



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY

Lecture 9, ACM RL Course 2024

# Model-based Reinforcement Learning

Weinan Zhang

Shanghai Jiao Tong University

<http://wnzhang.net>

Apr. 2024

# 2024年上海交通大学ACM班强化学习课程大纲

## 强化学习基础部分

(中文课件)

1. 强化学习、探索与利用
2. MDP和动态规划
3. 值函数估计
4. 无模型控制方法
5. 规划与学习
6. 参数化的值函数和策略
7. 深度强化学习价值方法
8. 深度强化学习策略方法

## 强化学习前沿部分

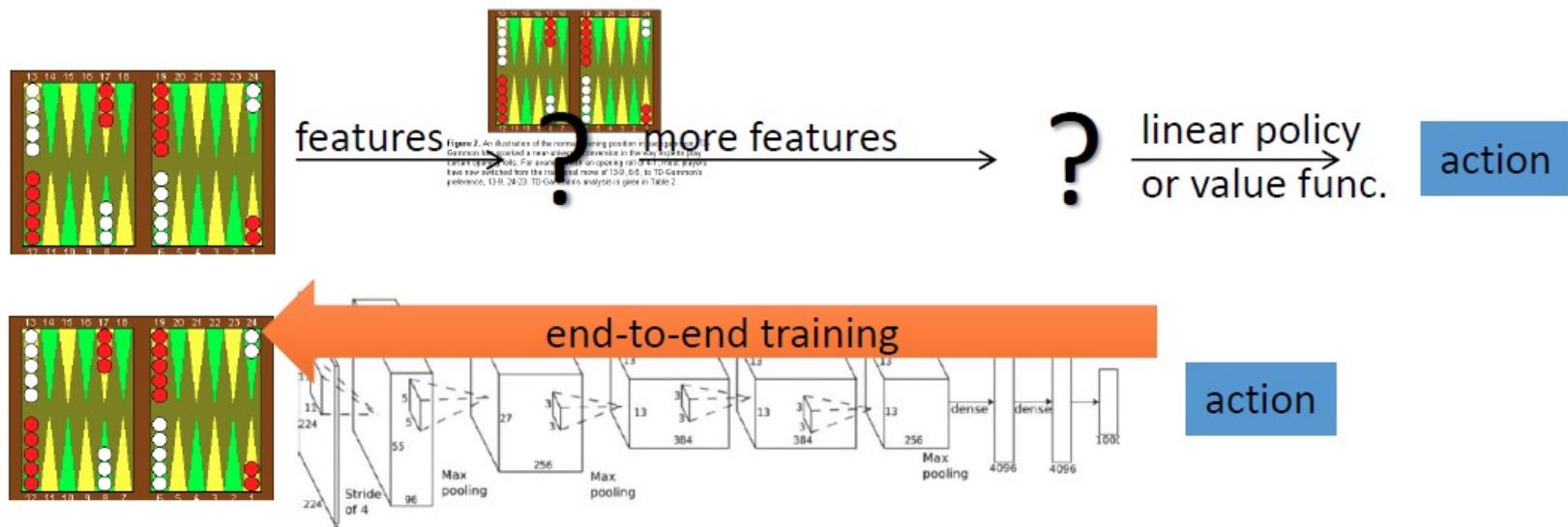
(英文课件)

9. 基于模型的深度强化学习
10. 离线强化学习
11. 模仿学习
12. 多智能体强化学习基础
13. 多智能体强化学习前沿
14. 基于扩散模型的强化学习
15. AI Agent与决策大模型
16. 技术交流与回顾

A Perspective of

# Overall Pathway of DRL

- Deep reinforcement learning gets appealing success
  - Atari, AlphaGo, DOTA 2, AlphaStar

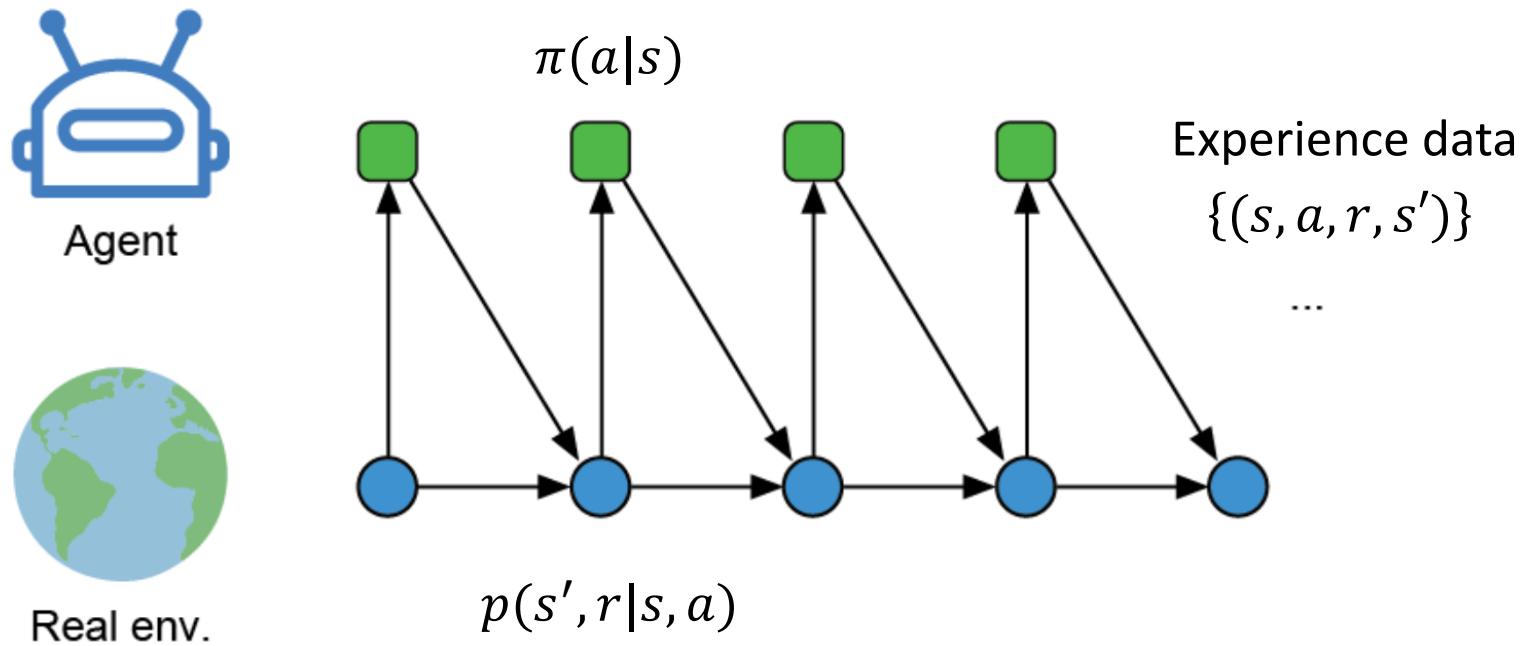


A Perspective of

# Overall Pathway of DRL

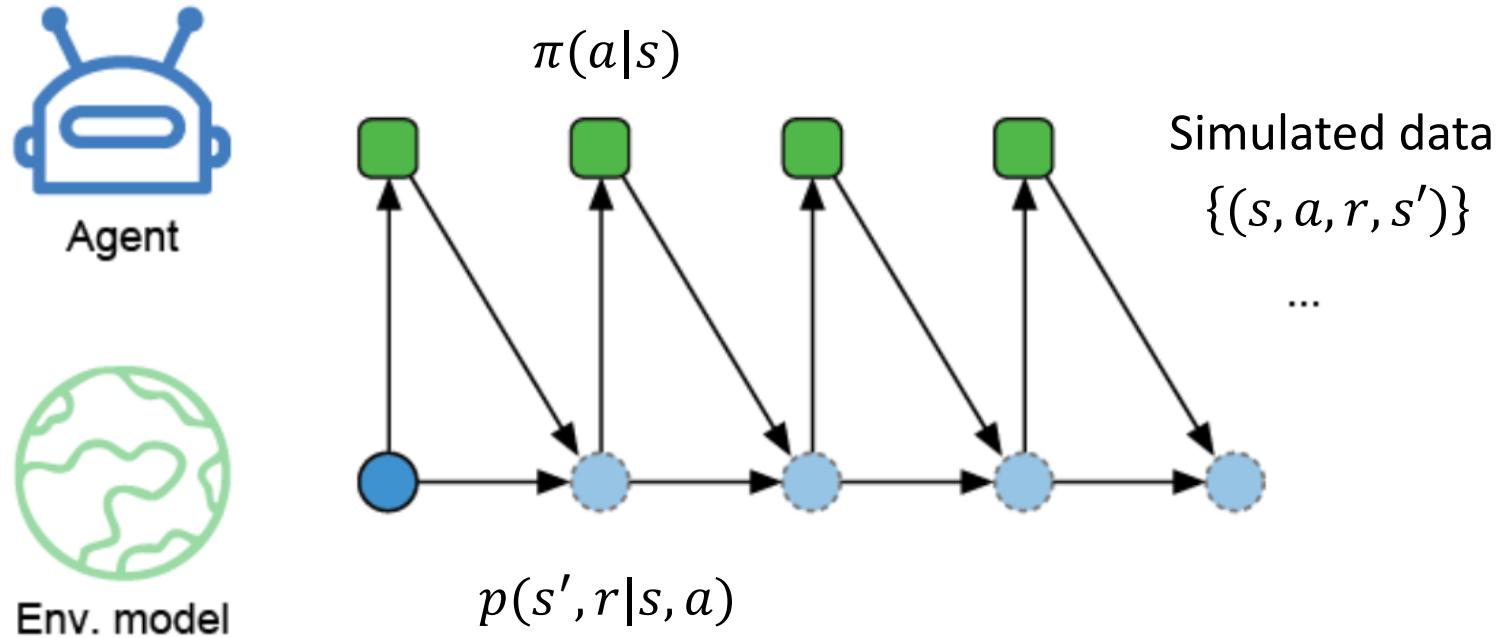
- Deep reinforcement learning gets appealing success
  - Atari, AlphaGo, DOTA 2, AlphaStar
- But DRL has very low data efficiency
  - Trial-and-error learning for deep networks
- A recent popular direction is model-based RL
  - Build a model  $p(s', r|s, a)$
  - Based on the model to train the policy
  - So that the data efficiency could be improved

# Interaction between Agent and Environment



- Real environment
  - State dynamics  $p(s'|s, a)$
  - Reward function  $r(s, a)$

# Interaction between Agent and Env. Model



- Real environment
  - State dynamics  $p(s'|s, a)$
  - Reward function  $r(s, a)$
- Environment model
  - State dynamics  $\hat{p}(s'|s, a)$
  - Reward function  $\hat{r}(s, a)$

# Model-free RL v.s. Model-based RL

- Model-based RL
  - On-policy learning once the model is learned
  - May not need further real interaction data once the model is learned (batch RL)
  - Always show higher sample efficiency than MFRL
  - Suffer from model compounding error
- Model-free RL
  - The best asymptotic performance
  - Highly suitable for DL architecture with big data
  - Off-policy methods still show instabilities
  - Very low sample efficiency & require huge amount of training data

# Model-based RL: Blackbox and Whitebox

## Model as a Blackbox

- Seamless to policy training algorithms
- The simulation data efficiency may still be low
- E.g., Dyna-Q, MPC, MBPO

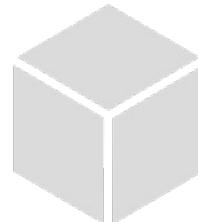
Focus of today's talk



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

## Model as a Whitebox

- Offer both data and gradient guidance for value and policy
- High data efficiency
- E.g., MAAC, SVG, PILCO



$$\rightarrow \frac{\partial V(s')}{\partial s'} \frac{\partial s'}{\partial a} \frac{\partial a}{\partial \theta} \Big|_{s' = f_\phi(s, a), a = \pi(s)}$$

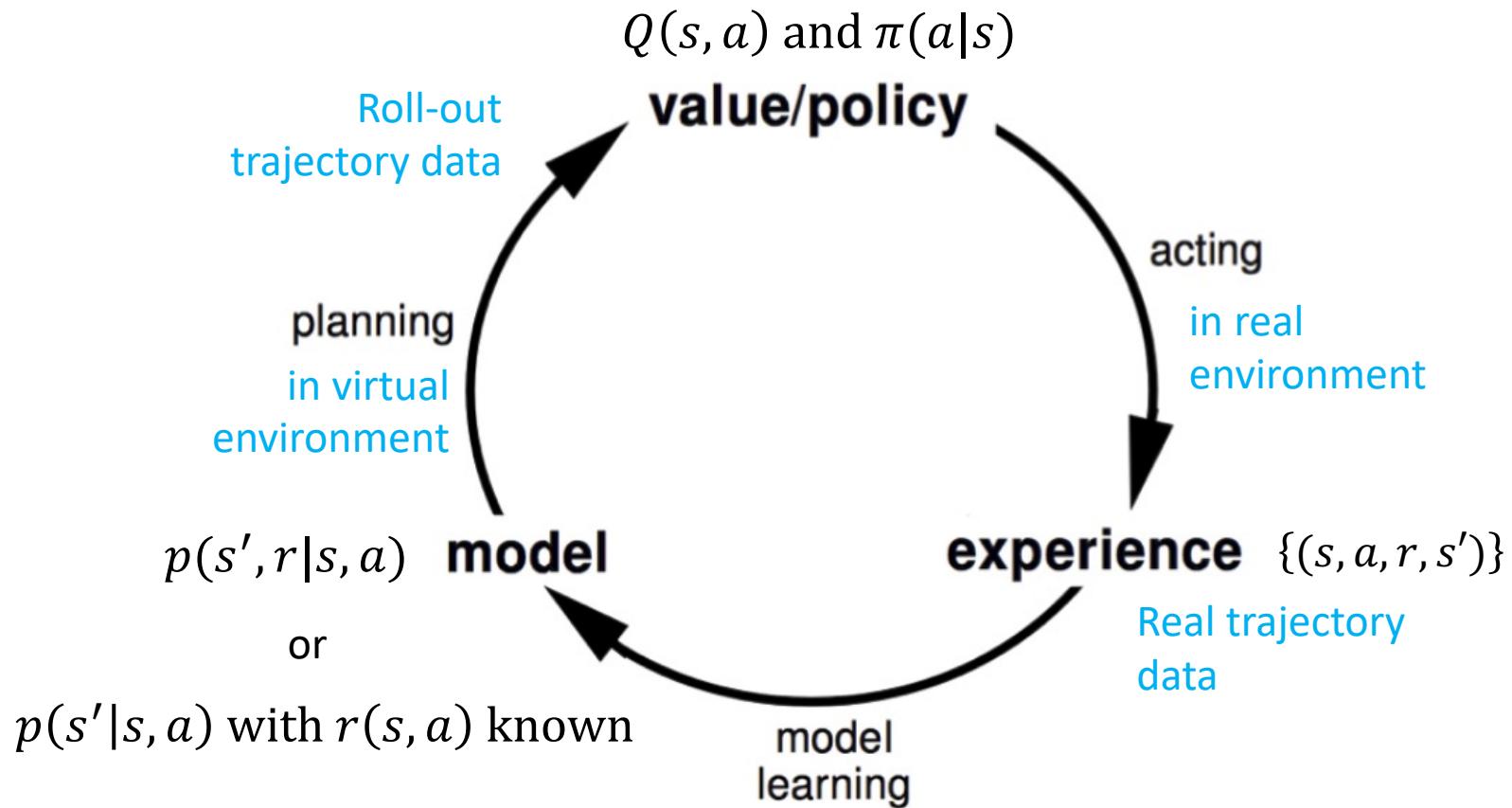
$$\sim\{(s, a, r, s')\}$$

# Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: BMPO, AMPO and AutoMBPO

Appendix: Backpropagation through paths: SVG and MAAC

# Model-based RL



Annotations based on Rich Sutton's figure

# Q-Planning

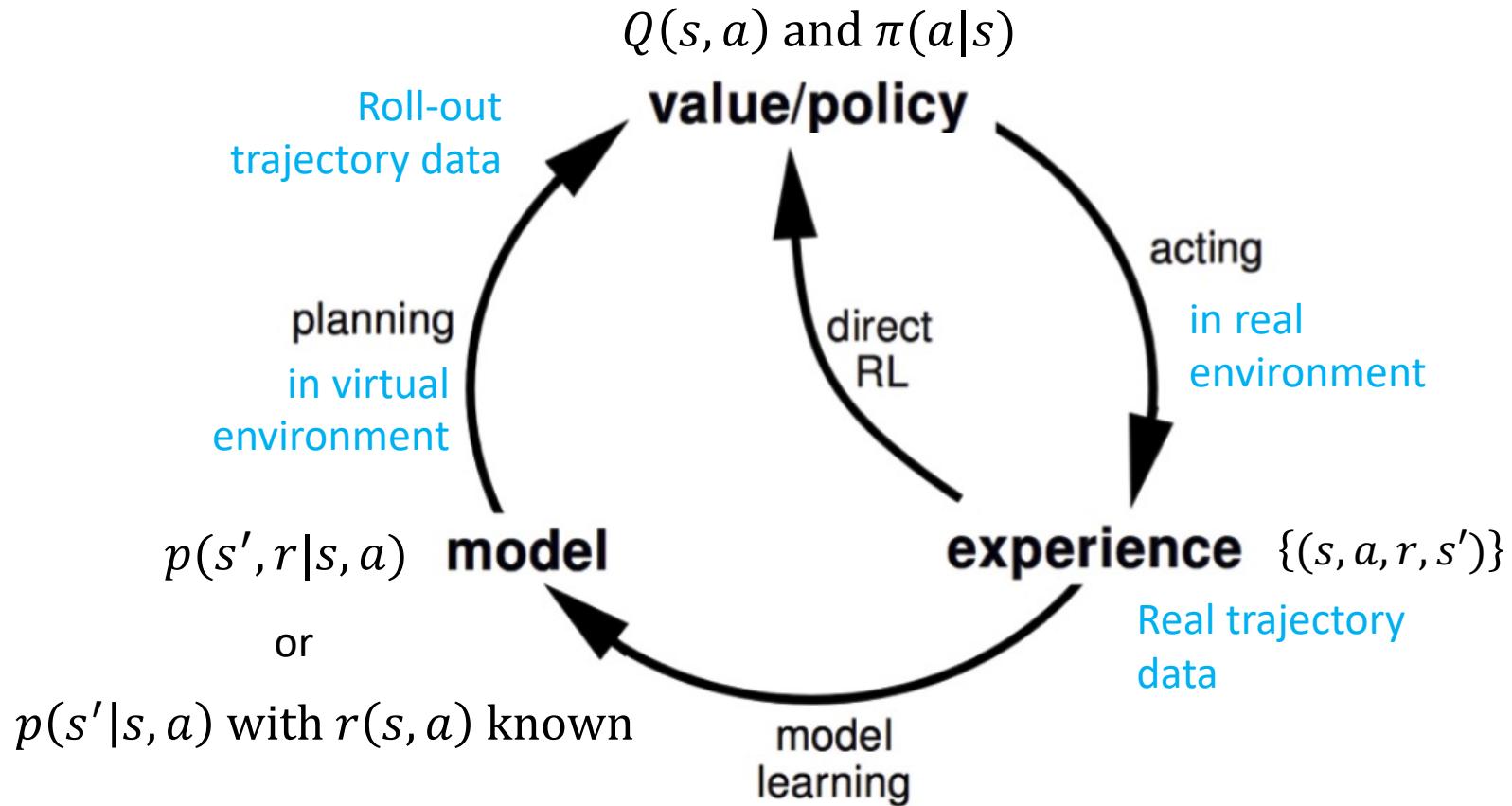
- Random-sample one-step tabular Q-planning
  - First, learn a model  $p(s',r|s,a)$  from experience data
  - Then perform one-step sampling by the model to learn the Q function

Do forever:

1. Select a state,  $S \in \mathcal{S}$ , and an action,  $A \in \mathcal{A}(s)$ , at random
2. Send  $S, A$  to a sample model, and obtain
  - a sample next reward,  $R$ , and a sample next state,  $S'$
3. Apply one-step tabular Q-learning to  $S, A, R, S'$ :  
$$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$$

- Here model learning and reinforcement learning are separate

# Dyna



Annotations based on Rich Sutton's figure

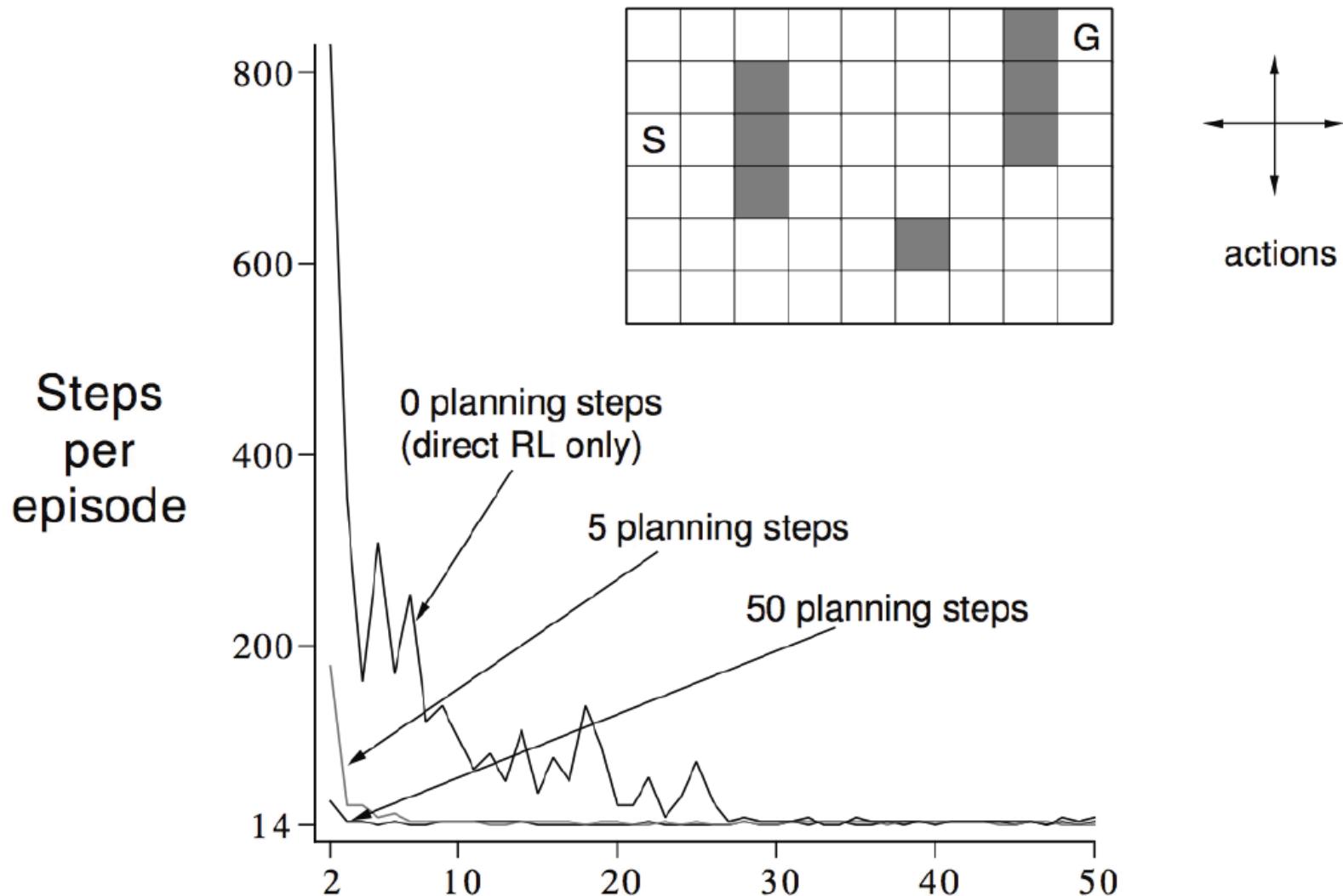
# Dyna-Q

Initialize  $Q(s, a)$  and  $Model(s, a)$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$

Do forever:

- (a)  $S \leftarrow$  current (nonterminal) state
- (b)  $A \leftarrow \varepsilon\text{-greedy}(S, Q)$
- (c) Execute action  $A$ ; observe resultant reward,  $R$ , and state,  $S'$
- (d)  $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
- (e)  $Model(S, A) \leftarrow R, S'$  (assuming deterministic environment)
- (f) Repeat  $n$  times:
  - $S \leftarrow$  random previously observed state
  - $A \leftarrow$  random action previously taken in  $S$
  - $R, S' \leftarrow Model(S, A)$
  - $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$

# Dyna-Q on a Simple Maze



# Key Questions of Deep MBRL

- How to properly **train the deep model** based on the agent-environment interaction data?
- Inevitably, the model is **to-some-extent inaccurate**. When to trust the model?
- How to **effectively use the model** to better train our policy?
- Does the model really help improve the **data efficiency**?

# Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: BMPO, AMPO and AutoMBPO

Appendix: Backpropagation through paths: SVG and MAAC

# Shooting Methods

Model can also be used to help **decision making** when interact with environment. For the current state  $s_0$ :

- Given an action sequence of length  $T$ :

$$[a_0, a_1, a_2, \dots, a_T]$$

- we can sample trajectory from the model:

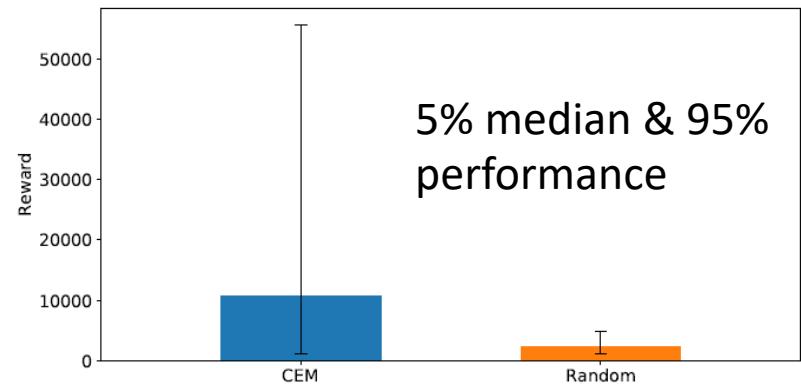
$$[s_0, a_0, \hat{r}_0, \hat{s}_1, a_1, \hat{r}_1, \hat{s}_2, a_2, \hat{r}_2, \dots, \hat{s}_T, a_T, \hat{r}_T]$$

- And then choose the action sequence with highest estimated return:

$$\hat{Q}(s, a) = \sum_{t=0}^T \gamma^t \hat{r}_t \quad \pi(s) = \arg \max_a \hat{Q}(s, a)$$

# Random Shooting (RS)

- The action sequences are randomly sampled
- Pros:
  - implementation simplicity
  - lower computational burden (no gradients)
  - no requirement to specify the task-horizon in advance
- Cons: high variance, may not sample the high reward action
- A refinement:  
**Cross Entropy Method (CEM)**
  - CEM samples actions from a distribution closer to previous action samples that yield high reward.

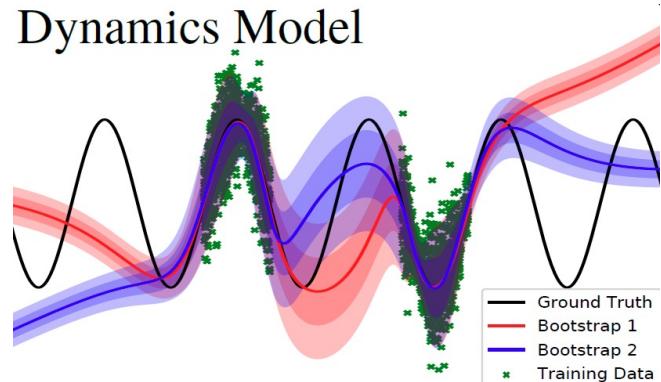


# PETS: Probabilistic Ensembles with Trajectory Sampling

Ensemble  $\text{loss}_{\text{P}}(\theta) = - \sum_{n=1}^N \log \tilde{f}_{\theta}(s_{n+1} | s_n, a_n)$

Gaussian NN  $\tilde{f} = \Pr(s_{t+1} | s_t, a_t) = \mathcal{N}(\mu_{\theta}(s_t, a_t), \Sigma_{\theta}(s_t, a_t))$

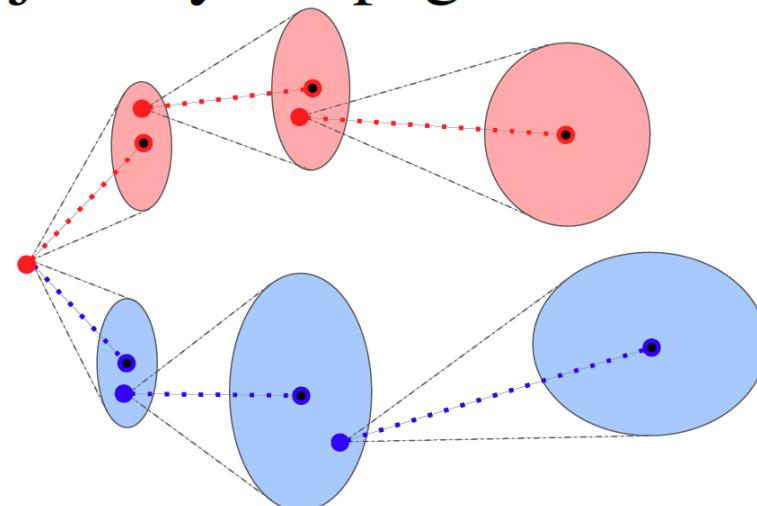
- Probabilistic ensemble (PE) dynamics model is shown as an ensemble of two bootstraps
  - Bootstrap disagreement far from data captures **epistemic** uncertainty: our subjective uncertainty due to a lack of data (**model variance**)
  - Each a probabilistic neural network that captures **aleatoric** uncertainty (**stochastic environment**).



# PETS: Probabilistic Ensembles with Trajectory Sampling

- The trajectory sampling (TS) propagation technique uses our dynamics model to re-sample each particle (**with associated bootstrap**) according to its probabilistic prediction at each point in time, up until horizon  $T$

## Trajectory Propagation

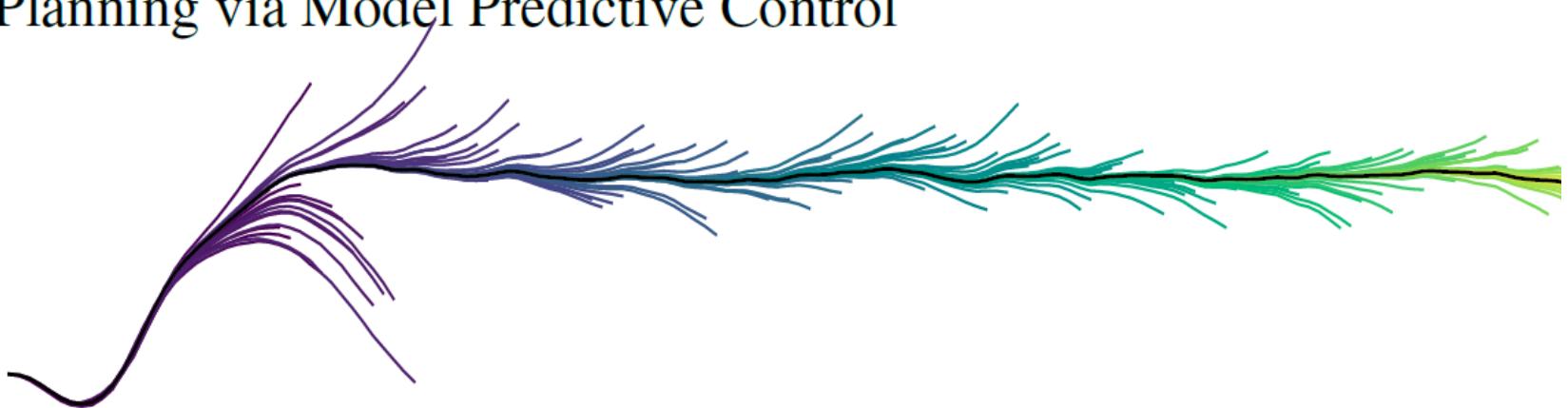


# PETS: Probabilistic Ensembles with Trajectory Sampling

- Planning:

At each time step, MPC algorithm **computes an optimal action sequence by sampling multiple sequences**, applies **the first action** in the sequence, and repeats until the task-horizon.

## Planning via Model Predictive Control



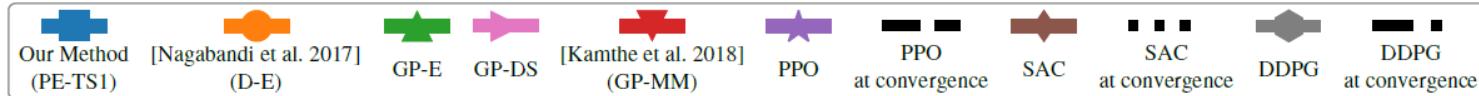
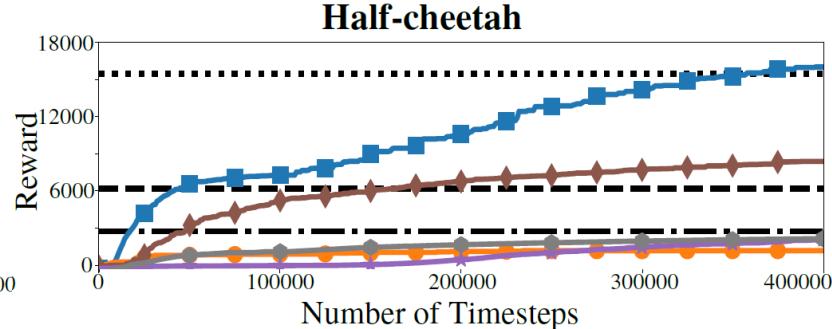
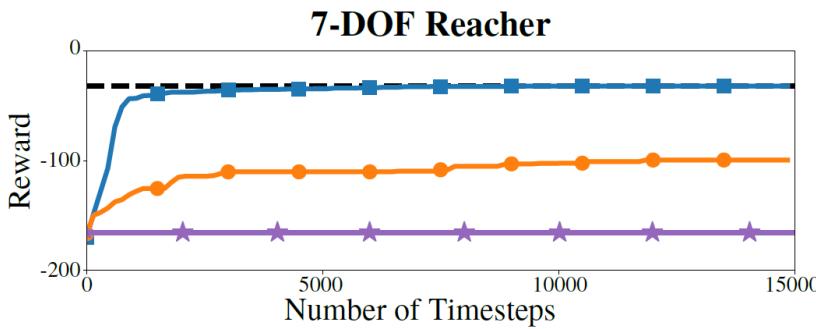
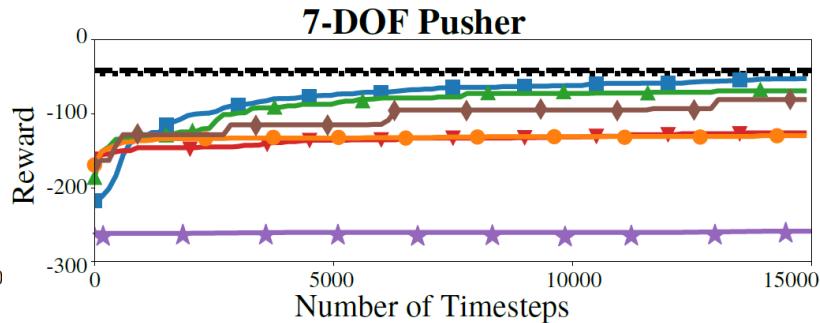
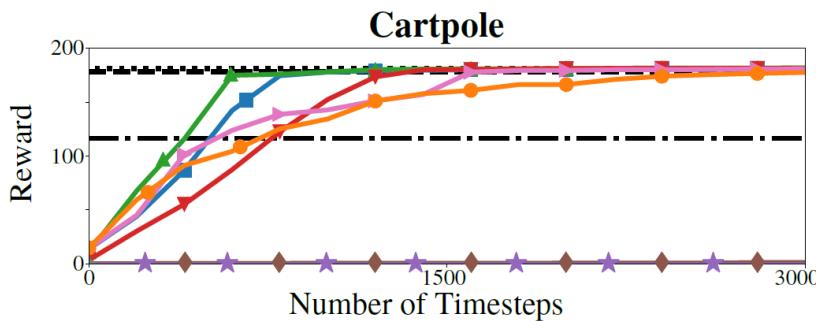
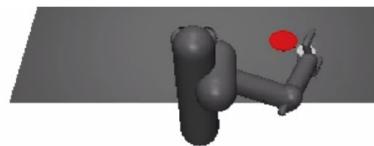
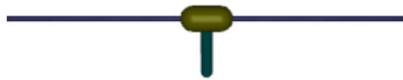
# PETS Algorithm

---

**Algorithm 1** Our model-based MPC algorithm ‘*PETS*’:

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

# PETS Experiments



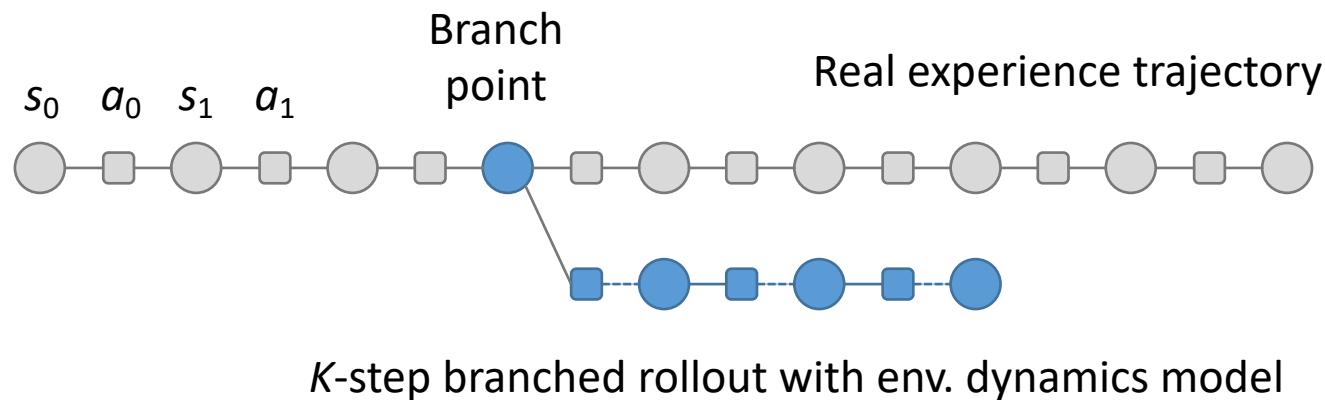
# Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: BMPO, AMPO and AutoMBPO

Appendix: Backpropagation through paths: SVG and MAAC

# Bound based on Model & Policy Error

- Branched rollout
  - Begin a rollout from a state under the previous policy's state distribution  $d_{\pi_D}(s)$  and run  $k$  steps according to  $\pi$  under the learned model  $p_\theta$
- Dyna can be viewed as a special case of  $k = 1$  branched rollout



# Bound based on Model & Policy Error

- Quantify **model error** and **policy shift** as

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

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

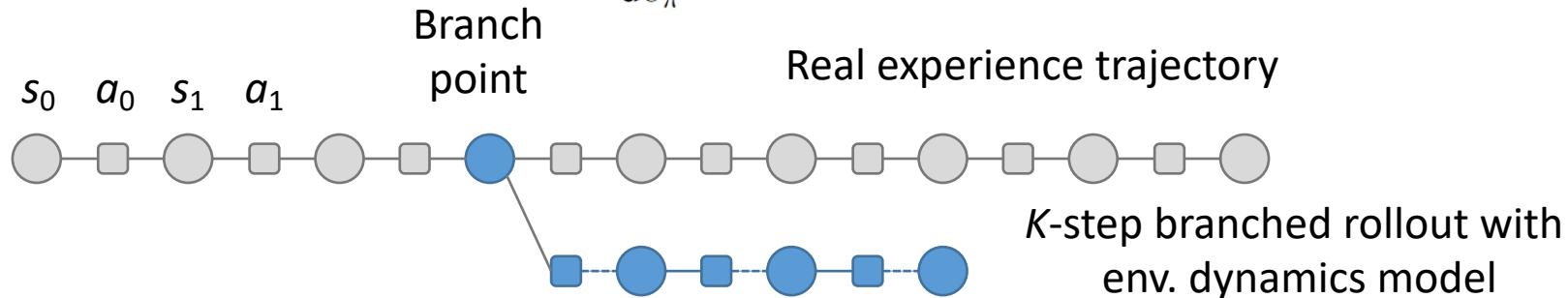
→ based on old policy sampled data

$$\epsilon_{m'} = \max_t \mathbb{E}_{s \sim \pi_t} [D_{TV}(p(s', r | s, a) \| p_\theta(s', r | s, a))]$$

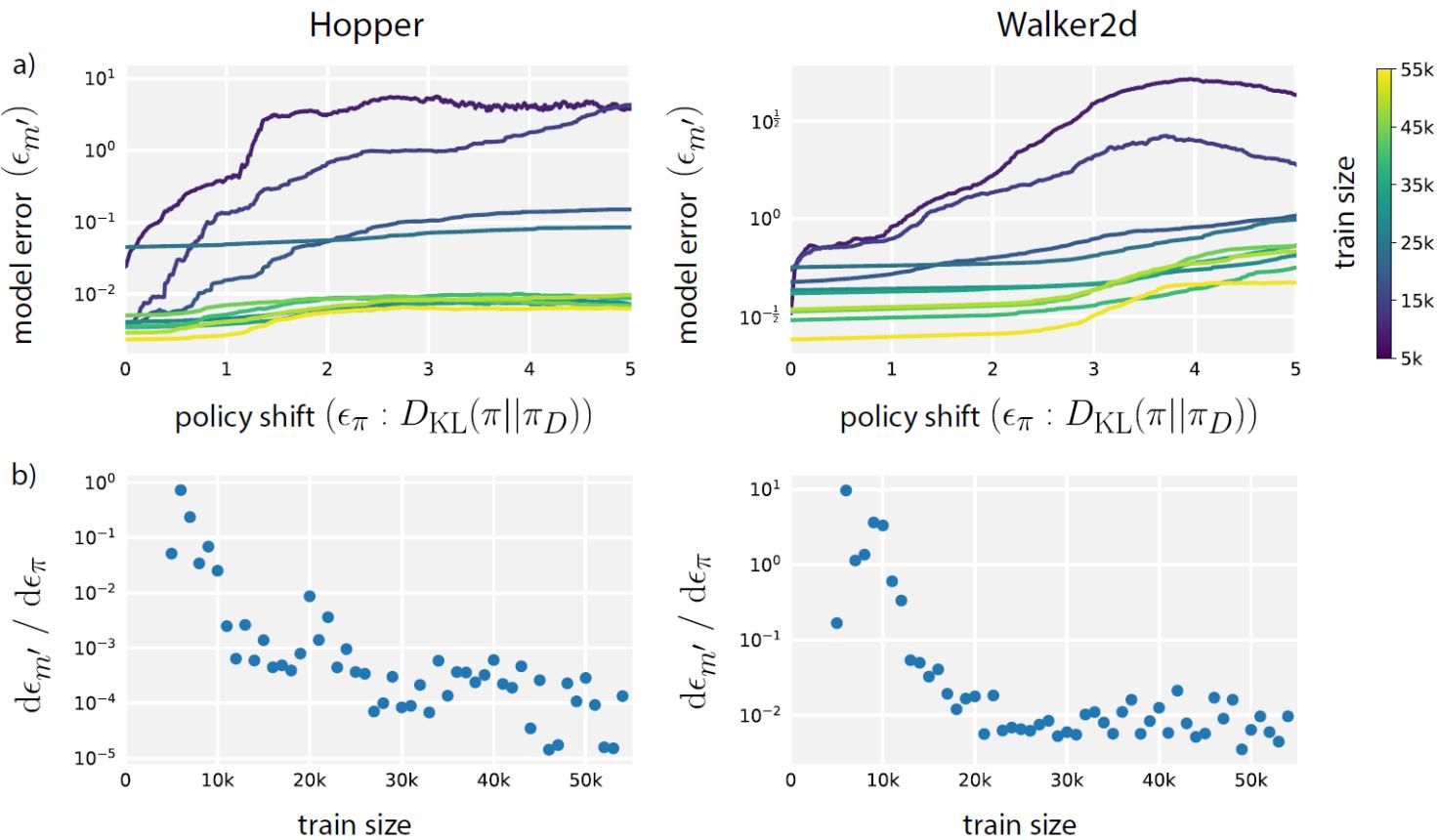
- The policy value discrepancy bound is written as

$$\eta[\pi] \geq \eta^{\text{branch}}[\pi] - 2r_{\max} \left[ \frac{\gamma^{k+1} \epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k \epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma} (\epsilon_{m'}) \right]$$

where **the optimal  $k > 0$**  if  $\frac{d\epsilon_{m'}}{d\epsilon_\pi}$  is sufficiently small



# Empirical Analysis of $\frac{d\epsilon_{m'}}{d\epsilon_\pi}$



$$\eta[\pi] \geq \eta^{\text{branch}}[\pi] - 2r_{\max} \left[ \frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k\epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}(\epsilon_{m'}) \right]$$

where the optimal  $k > 0$  if  $\frac{d\epsilon_{m'}}{d\epsilon_\pi}$  is sufficiently small

# MBPO Algorithm

---

## Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

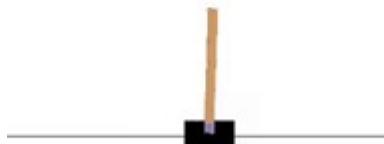
---

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

- **Remarks**

- Branch out from the real trajectories (instead from  $s_0$ )
- Branch rollout  $k$  steps depends on model & policy
- Use a model-free RL method (Soft AC) to update policy

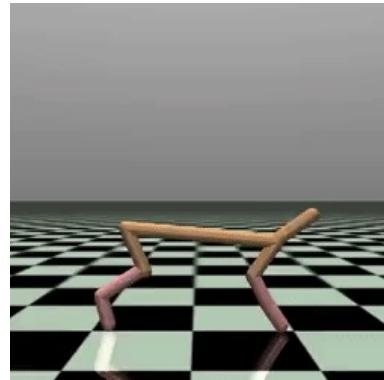
# Experiment Environments



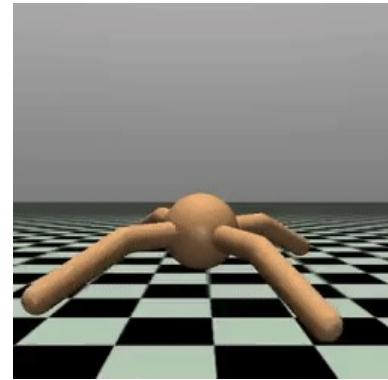
Cartpole



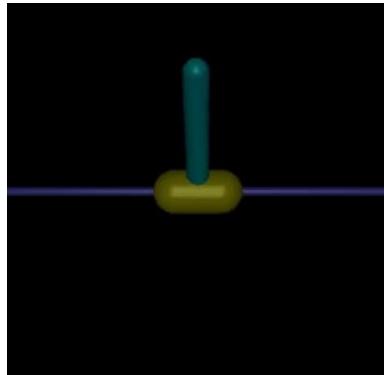
Swimmer



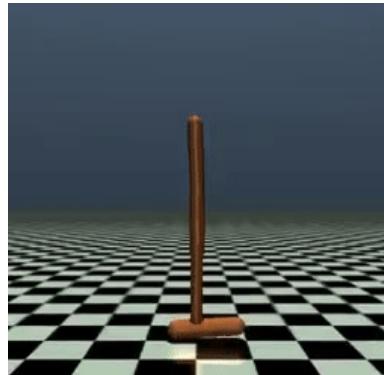
HalfCheetah



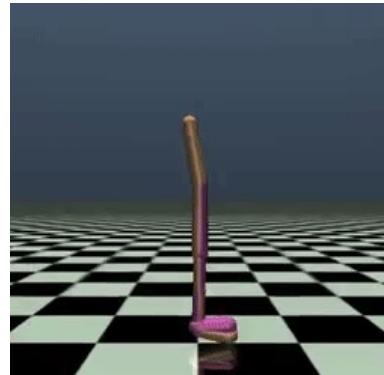
Ant



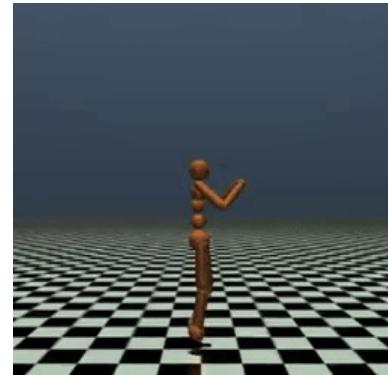
InvertedPendulum



Hopper

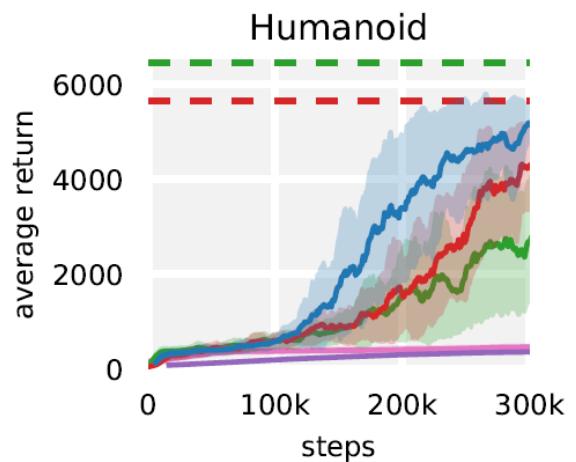
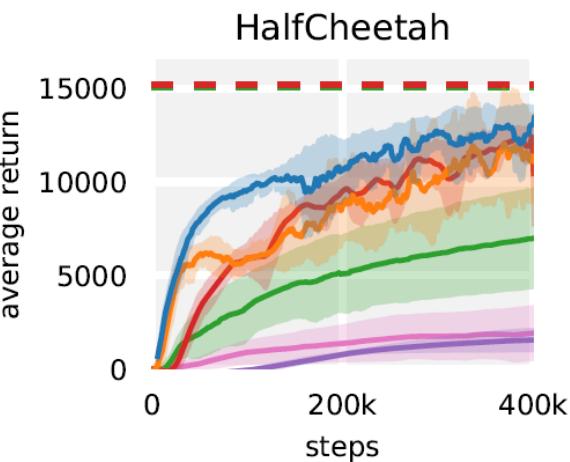
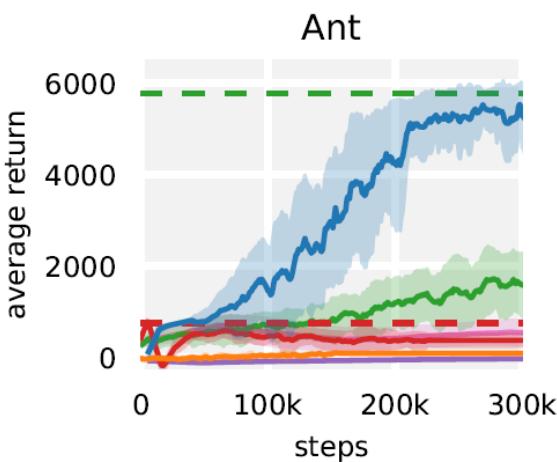
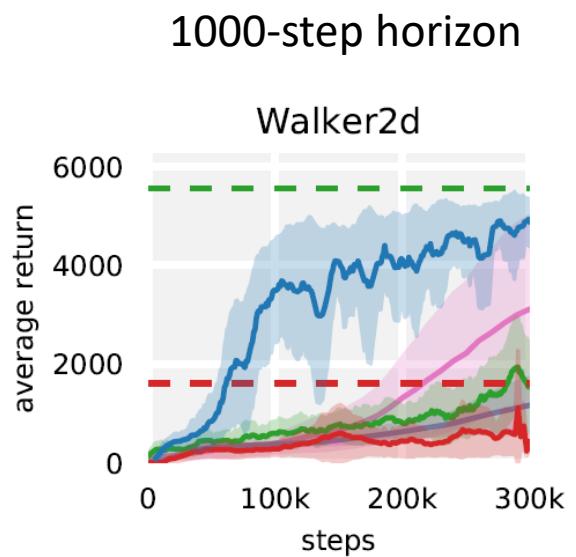
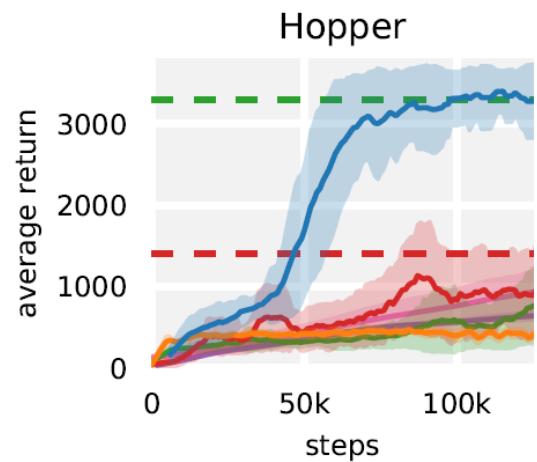
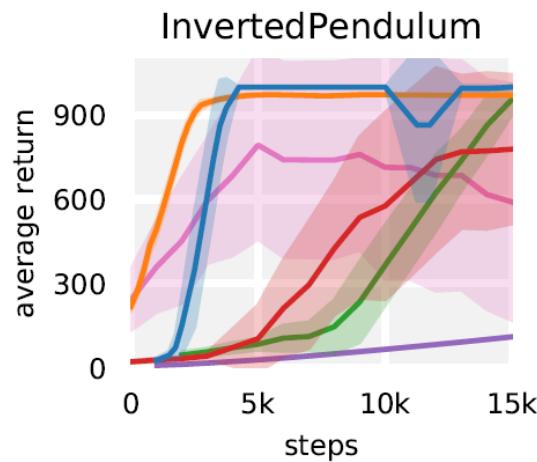


Walker2d



Humanoid

# MBPO Experiments



— MBPO   — SAC   — PPO   — PETS   — STEVE   — SLBO   - - convergence

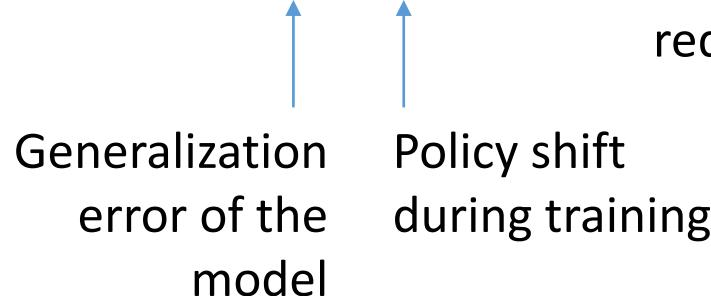
# Summary of Theoretic Analysis in MBRL

1. The qualitative relationship between policy value discrepancy and sample efficiency

Lower  $|\eta[\pi] - \hat{\eta}[\pi]| \rightarrow$  more use of model  $\rightarrow$  higher sample efficiency

2. The quantitative relationship between model error (and policy shift) and policy value discrepancy

$$|\eta[\pi] - \hat{\eta}[\pi]| \leq C(\epsilon_m, \epsilon_\pi)$$



1. Derive the bound
2. Design algorithms to reduce the  $C$  term

# Content

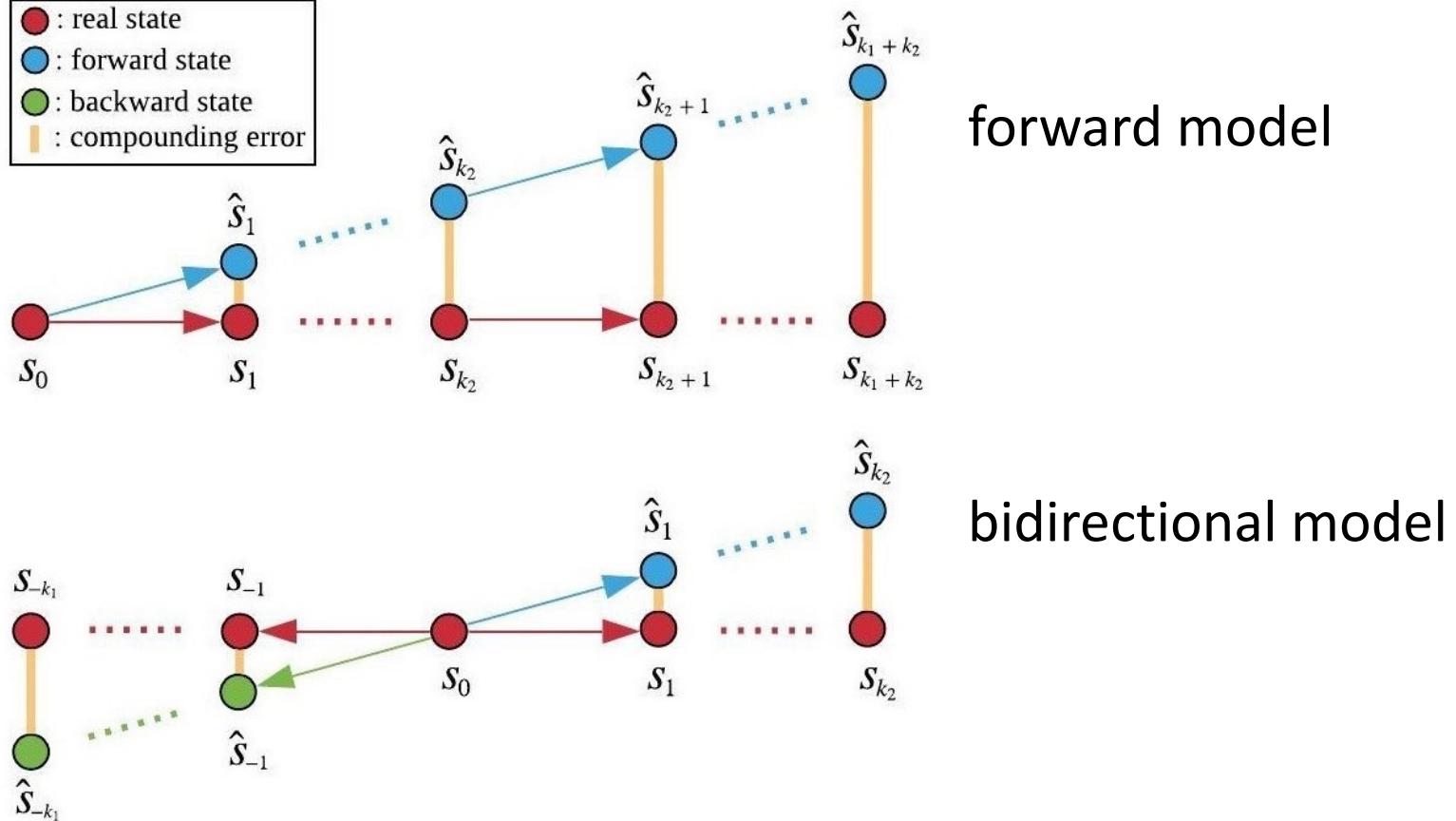
1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: [BMPO](#), AMPO and AutoMBPO

Appendix: Backpropagation through paths: SVG and MAAC

# Bidirectional Model

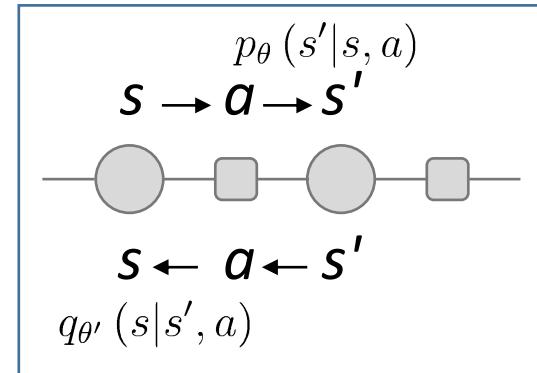
**Key finding:** Generating trajectories with the **same length**, the compounding error of the **bidirectional model** will be less than that of the **forward model**.

- : real state
- : forward state
- : backward state
- : compounding error



# Dynamics Model Learning

- An ensemble of bootstrapped probabilistic networks are used to parameterize both the forward model  $p_\theta(s'|s, a)$  and the backward model  $q_{\theta'}(s|s', a)$ .
- Each probabilistic neural network outputs a Gaussian distribution with diagonal covariance and is trained via maximum likelihood. The corresponding loss functions are:



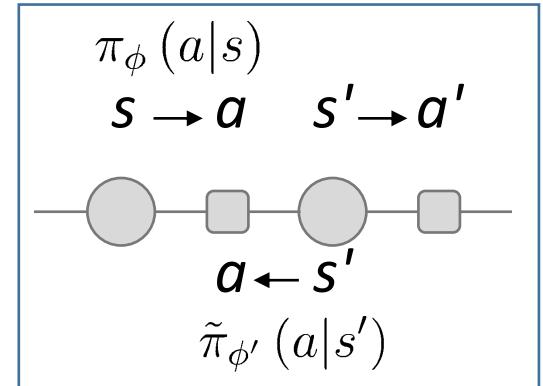
$$\mathcal{L}_f(\theta) = \sum_{t=1}^N [\mu_\theta(s_t, a_t) - s_{t+1}]^\top \Sigma_\theta^{-1}(s_t, a_t) [\mu_\theta(s_t, a_t) - s_{t+1}] + \log \det \Sigma_\theta(s_t, a_t)$$

$$\mathcal{L}_b(\theta') = \sum_{t=1}^N [\mu_{\theta'}(s_{t+1}, a_t) - s_t]^\top \Sigma_{\theta'}^{-1}(s_{t+1}, a_t) [\mu_{\theta'}(s_{t+1}, a_t) - s_t] + \log \det \Sigma_{\theta'}(s_{t+1}, a_t)$$

where  $\mu$  and  $\Sigma$  are the mean and covariance respectively, and  $N$  denotes the total number of real transition data.

# Backward Policy

- In the forward model rollout, actions are selected by the current policy  $\pi_\phi(a|s)$ .
- To sample trajectories backwards, we need to learn a backward policy  $\tilde{\pi}_{\phi'}(a|s')$  to take actions given the next state.



- The backward policy can be trained by either **maximum likelihood estimation**:

$$\mathcal{L}_{MLE}(\phi') = - \sum_{t=0}^N \log \tilde{\pi}_{\phi'}(a_t | s_{t+1})$$

- or conditional GAN (GAIL):

$$\min_{\tilde{\pi}} \max_D V(D, \tilde{\pi}) = \mathbb{E}_{(a, s') \sim \pi} [\log D(a, s')] + \mathbb{E}_{s' \sim \pi} [\log (1 - D(\tilde{\pi}(\cdot | s'), s'))]$$

# Other Components of BMPO

- **State Sampling Strategy:** instead of random chosen states from environment replay buffer, we sample **high value states to begin rollouts** according to a Boltzmann distribution.

$$p(s) \propto e^{\beta V(s)}$$

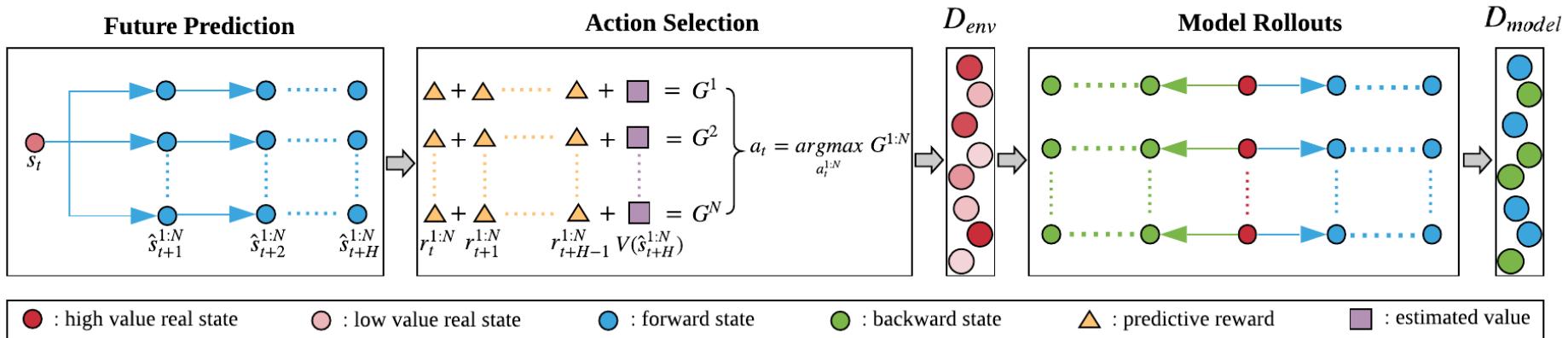
- Thus, the agent could learn to reach high-value states through backward rollouts, and also learn to act better after these states through forward rollouts
- **Incorporating MPC** to refine the action taken in real environment.
  - At each time-step, candidate action sequences are generated from **current policy** and the corresponding trajectories are simulated by the learned model.
  - Then the first action of the sequence that yields the highest accumulated rewards is selected:

$$a_t = \operatorname{argmax}_{\substack{a_t^{1:N} \sim \pi}} \sum_{t'=t}^{t+H-1} \gamma^{t'-t} r(s_{t'}, a_{t'}) + \gamma^H V(s_{t+H})$$



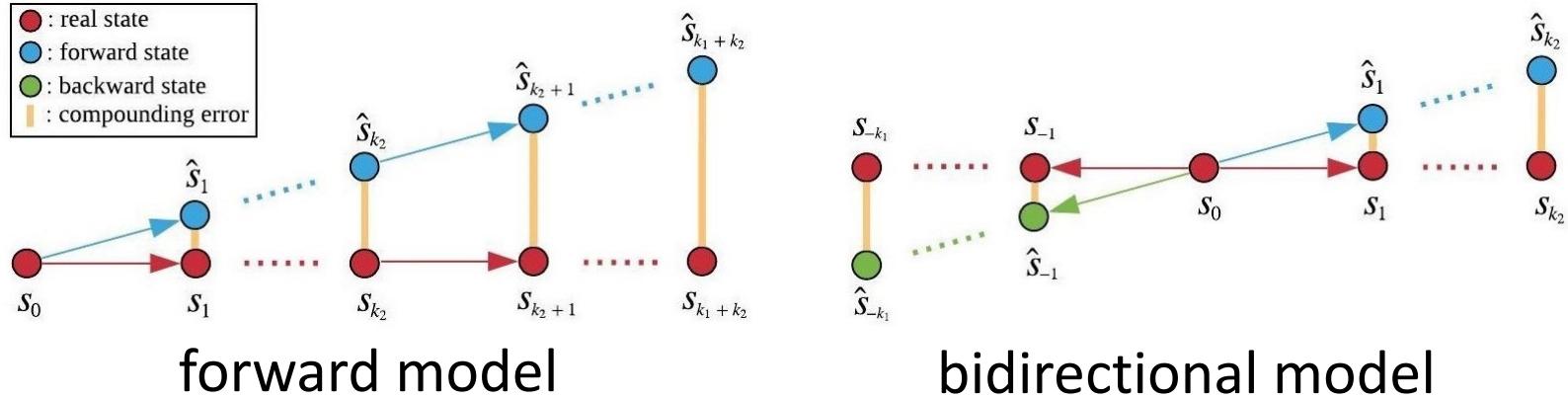
Truncate the rollout with value function

# Overall Algorithm of BMPO



1. When interacting with the environment, the agent uses the model to perform MPC-based action selection
2. Data is stored in  $D_{env}$ , where the value of state increases from light red to dark red
3. High value states are then sampled from  $D_{env}$  to perform bidirectional model rollouts, which are stored in  $D_{model}$
4. Model-free method (e.g., soft actor-critic) is used to train the policy based on the data from  $D_{model}$

# Theoretical Analysis



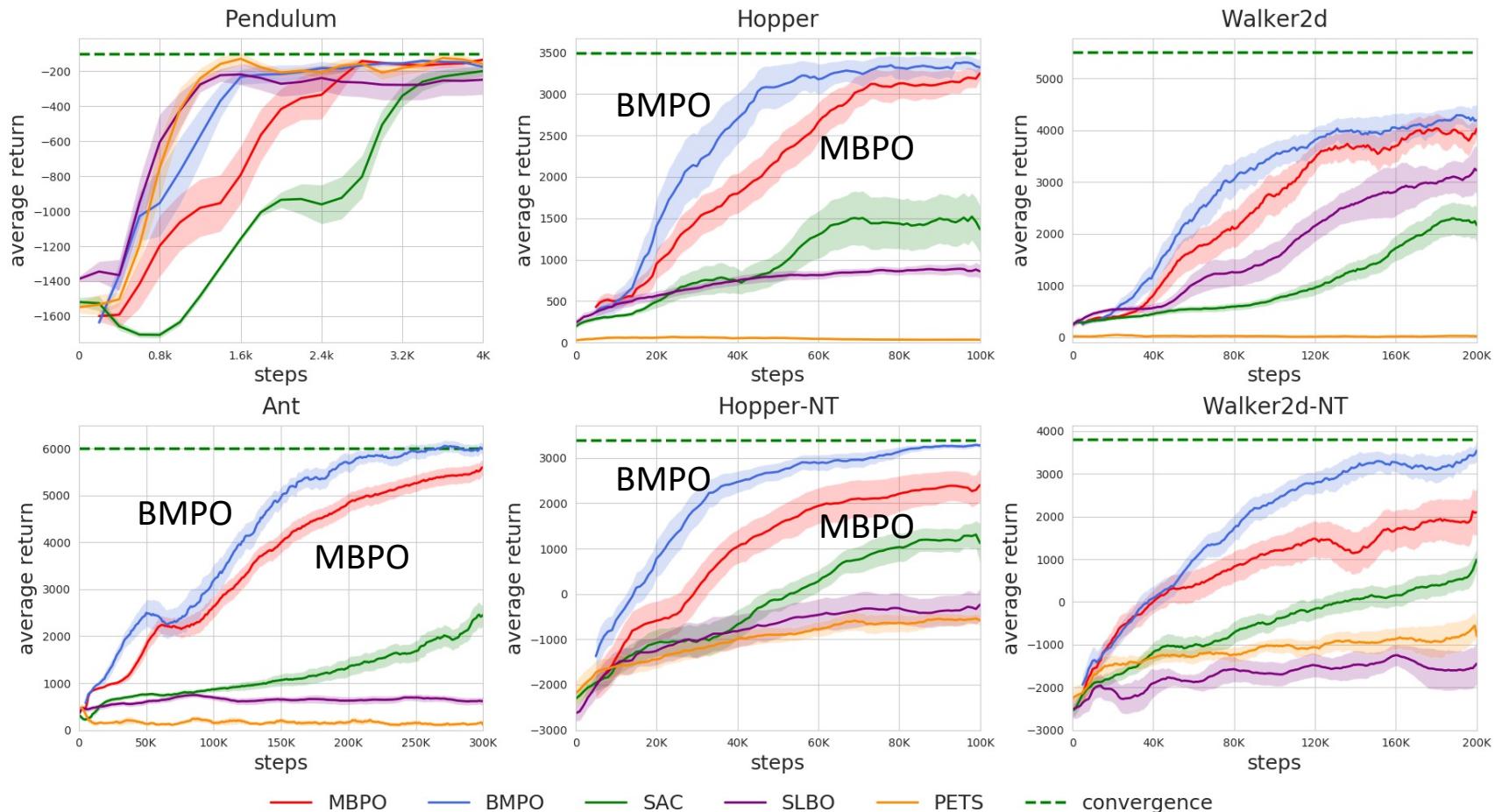
**Bidirectional model:**

$$|\eta[\pi] - \eta^{\text{branch}}[\pi]| \leq 2r_{\max} \left[ \frac{\gamma^{k_1+k_2+1} \epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k_1+k_2} \epsilon_\pi}{(1-\gamma)} + \frac{\max(k_1, k_2) \epsilon_m}{1-\gamma} \right].$$

**Forward model:**

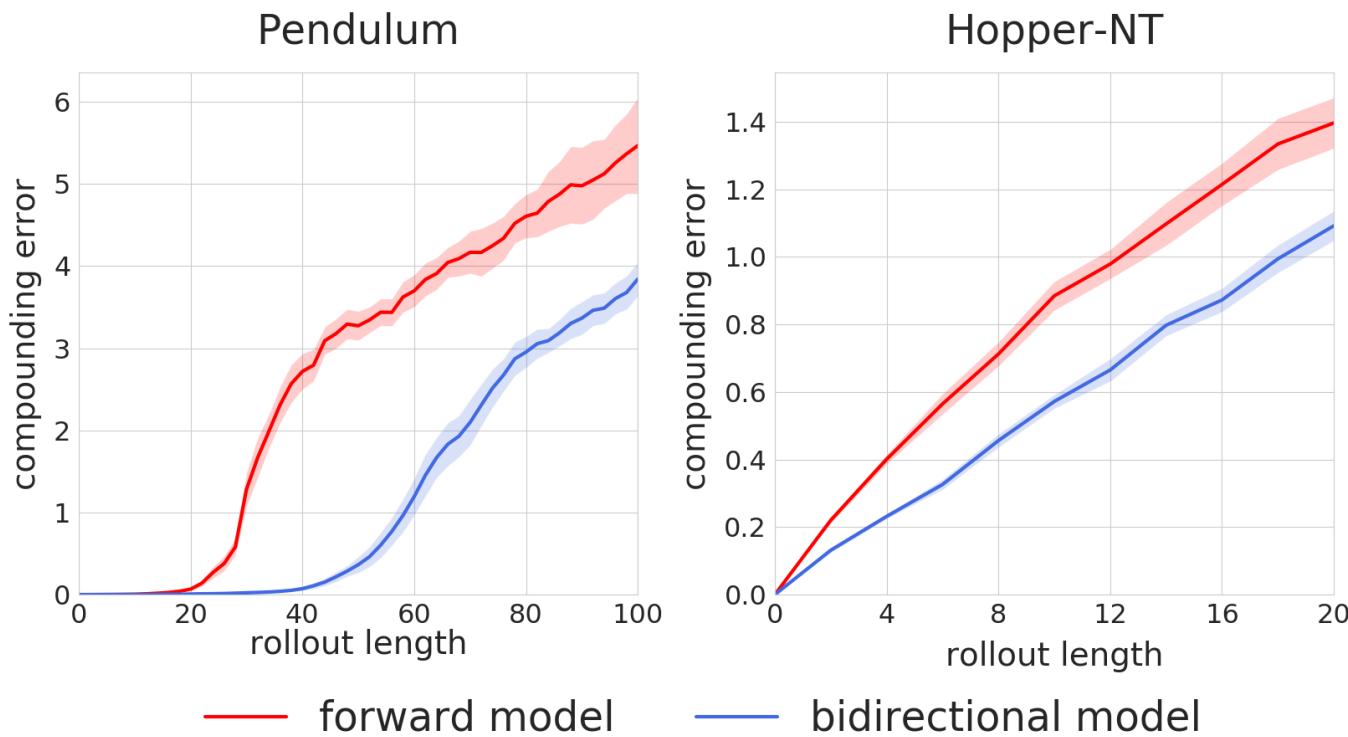
$$|\eta[\pi] - \eta^{\text{branch}}[\pi]| \leq 2r_{\max} \left[ \frac{\gamma^{k_1+k_2+1} \epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k_1+k_2} \epsilon_\pi}{(1-\gamma)} + \frac{(k_1+k_2) \epsilon_m}{1-\gamma} \right].$$

# Comparison with State-of-the-Arts



Compared with previous state-of-the-art baselines, BMPO (blue) learns faster and has better asymptotic performance than previous model-based algorithms using only the forward model.

# Model Compounding Error



$$\text{Error}_{\text{for}} = \frac{1}{2h} \sum_{i=1}^{2h} \|\hat{s}_i - s_i\|_2^2$$
$$\text{Error}_{\text{bi}} = \frac{1}{2h} \sum_{i=1}^h \left( \|\hat{s}_{h+i} - s_{h+i}\|_2^2 + \|\hat{s}_{h-i} - s_{h-i}\|_2^2 \right)$$

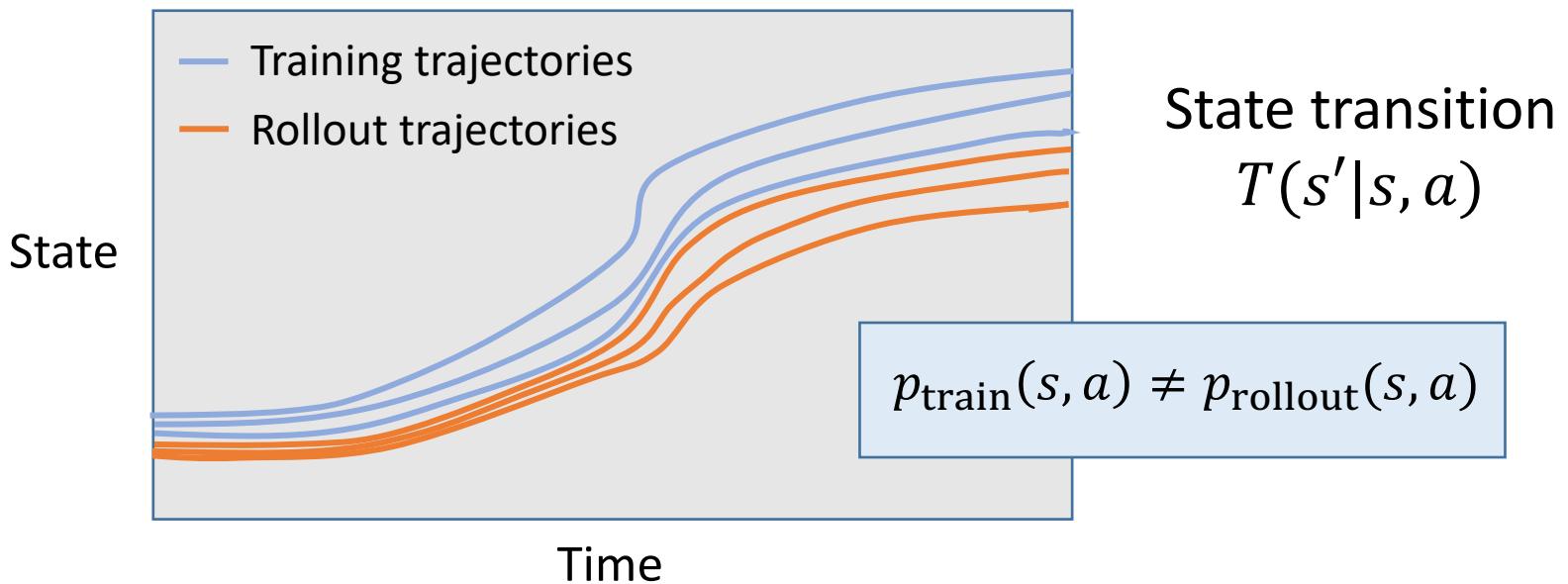
# Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: BMPO, [AMPO](#) and AutoMBPO

Appendix: Backpropagation through paths: SVG and MAAC

# Distribution Mismatch in MBRL

- One potential problem
  - Distribution mismatch between **real data (in model learning)** and **simulated data (in model usage)**
  - It's the source of compounding model error



# Deal with Distribution Mismatch

- In model learning
  - Design different architectures and loss functions
  - Make the rollouts more like real
- Asadi, Kavosh, et al. "Combating the Compounding-Error Problem with a Multi-step Model." arXiv preprint arXiv:1905.13320 (2019).  
Farahmand, Amir-massoud, Andre Barreto, and Daniel Nikovski. "Value-aware loss function for model-based reinforcement learning." Artificial Intelligence and Statistics. 2017.
- In model usage
  - Design careful rollout schemes
  - Stop the rollout before the generated data departure
- Janner, Michael, et al. "When to trust your model: Model-based policy optimization." *Advances in Neural Information Processing Systems*. 2019.  
Buckman, Jacob, et al. "Sample-efficient reinforcement learning with stochastic ensemble value expansion." *Advances in Neural Information Processing Systems*. 2018.
- Although alleviated, the problem still exists

# Notations

- Environment:  $T(s'|s, a)$ , model:  $\hat{T}(s'|s, a)$
- Occupancy measure (normalized)

$$\rho_T^\pi(s, a) = (1 - \gamma) \cdot \pi(a|s) \sum_t \gamma^t P_{T,t}^\pi(s)$$

- State visit distribution

$$\nu_T^\pi(s) = (1 - \gamma) \sum_t \gamma^t P_{T,t}^\pi(s)$$

- Integral probability metric

$$d_{\mathcal{F}}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim \mathbb{P}}[f(x)] - \mathbb{E}_{x \sim \mathbb{Q}}[f(x)]$$

- IPM measures to many well-known distances
  - Wasserstein-1 distance, Maximum Mean Discrepancy

# A Lower Bound for Expected Return

**Theorem 3.1.** Let  $R := \sup_{s,a} r(s,a) < \infty$ ,  $\mathcal{F} := \cup_{s' \in \mathcal{S}} \mathcal{F}_{s'}$  and define  $\epsilon_\pi := 2d_{\text{TV}}(\nu_T^\pi, \nu_T^{\pi_D})$ . Under the assumption of Lemma 3.1, the expected return  $\eta[\pi]$  admits the following bound:

$$\eta[\pi] \geq \hat{\eta}[\pi] - R \cdot \epsilon_\pi - \gamma R \cdot d_{\mathcal{F}}(\rho_T^{\pi_D}, \rho_{\hat{T}}^\pi) \cdot \text{Vol}(\mathcal{S}) \\ - \gamma R \cdot \mathbb{E}_{(s,a) \sim \rho_T^{\pi_D}} \sqrt{2D_{\text{KL}}(T(\cdot|s,a) \parallel \hat{T}(\cdot|s,a))},$$

where  $\text{Vol}(\mathcal{S})$  is the volume of state space  $\mathcal{S}$ .

policy learning in model + model learning = basic MBRL

# A Lower Bound for Expected Return

**Theorem 3.1.** Let  $R := \sup_{s,a} r(s,a) < \infty$ ,  $\mathcal{F} := \cup_{s' \in \mathcal{S}} \mathcal{F}_{s'}$  and define  $\epsilon_\pi := 2d_{\text{TV}}(\nu_T^\pi, \nu_T^{\pi_D})$ . Under the assumption of Lemma 3.1, the expected return  $\eta[\pi]$  admits the following bound:

$$\eta[\pi] \geq \hat{\eta}[\pi] - \boxed{R \cdot \epsilon_\pi} - \gamma R \cdot \boxed{d_{\mathcal{F}}(\rho_T^{\pi_D}, \rho_{\hat{T}}^\pi)} \cdot \text{Vol}(\mathcal{S}) \\ - \gamma R \cdot \mathbb{E}_{(s,a) \sim \rho_T^{\pi_D}} \sqrt{2D_{\text{KL}}(T(\cdot|s,a) \parallel \hat{T}(\cdot|s,a))},$$

where  $\text{Vol}(\mathcal{S})$  is the volume of state space  $\mathcal{S}$ .

reliable exploitation in  
batch RL

distribution distance

(violate the rule of exploration)

# Review: Domain Adaptation

**Theorem 1.** Let  $\mu_s, \mu_t \in \mathcal{P}(\mathcal{X})$  be two probability measures. Assume the hypotheses  $h \in H$  are all  $K$ -Lipschitz continuous for some  $K$ . Then, for every  $h \in H$  the following holds

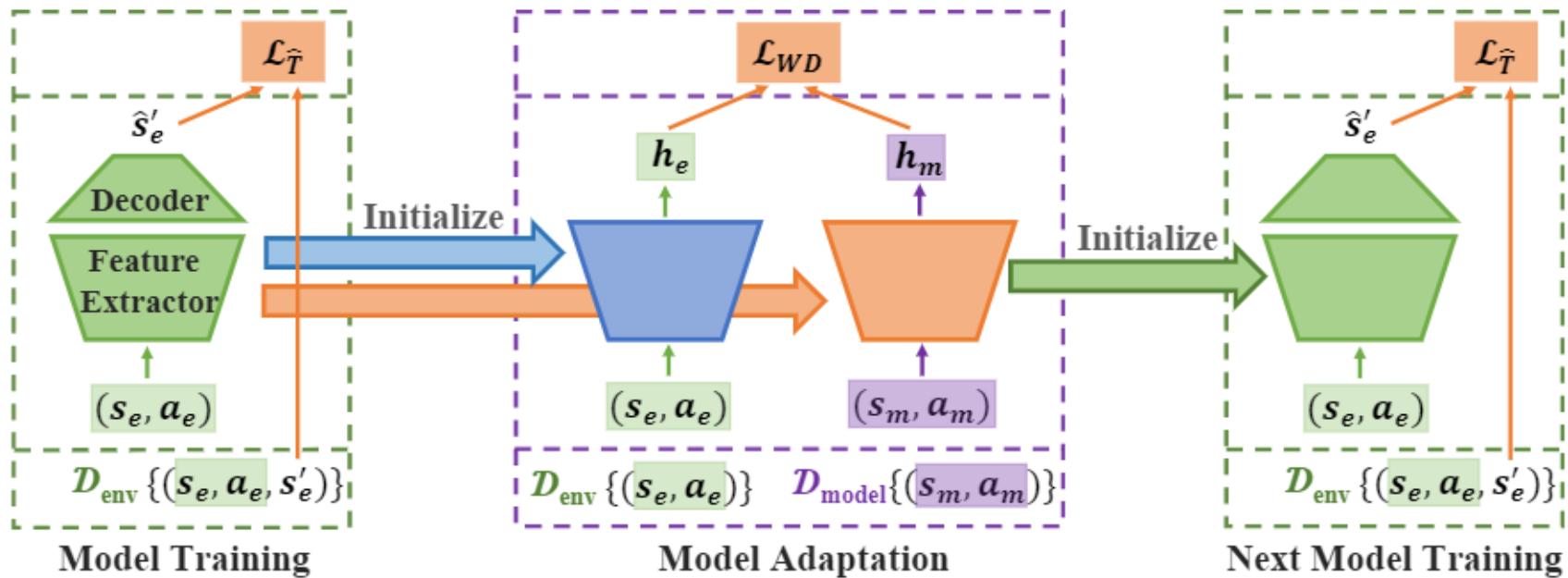
$$\epsilon_t(h) \leq \epsilon_s(h) + 2KW_1(\mu_s, \mu_t) + \lambda \quad (11)$$

error in target domain      error in source domain      Wasserstein distance      Constant

Inspiration: aligning the two feature distributions in MBRL

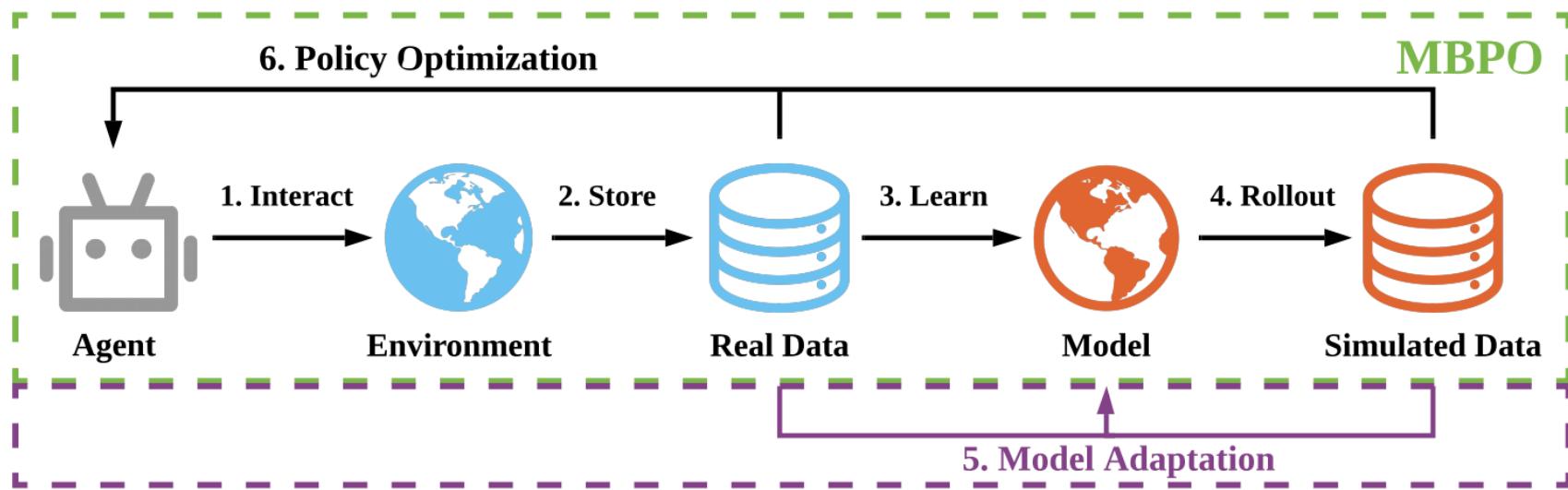
# Unsupervised Model Adaptation

- Source domain: model training data
- Target domain: model rollout data
- Aligning the latent feature distributions by minimizing IPMs according to the lower bound



# AMPO

- Adaptation augmented Model-based Policy Optimization



- AMPO: Wasserstein-1 distance (WGAN)
- Variants: use other distribution divergence, e.g. MMD

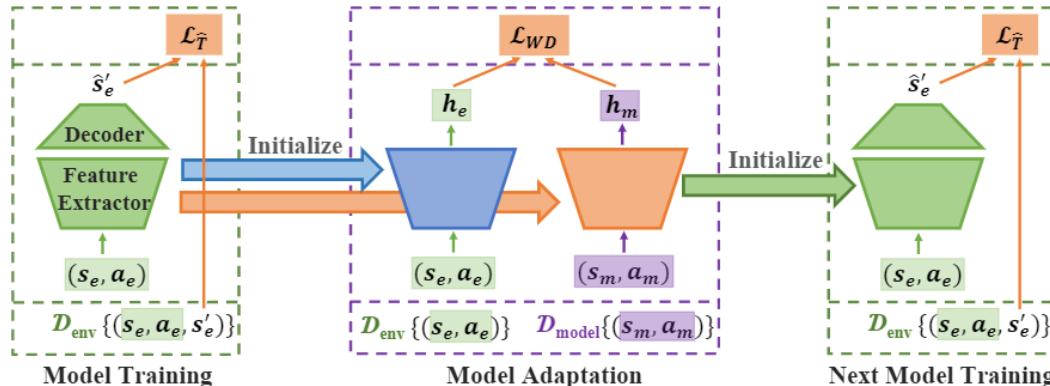
# Overall Algorithm

---

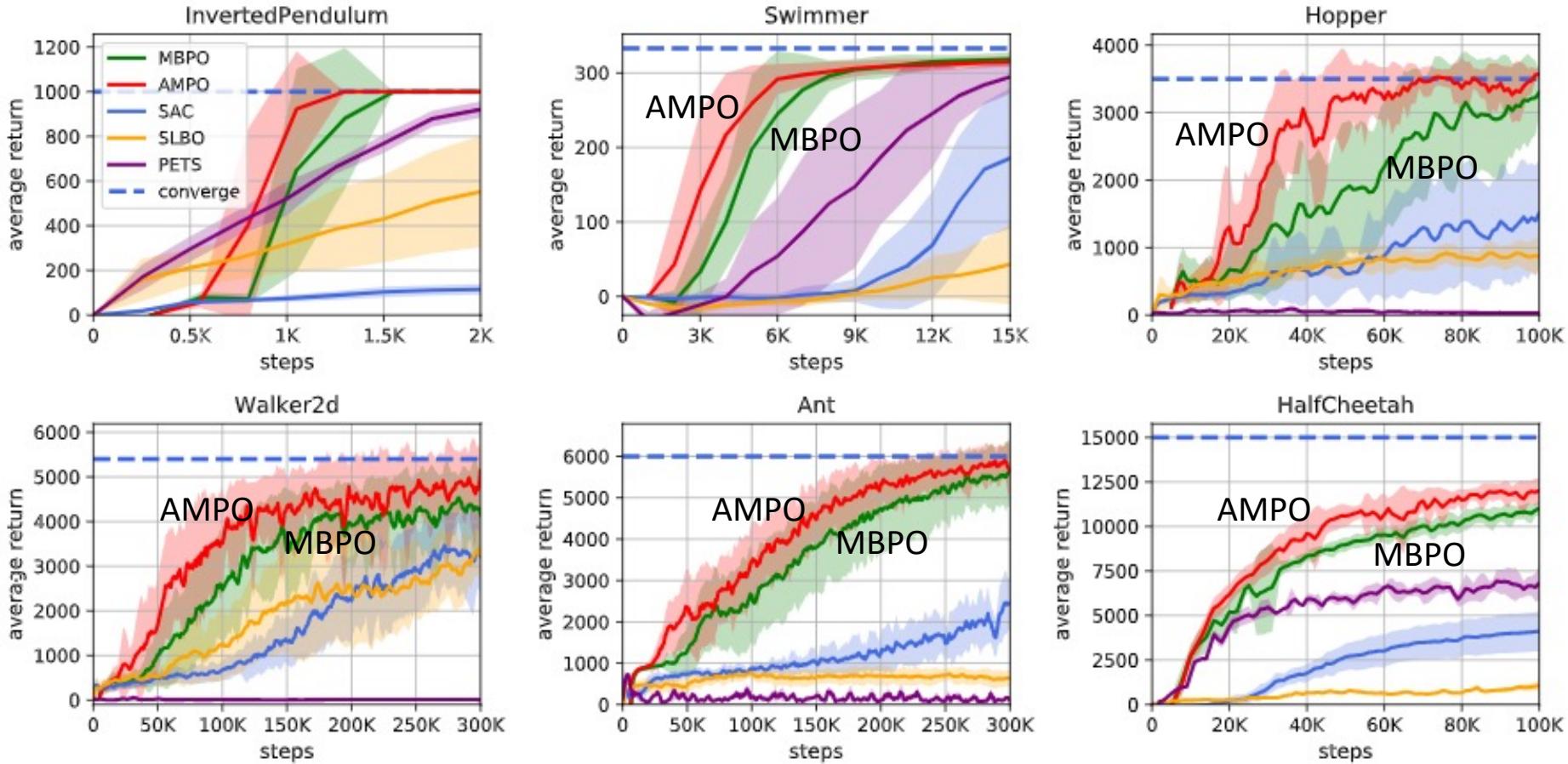
**Algorithm 1** AMPO

---

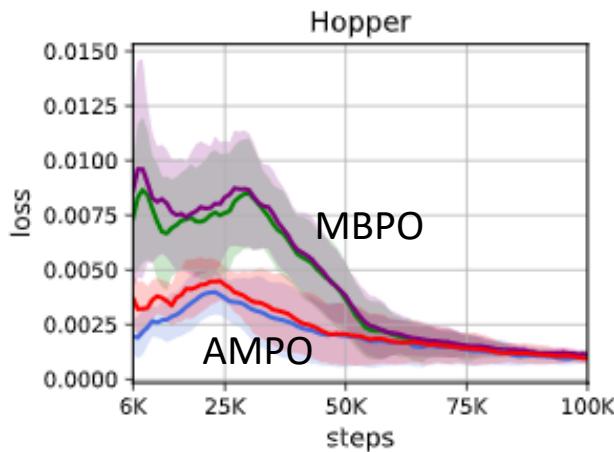
- 1: Initialize policy  $\pi_\phi$ , dynamics model  $\hat{T}_\theta$ , environment buffer  $\mathcal{D}_e$ , model buffer  $\mathcal{D}_m$
- 2: **repeat**
- 3:   Take an action in the environment using the policy  $\pi_\phi$ ; add the sample  $(s, a, s', r)$  to  $\mathcal{D}_e$
- 4:   **if** every  $E$  real timesteps are finished **then**
- 5:     Perform  $G_1$  gradient steps to train the model  $\hat{T}_\theta$  with samples from  $\mathcal{D}_e$
- 6:     **for**  $F$  model rollouts **do**
- 7:       Sample a state  $s$  uniformly from  $\mathcal{D}_e$
- 8:       Use policy  $\pi_\phi$  to perform a  $k$ -step model rollout starting from  $s$ ; add to  $\mathcal{D}_m$
- 9:     **end for**
- 10:    Perform  $G_2$  gradient steps to train the feature extractor with samples  $(s, a)$  from both  $\mathcal{D}_e$  and  $\mathcal{D}_m$  by the model adaptation loss  $\mathcal{L}_{WD}$
- 11:   **end if**
- 12:   Perform  $G_3$  gradient steps to train the policy  $\pi_\phi$  with samples  $(s, a, s', r)$  from  $\mathcal{D}_e \cup \mathcal{D}_m$
- 13: **until** certain number of real samples



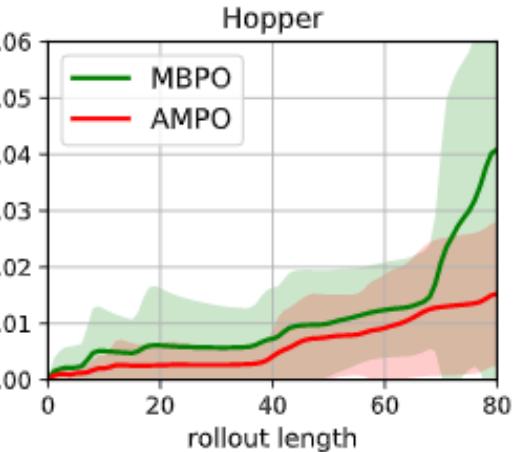
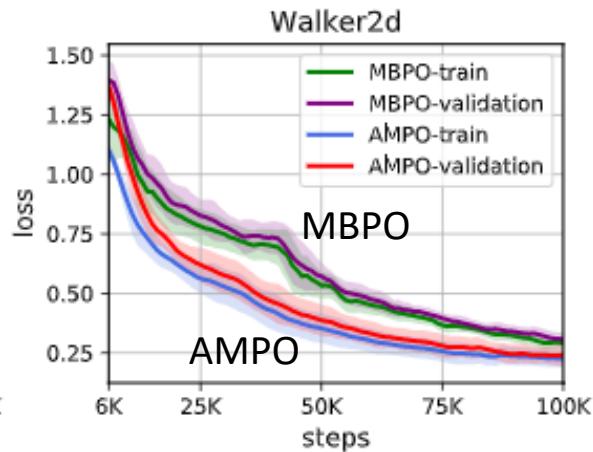
# Comparison with State-of-the-Arts



# Model Loss Evaluation



(a) One-step model losses.



(b) Compounding errors.

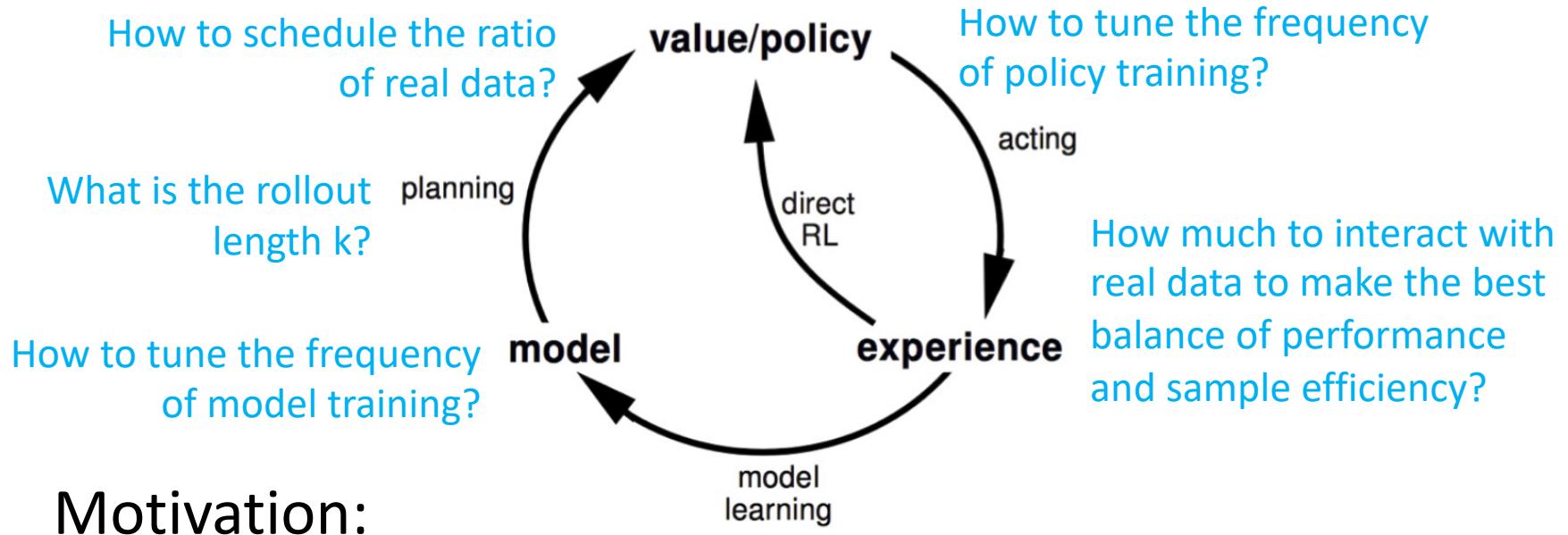
- Model adaptation makes the model more accurate
- AMPO achieves smaller compounding errors than MBPO

# Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: BMPO, AMPO and [AutoMBPO](#)

Appendix: Backpropagation through paths: SVG and MAAC

# AutoMBPO: better understanding of MBRL hyper-parameters



## Motivation:

- MBRL methods are sensitive to the primary hyper-parameters, e.g., ratio of real data and simulated data, model and policy training frequency, rollout length.
- To provide a better understanding of MBRL through hyper-parameter scheduling.

# Analysis of real ratio schedule

Return discrepancy upper bound:

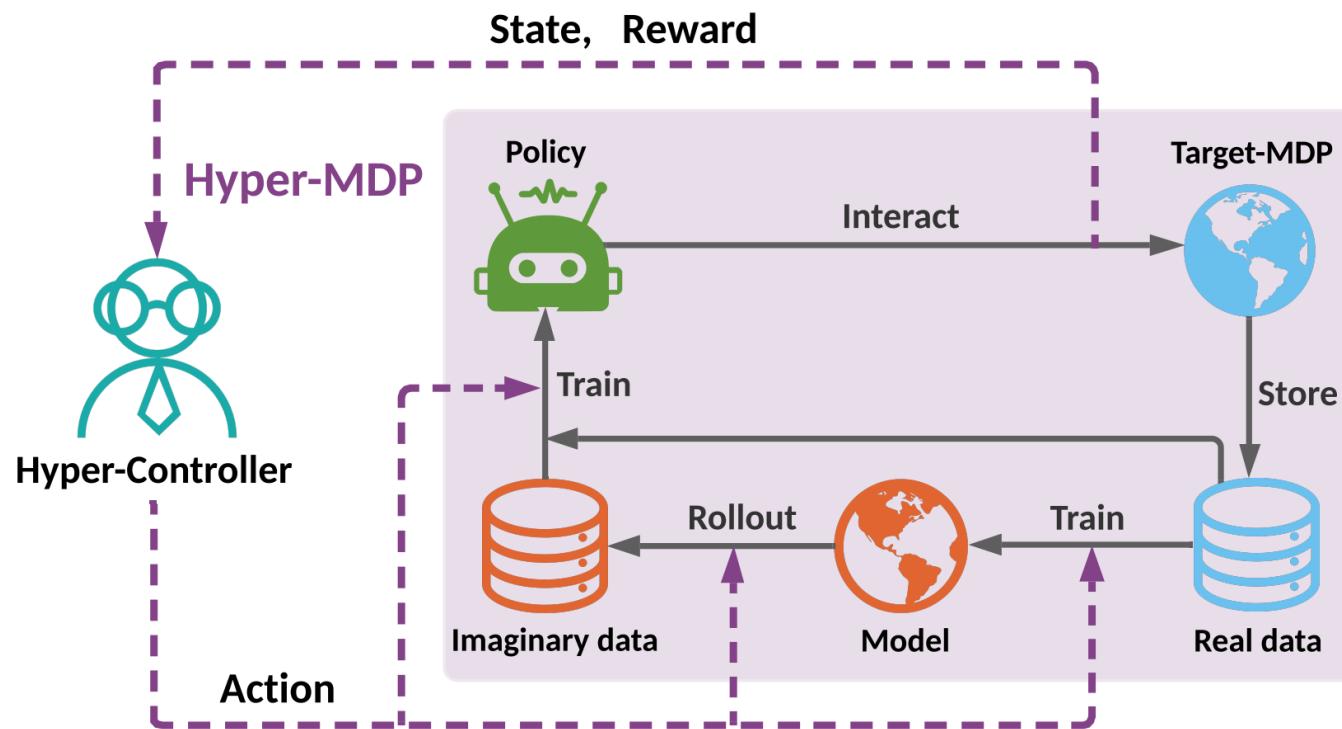
$$\|V^* - V^{\pi_K}\|_{p,\rho} \leq \frac{2\gamma}{(1-\gamma)^2} C_{\rho,\mu}^{1/p} d_{p,\mu}(B\mathcal{F}, \mathcal{F}) + O\left(\left(\frac{\beta|\mathcal{A}|}{N_{\text{real}}}\left(\log\left(\frac{N_{\text{real}}}{\beta|\mathcal{A}|}\right) + \log\left(\frac{K}{\delta}\right)\right)\right)^{\frac{1}{2p}}\right) + O\left(\Phi^{-1}\left(1 - \frac{\beta\delta}{8KN_{\text{real}}(1-\beta)}\right)\sigma\right) + O\left(\gamma^{K/p} V_{\max}\right).$$

Real Ratio ←  $\beta|\mathcal{A}|$   
Real data number ←  $N_{\text{real}}$

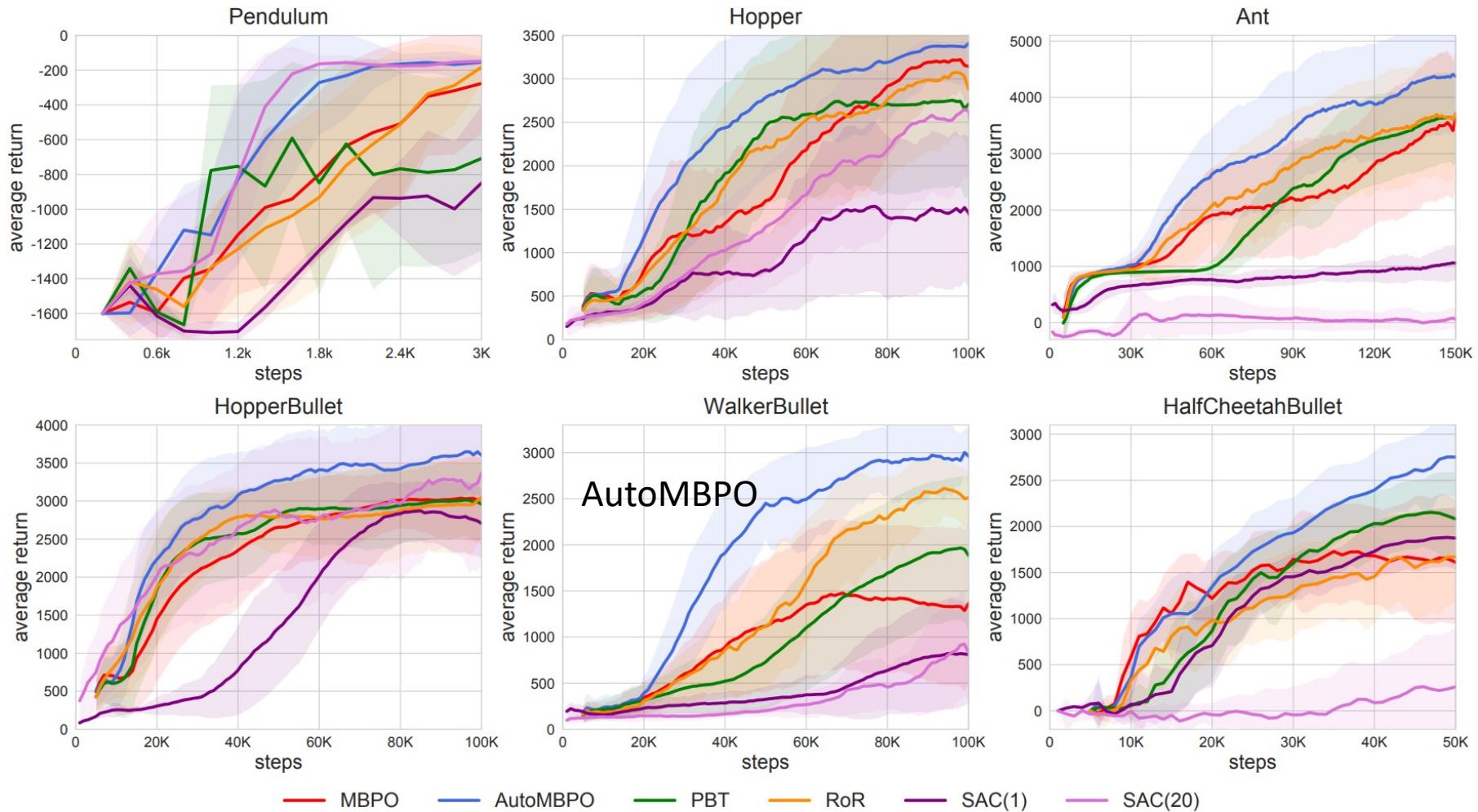
As  $\beta/N_{\text{real}}$  increases, the second term increases while the third term decreases. So there exists an optimal value for  $\beta/N_{\text{real}}$ . Since  $N_{\text{real}}$  increases during training, gradually increasing  $\beta$  is promising to achieve good performance.

# AutoMBPO Framework

Idea: formulate hyperparameter scheduling as an MDP and then adopt some RL algorithm (e.g., PPO in the paper) to solve it.

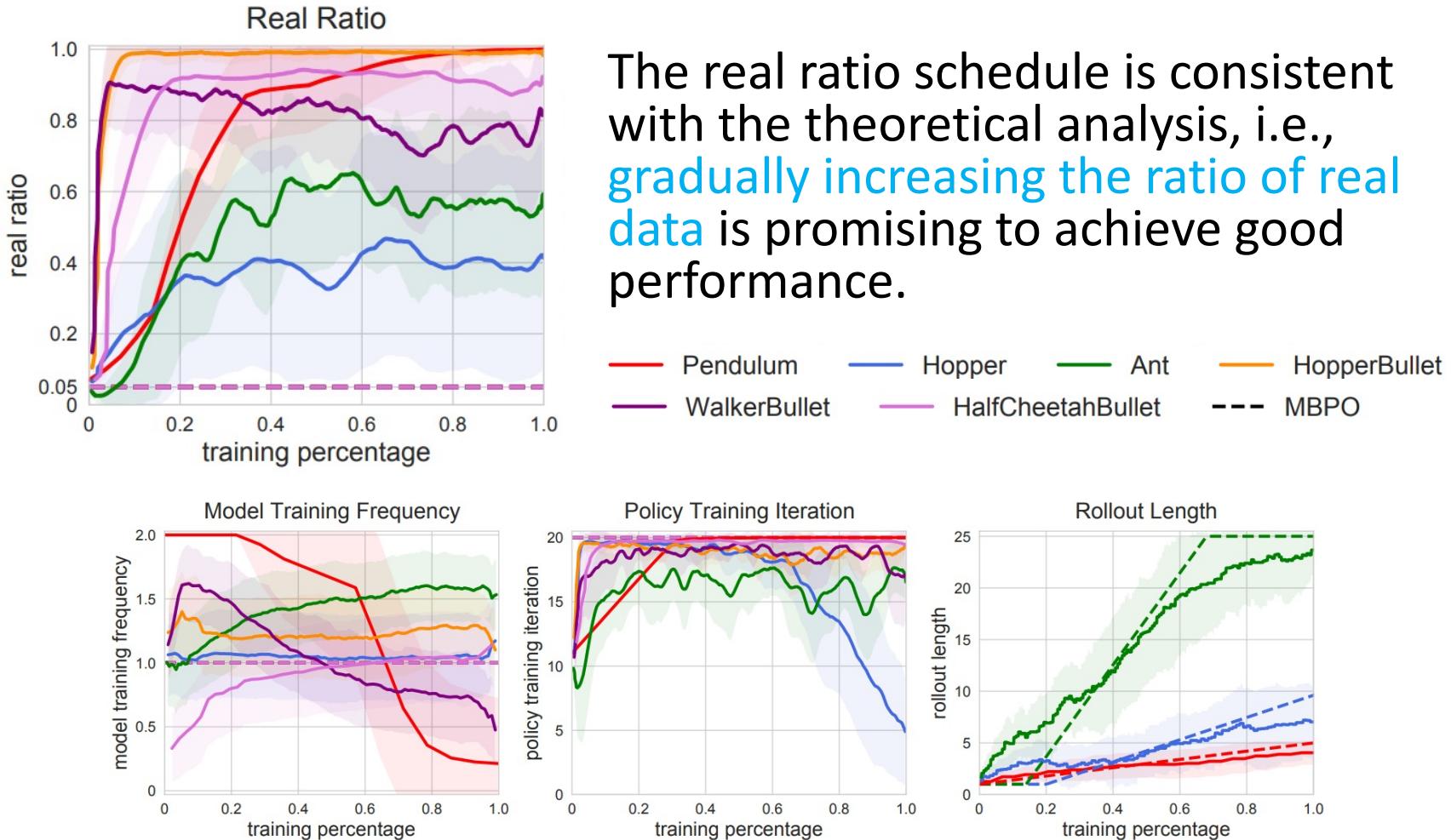


# AutoMBPO Experiments



# AutoMBPO Experiments

## Hyperparameter Schedule Visualization:

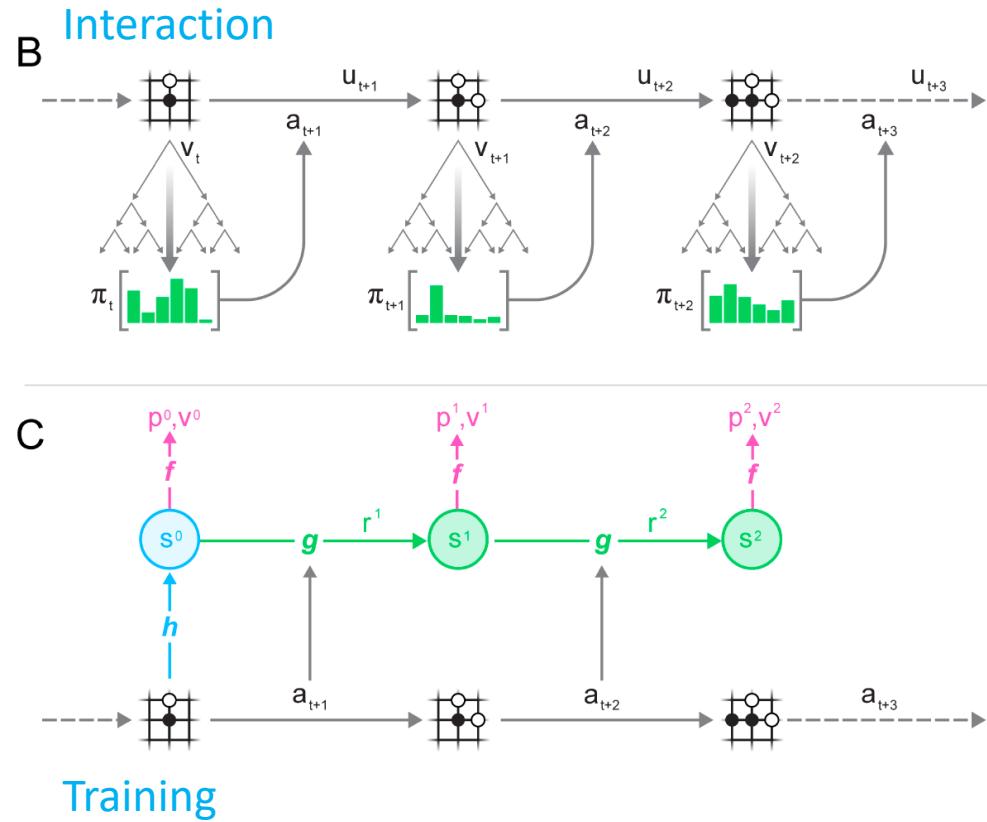
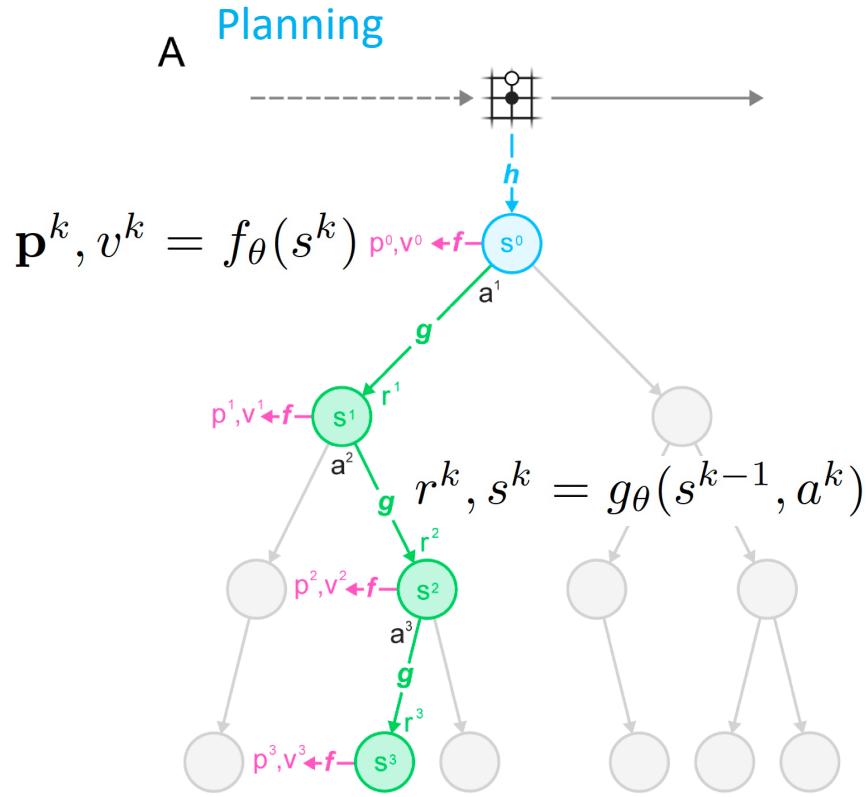


The real ratio schedule is consistent with the theoretical analysis, i.e., **gradually increasing the ratio of real data** is promising to achieve good performance.

More to investigation

# MuZero (and AlphaZero)

Learn a model  $(h, f, g)$  to perform planning and yield an action

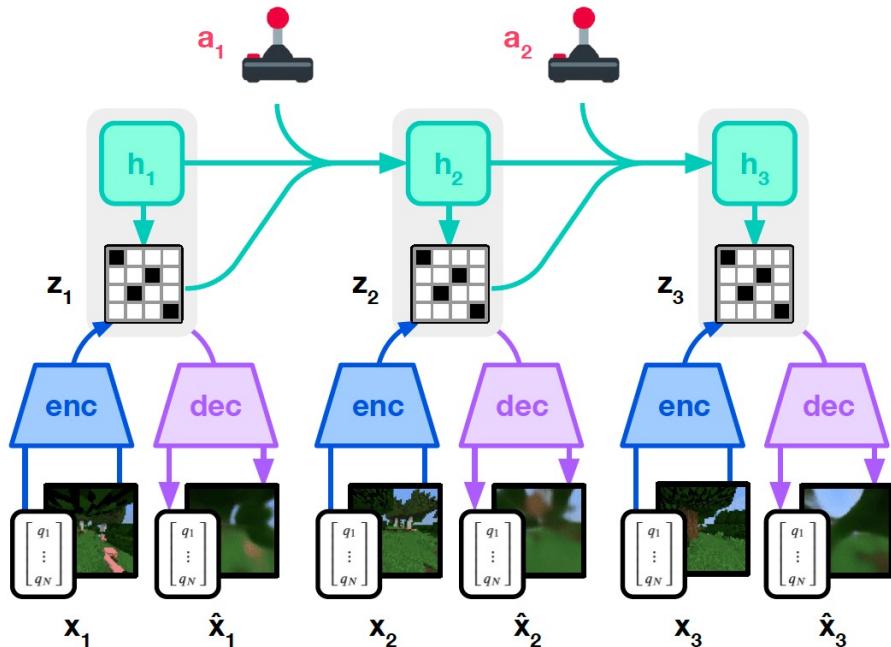


Training loss  $l = (z - v)^2 - \pi^\top \log p + c\|\theta\|^2,$

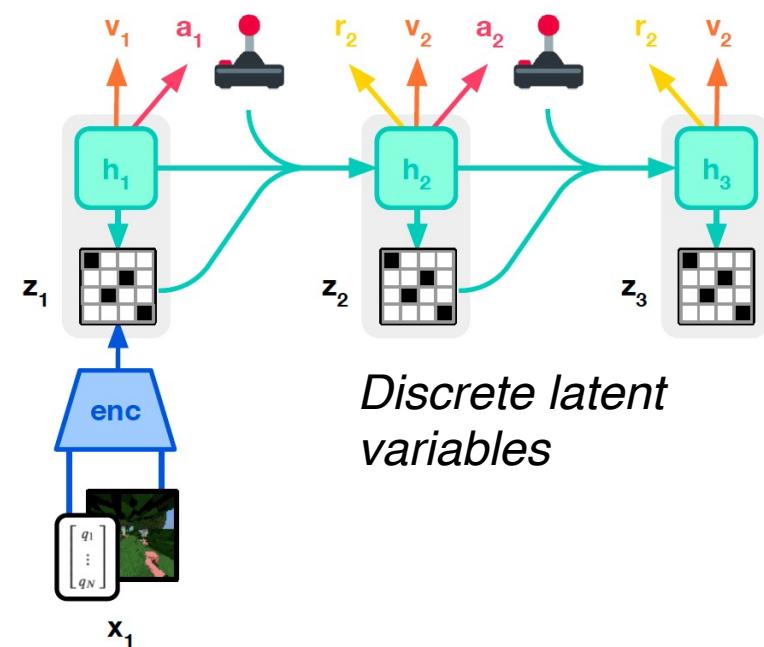
More to investigation

# DreamerV3 (and Dreamer Series)

DreamerV3 learns a world model from experience for rich perception and imagination training.



(a) World Model Learning



*Discrete latent variables*

(b) Actor Critic Learning

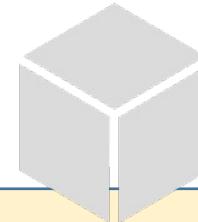
The world model encodes sensory inputs into a discrete representation  $z_t$  that is predicted by a sequence model with recurrent state  $h_t$  given actions  $a_t$ .

# Summary of Model-based RL



## Model as a Blackbox

- Sample experience data and train the policy in the manner of model-free RL
- Seamless to policy training algorithms
- Easy for rollout planning
- The simulation data efficiency may still be low
- E.g., Dyna-Q, MPC, MBPO



## Model as a Whitebox

- Environment model, including transition dynamics and reward function, is a differentiable stochastic mapping
- Offer both data and gradient guidance for value and policy
- High data efficiency
- E.g., MAAC, SVG, PILCO

# Future Directions of MBRL

- Environment model learning
  - Learning objectives
  - Environment imitation
  - Complex simulation
  - MBRL with true model in simulation
- Better understanding of bounds
  - When to have tighter bounds
  - Rollout length and when to rollout
- Multi-agent MBRL
  - An environment of multi-agent dynamics
  - Sampling the joint action sequences with the model and joint policy

# Thank You! Questions?

Hang Lai, Weinan Zhang et al. Bidirectional Model-based Policy Optimization. ICML 2020.  
<https://arxiv.org/abs/2007.01995>

Jian Shen, Weinan Zhang et al. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020. <https://arxiv.org/abs/2010.09546>

Hang Lai, Weinan Zhang et al. On Effective Scheduling of Model-based Reinforcement Learning. NeurIPS 2021. <https://arxiv.org/abs/2111.08550>

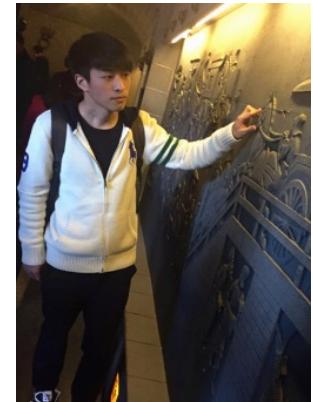
Fan-ming Luo et al. A survey on model-based reinforcement learning. SCIS 2024.  
<https://arxiv.org/abs/2206.09328>



上海交通大学  
SHANGHAI JIAO TONG UNIVERSITY



Hang Lai



Jian Shen

Weinan Zhang Associate Professor, Shanghai Jiao Tong University  
<http://wnzhang.net>

# Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS & PETS
3. Branched rollout method: MBPO
4. Recent work: BMPO & AMPO

Appendix: Backpropagation through paths: SVG and MAAC

# Deterministic Policy Gradient

- A critic module for state-action value estimation

$$Q^w(s, a) \simeq Q^\pi(s, a)$$

$$L(w) = \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} [(Q^w(s, a) - Q^\pi(s, a))^2]$$

- With the differentiable critic, the deterministic continuous-action actor can be updated as
  - Deterministic policy gradient theorem

$$J(\pi_\theta) = \mathbb{E}_{s \sim \rho^\pi} [Q^\pi(s, a)]$$

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim \rho^\pi} [\nabla_\theta \pi_\theta(s) \nabla_a Q^\pi(s, a)|_{a=\pi_\theta(s)}]$$

On-policy

Chain rule

# From Deterministic to Stochastic

- For deterministic environment, i.e., reward and transition, and policy, i.e., a function mapping state to action

$$\mathbf{a} = \pi(\mathbf{s}; \theta) \quad \mathbf{s}' = \mathbf{f}(\mathbf{s}, \mathbf{a})$$

$$V(\mathbf{s}) = r(\mathbf{s}, \mathbf{a}) + \gamma V'(\mathbf{f}(\mathbf{s}, \mathbf{a}))$$

$$g_x \triangleq \partial g(x, y) / \partial x$$

$$V_{\mathbf{s}} = r_{\mathbf{s}} + r_{\mathbf{a}} \pi_{\mathbf{s}} + \gamma V'_{\mathbf{s}'} (\mathbf{f}_{\mathbf{s}} + \mathbf{f}_{\mathbf{a}} \pi_{\mathbf{s}}),$$

$$V_{\theta} = r_{\mathbf{a}} \pi_{\theta} + \gamma V'_{\mathbf{s}'} \mathbf{f}_{\mathbf{a}} \pi_{\theta} + \gamma V'_{\theta}.$$

# From Deterministic to Stochastic

- Math: reparameterization of distributions
  - Conditional Gaussian distribution

$$p(y|x) = \mathcal{N}(y|\mu(x), \sigma^2(x))$$

$$\Rightarrow y = \mu(x) + \sigma(x)\xi, \text{ where } \xi \sim \mathcal{N}(0, 1)$$

- Consider conditional densities whose samples are generated by a **deterministic** function of an input noise variable

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \xi), \text{ where } \xi \sim \rho(\cdot)$$

$$\mathbb{E}_{p(\mathbf{y}|\mathbf{x})}\mathbf{g}(\mathbf{y}) = \int \mathbf{g}(\mathbf{f}(\mathbf{x}, \xi))\rho(\xi)d\xi$$

$$g_x \triangleq \partial g(x, y)/\partial x$$

**derivative**  $\nabla_{\mathbf{x}}\mathbb{E}_{p(\mathbf{y}|\mathbf{x})}\mathbf{g}(\mathbf{y}) = \mathbb{E}_{\rho(\xi)}\mathbf{g}_y \mathbf{f}_x \approx \frac{1}{M} \sum_{i=1}^M \mathbf{g}_y \mathbf{f}_x|_{\xi=\xi_i}$

# From Deterministic to Stochastic

- Deterministic version for reference

$$\mathbf{a} = \pi(\mathbf{s}; \theta) \quad \mathbf{s}' = \mathbf{f}(\mathbf{s}, \mathbf{a})$$

$$V(\mathbf{s}) = r(\mathbf{s}, \mathbf{a}) + \gamma V'(\mathbf{f}(\mathbf{s}, \mathbf{a}))$$

- Stochastic environment, policy, value and gradients

$$V_{\mathbf{s}} = r_{\mathbf{s}} + r_{\mathbf{a}}\pi_{\mathbf{s}} + \gamma V'_{\mathbf{s}'}(\mathbf{f}_{\mathbf{s}} + \mathbf{f}_{\mathbf{a}}\pi_{\mathbf{s}}),$$

$$V_{\theta} = r_{\mathbf{a}}\pi_{\theta} + \gamma V'_{\mathbf{s}'}\mathbf{f}_{\mathbf{a}}\pi_{\theta} + \gamma V'_{\theta}.$$

$$\mathbf{a} = \pi(\mathbf{s}, \eta; \theta) \quad \mathbf{s}' = \mathbf{f}(\mathbf{s}, \mathbf{a}, \xi) \quad \eta \sim \rho(\eta) \text{ and } \xi \sim \rho(\xi)$$

$$V(\mathbf{s}) = \mathbb{E}_{\rho(\eta)} \left[ r(\mathbf{s}, \pi(\mathbf{s}, \eta; \theta)) + \gamma \mathbb{E}_{\rho(\xi)} [V'(\mathbf{f}(\mathbf{s}, \pi(\mathbf{s}, \eta; \theta), \xi))] \right]$$

$$V_{\mathbf{s}} = \mathbb{E}_{\rho(\eta)} \left[ r_{\mathbf{s}} + r_{\mathbf{a}}\pi_{\mathbf{s}} + \gamma \mathbb{E}_{\rho(\xi)} V'_{\mathbf{s}'}(\mathbf{f}_{\mathbf{s}} + \mathbf{f}_{\mathbf{a}}\pi_{\mathbf{s}}) \right],$$

$$V_{\theta} = \mathbb{E}_{\rho(\eta)} \left[ r_{\mathbf{a}}\pi_{\theta} + \gamma \mathbb{E}_{\rho(\xi)} [V'_{\mathbf{s}'}\mathbf{f}_{\mathbf{a}}\pi_{\theta} + V'_{\theta}] \right].$$

$$g_x \triangleq \partial g(x, y) / \partial x$$

# SVG Algorithm and Experiments

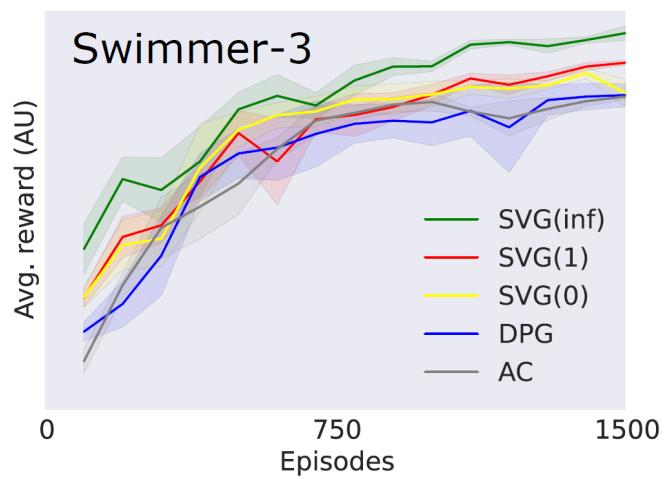
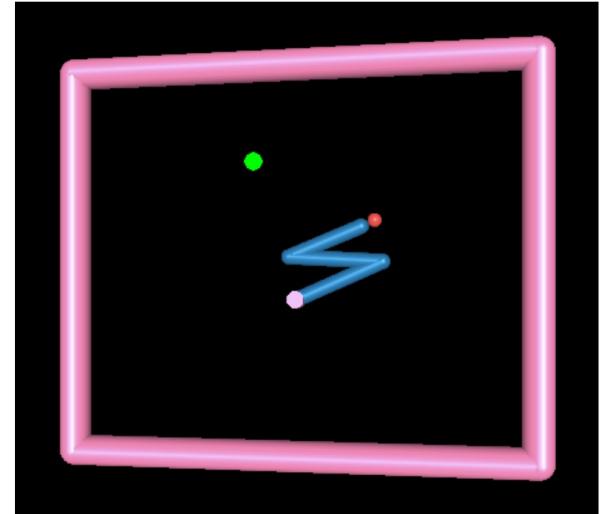
---

## Algorithm 1 SVG( $\infty$ )

---

```
1: Given empty experience database  $\mathcal{D}$ 
2: for trajectory = 0 to  $\infty$  do
3:   for  $t = 0$  to  $T$  do
4:     Apply control  $\mathbf{a} = \pi(\mathbf{s}, \eta; \theta)$ ,  $\eta \sim \rho(\eta)$ 
5:     Insert  $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$  into  $\mathcal{D}$ 
6:   end for
7:   Train generative model  $\hat{\mathbf{f}}$  using  $\mathcal{D}$ 
8:    $v'_{\mathbf{s}} = 0$  (finite-horizon)
9:    $v'_{\theta} = 0$  (finite-horizon)
10:  for  $t = T$  down to 0 do
11:    Infer  $\xi|(\mathbf{s}, \mathbf{a}, \mathbf{s}')$  and  $\eta|(\mathbf{s}, \mathbf{a})$ 
12:     $v_{\theta} = [r_{\mathbf{a}}\pi_{\theta} + \gamma(v'_{\mathbf{s}'}\hat{\mathbf{f}}_{\mathbf{a}}\pi_{\theta} + v'_{\theta})]|_{\eta, \xi}$ 
13:     $v_{\mathbf{s}} = [r_{\mathbf{s}} + r_{\mathbf{a}}\pi_{\mathbf{s}} + \gamma v'_{\mathbf{s}'}(\hat{\mathbf{f}}_{\mathbf{s}} + \hat{\mathbf{f}}_{\mathbf{a}}\pi_{\mathbf{s}})]|_{\eta, \xi}$ 
14:  end for
15:  Apply gradient-based update using  $v_{\theta}^0$ 
16: end for
```

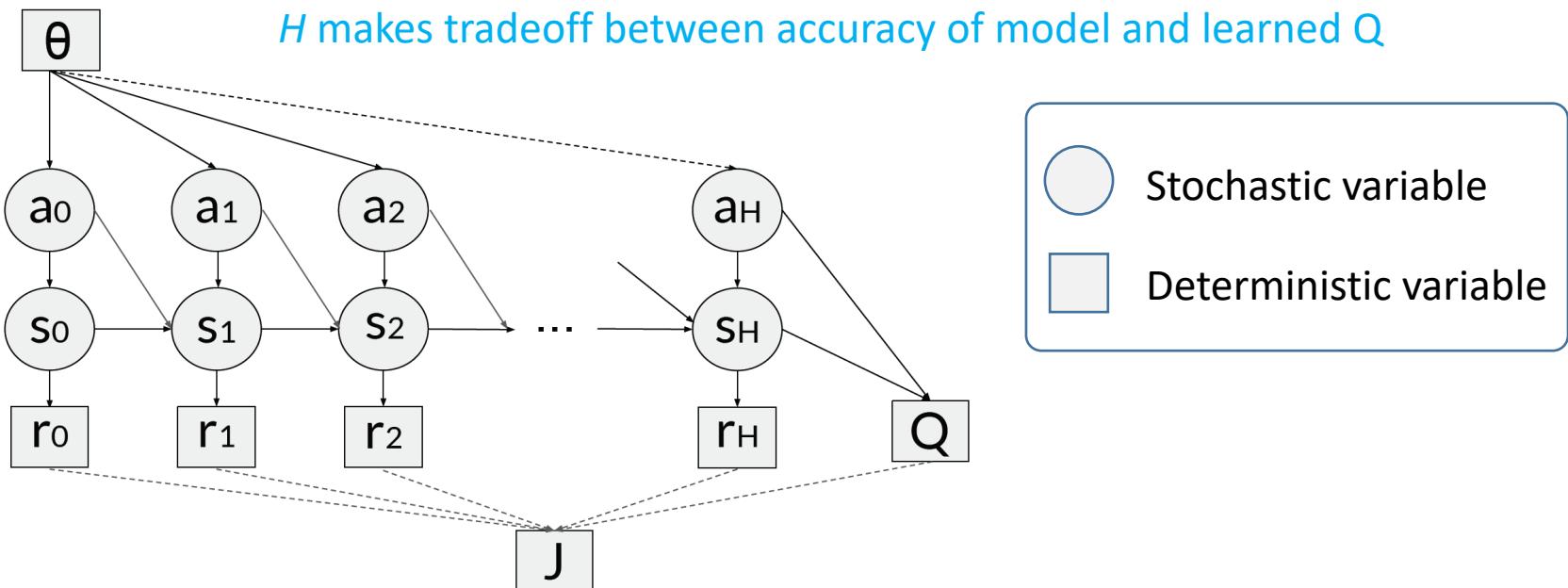
---



# Model-Augmented Actor Critic

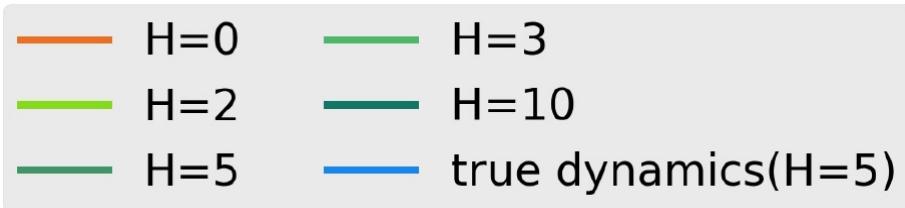
- The objective function can be directly built as a stochastic computation graph

$$J_\pi(\theta) = \mathbb{E} \left[ \sum_{t=0}^{H-1} \gamma^t r(s_t) + \gamma^H \hat{Q}(s_H, a_H) \right] \quad \begin{aligned} s_{t+1} &\sim \hat{f}(s_t, a_t) \\ a_t &\sim \pi_\theta(s_t) \end{aligned}$$



# Theoretic Bounds

Large  $H$ , i.e., long rollout, brings large gradient error, thus the model error



**Lemma 4.1** (Gradient Error). *Let  $\hat{f}$  and  $\hat{Q}$  be the learned approximation of the dynamics  $f$  and  $Q$ -function  $Q$ , respectively. Assume that  $Q$  and  $\hat{Q}$  have  $L_q/2$ -Lipschitz continuous gradient and  $f$  and  $\hat{f}$  have  $L_f/2$ -Lipschitz continuous gradient. Let  $\epsilon_f = \max_t \|\nabla \hat{f}(\hat{s}_t, \hat{a}_t) - \nabla f(s_t, a_t)\|_2$  be the error on the model derivatives and  $\epsilon_Q = \|\nabla \hat{Q}(\hat{s}_H, \hat{a}_H) - \nabla Q(s_H, a_H)\|_2$  the error on the  $Q$ -function derivative. Then the error on the gradient between the learned objective and the true objective can be bounded by:*

$$\mathbb{E} \left[ \|\nabla_{\theta} J_{\pi} - \nabla_{\theta} \hat{J}_{\pi}\|_2 \right] \leq c_1(H) \epsilon_f + c_2(H) \epsilon_Q$$

# Theoretic Bounds

## One-step gradient approximation error

**Lemma 4.2** (Total Variation Bound). *Under the assumptions of the Lemma 4.1, let  $\theta = \theta_o + \alpha \nabla_{\theta} J_{\pi}$  be the parameters resulting from taking a gradient step on the exact objective, and  $\hat{\theta} = \theta_o + \alpha \nabla_{\theta} \hat{J}_{\pi}$  the parameters resulting from taking a gradient step on approximated objective, where  $\alpha \in \mathbb{R}^+$ . Then the following bound on the total variation distance holds*

$$\max_s D_{TV}(\pi_{\theta} || \pi_{\hat{\theta}}) \leq \alpha c_3 (\epsilon_f c_1(H) + \epsilon_Q c_2(H))$$

## Policy value lower bound

**Theorem 4.1** (Monotonic Improvement). *Under the assumptions of the Lemma 4.1, be  $\theta'$  and  $\hat{\theta}$  as defined in Lemma 4.2, and assuming that the reward is bounded by  $r_{\max}$ . Then the average return of the  $\pi_{\hat{\theta}}$  satisfies*

$$J_{\pi}(\hat{\theta}) \geq J_{\pi}(\theta) - \frac{2\alpha r_{\max}}{1-\gamma} \alpha c_3 (\epsilon_f c_1(H) + \epsilon_Q c_2(H))$$

**Dynamics error**  $\epsilon_f = \max_t \|\nabla \hat{f}(\hat{s}_t, \hat{a}_t) - \nabla f(s_t, a_t)\|_2$

**Value function error**  $\epsilon_Q = \|\nabla \hat{Q}(\hat{s}_H, \hat{a}_H) - \nabla Q(s_H, a_H)\|_2$

# MAAC Algorithm

---

**Algorithm 1** MAAC

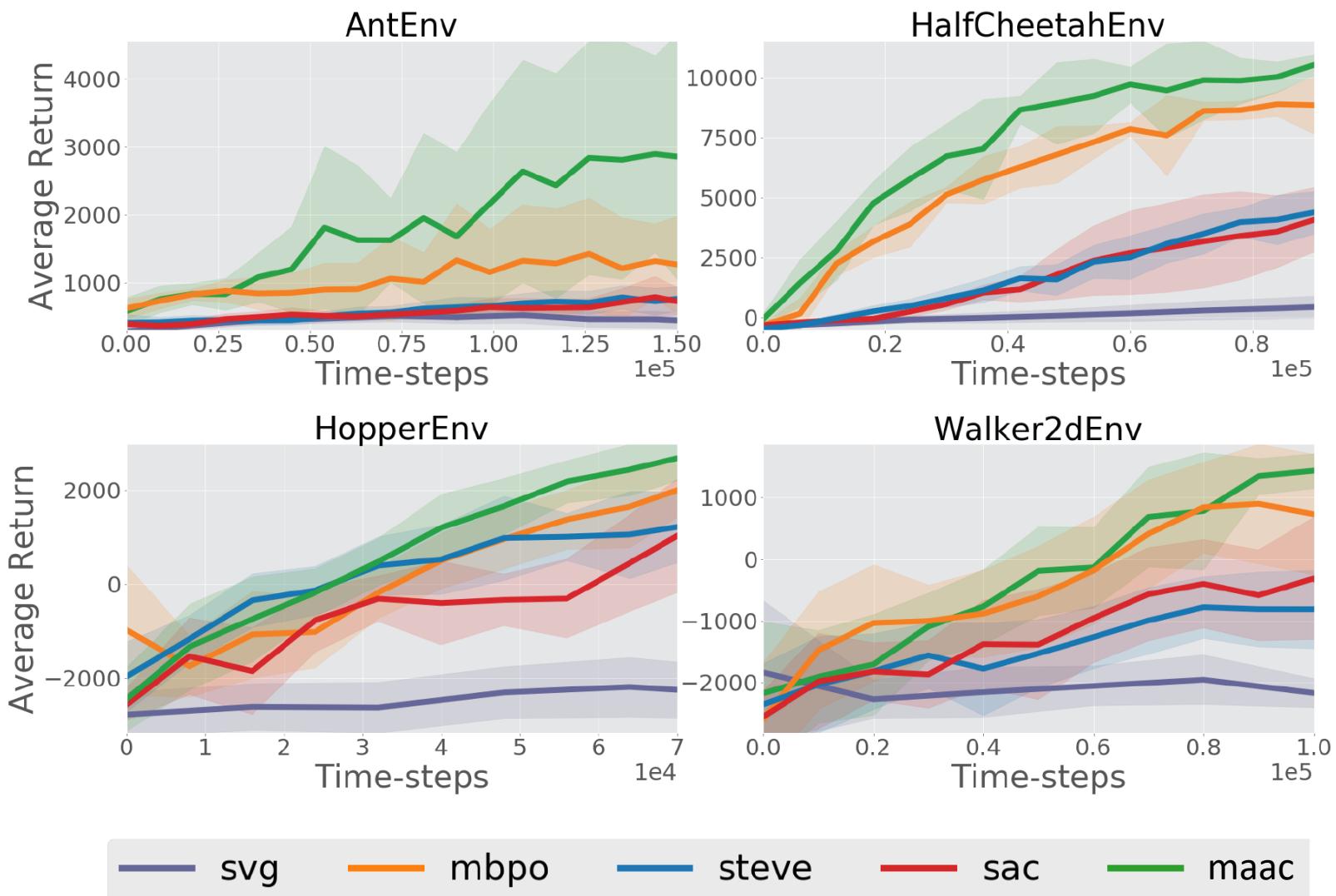
---

- 1: Initialize the policy  $\pi_{\theta}$ , model  $\hat{f}_{\phi}, \hat{Q}_{\psi}, \mathcal{D}_{\text{env}} \leftarrow \emptyset$ , and the model dataset  $\mathcal{D}_{\text{model}} \leftarrow \emptyset$
- 2: **repeat**
- 3:   Sample trajectories from the real environment with policy  $\pi_{\theta}$ . Add them to  $\mathcal{D}_{\text{env}}$ .
- 4:   **for**  $i = 1 \dots G_1$  **do**
- 5:      $\phi \leftarrow \phi - \beta_f \nabla_{\phi} J_f(\phi)$  using data from  $\mathcal{D}_{\text{env}}$ .
- 6:   **end for**
- 7:   Sample trajectories  $\mathcal{T}$  from  $\hat{f}_{\phi}$ . Add them to  $\mathcal{D}_{\text{model}}$ .
- 8:    $\mathcal{D} \leftarrow \mathcal{D}_{\text{model}} \cup \mathcal{D}_{\text{env}}$
- 9:   **for**  $i = 1 \dots G_2$  **do**
- 10:     Update  $\theta \leftarrow \theta + \beta_{\pi} \nabla_{\theta} J_{\pi}(\theta)$  using data from  $\mathcal{D}$
- 11:     Update  $\psi \leftarrow \psi - \beta_Q \nabla_{\psi} J_Q(\psi)$  using data from  $\mathcal{D}$
- 12:   **end for**
- 13: **until** the policy performs well in the real environment
- 14: **return** Optimal parameters  $\theta^*$

---

- Model learning: PILCO - ensemble of GPs  $\{\hat{f}_{\phi_1}, \dots, \hat{f}_{\phi_M}\}$
- Policy Optimization  $J_{\pi}(\theta) = \mathbb{E} \left[ \sum_{t=0}^{H-1} \gamma^t r(\hat{s}_t) + \gamma^H Q_{\psi}(\hat{s}_H, a_H) \right] + \beta \mathcal{H}(\pi_{\theta})$
- Q-function Learning  $J_Q(\psi) = \mathbb{E}[(Q_{\psi}(s_t, a_t) - (r(s_t, a_t) + \gamma Q_{\psi}(s_{t+1}, a_{t+1})))^2]$

# Experiments



References of

# Backpropagation through Paths

- PILCO
  - Deisenroth, Marc, and Carl E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." ICML 2011.
- SVG
  - Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.
- MAAC
  - Ignasi Clavera, Yao Fu, Pieter Abbeel. Model-Augmented Actor Critic: Backpropagation through paths. ICLR 2020.