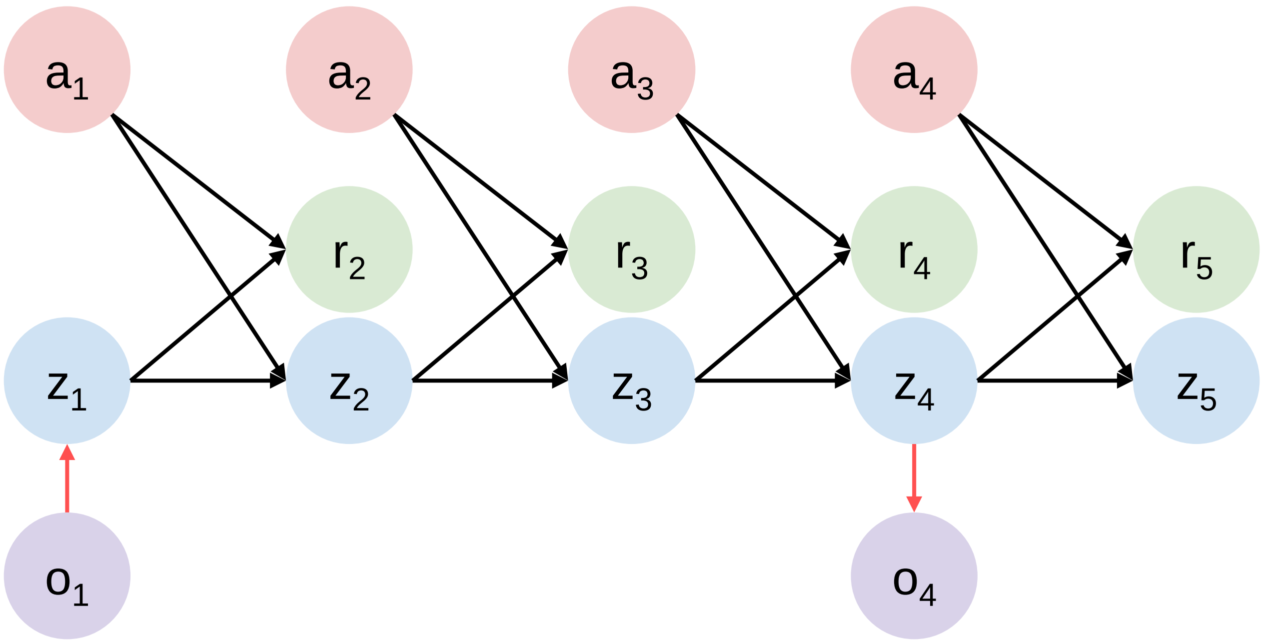
Janner, Michael, et al. "When to trust your model: Model-based policy optimization." NIPS, 2019.

Model-based RL with Images

Dreamer: Latent Imagination

- Latent state-transition models

Reconstruct high-level observation to low-level **latent state**

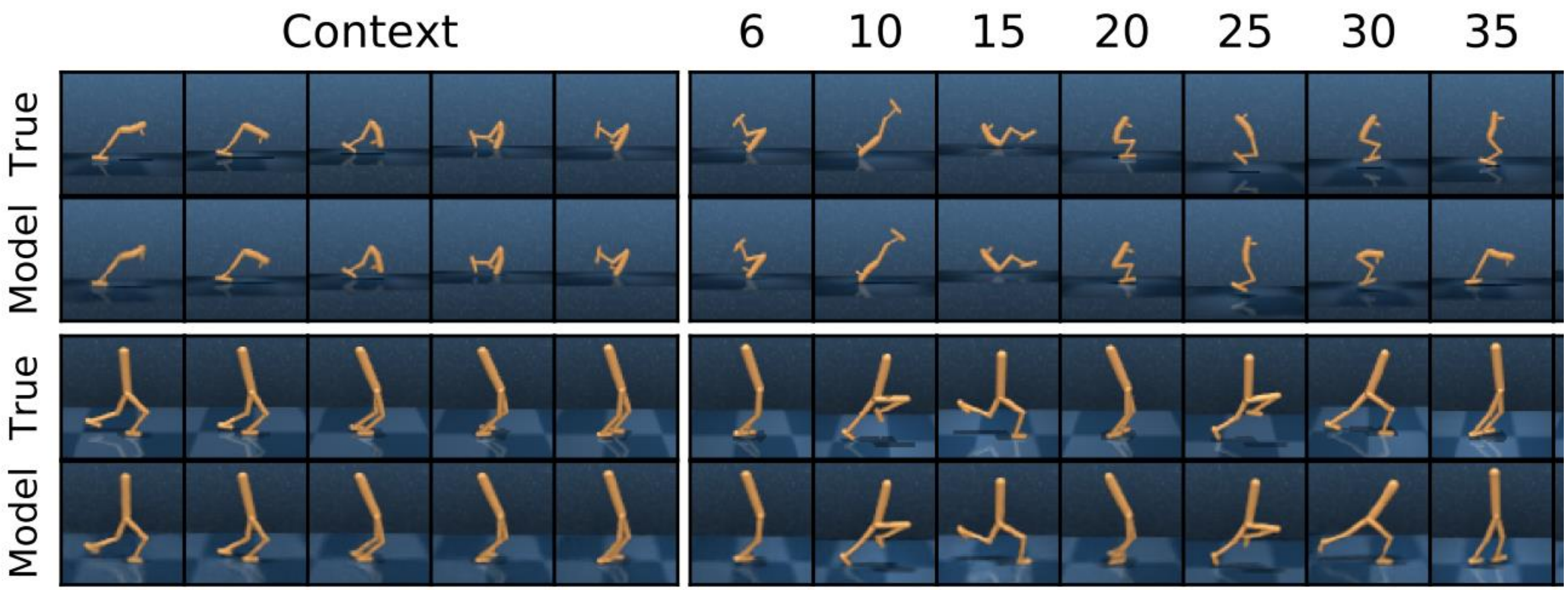


Hafner, Danijar, et al. "Dream to control: Learning behaviors by latent imagination." arXiv preprint arXiv:1912.01603 (2019).

Model-based RL with Images

Dreamer: Latent Imagination

- Latent state-transition models

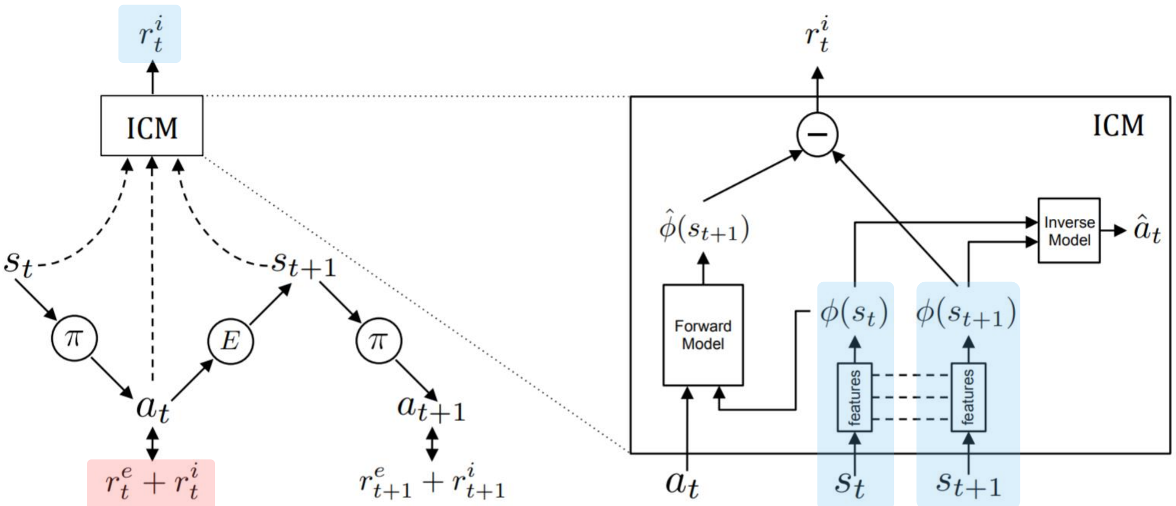


Hafner, Danijar, et al. "Dream to control: Learning behaviors by latent imagination." arXiv preprint arXiv:1912.01603 (2019).

Inverse Model

Intrinsic Curiosity Module (ICM)

- Besides extrinsic reward, agent set **intrinsic reward** to express its familiarity with the true environment. (Curiosity Driven)
- Model Prediction: $f(s_t, a_t) \rightarrow s_{t+1}$
- Intrinsic Reward: $r_t^{in} \sim \| f(s_t, a_t) - s_{t+1} \|_2^2$



- **Forward Dynamics Model:**

  $f_{\psi F}(\phi(s_t), a_t) \rightarrow \hat{\phi}(s_{t+1})$

- **Inverse Model:**

  $g_{\psi I}(\phi(s_t), \phi(s_{t+1})) \rightarrow \hat{a}_t.$

- **Intrinsic Reward:**

  $r_t^i = \left\| \hat{\phi}(s_{t+1}) - \phi(s_{t+1}) \right\|_2^2$

Pathak, Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." International conference on machine learning. ICML, 2017.

Deep Reinforcement Learning

谢 谢！

中山大学计算机学院