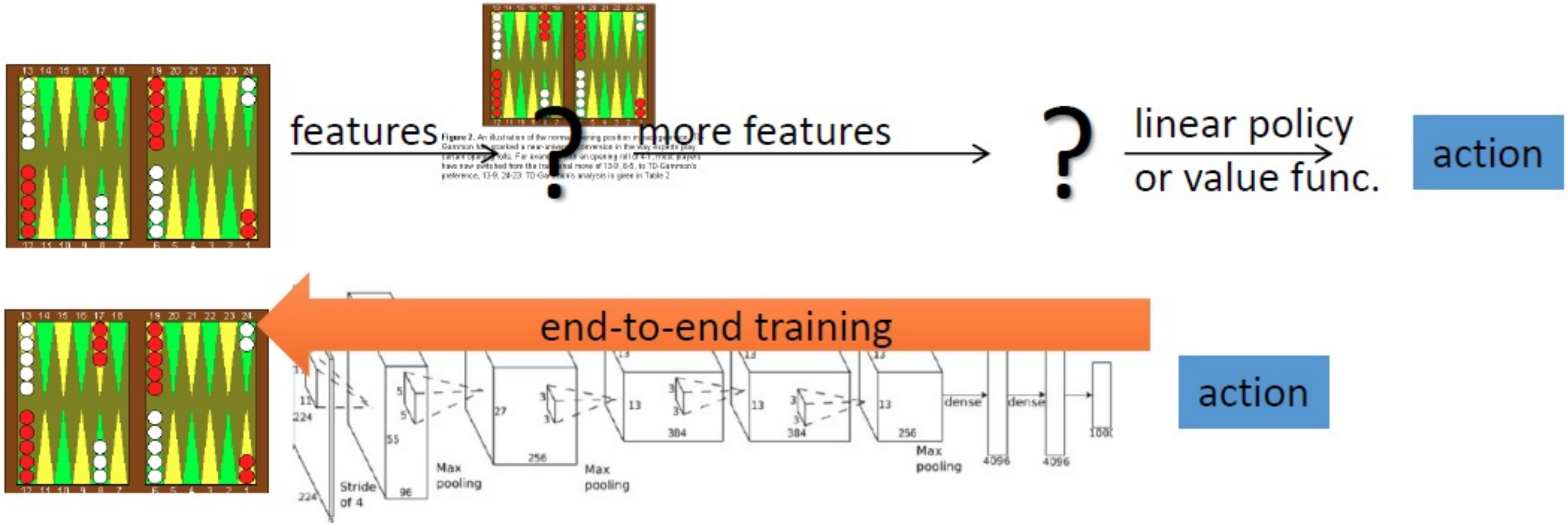
Model-based Reinforcement Learning

A Perspective of

Overall Pathway of DRL

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



Slide from Sergey Levine. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf

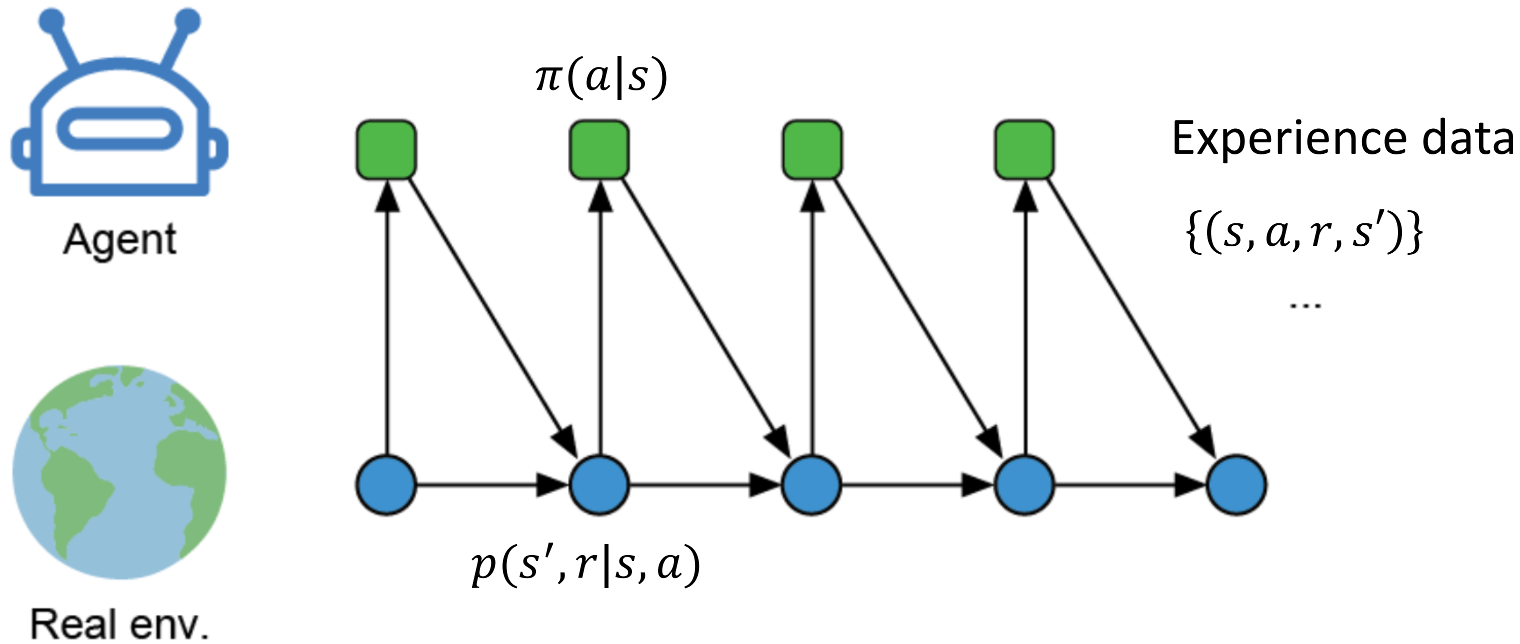
A Perspective of

Overall Pathway of DRL

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

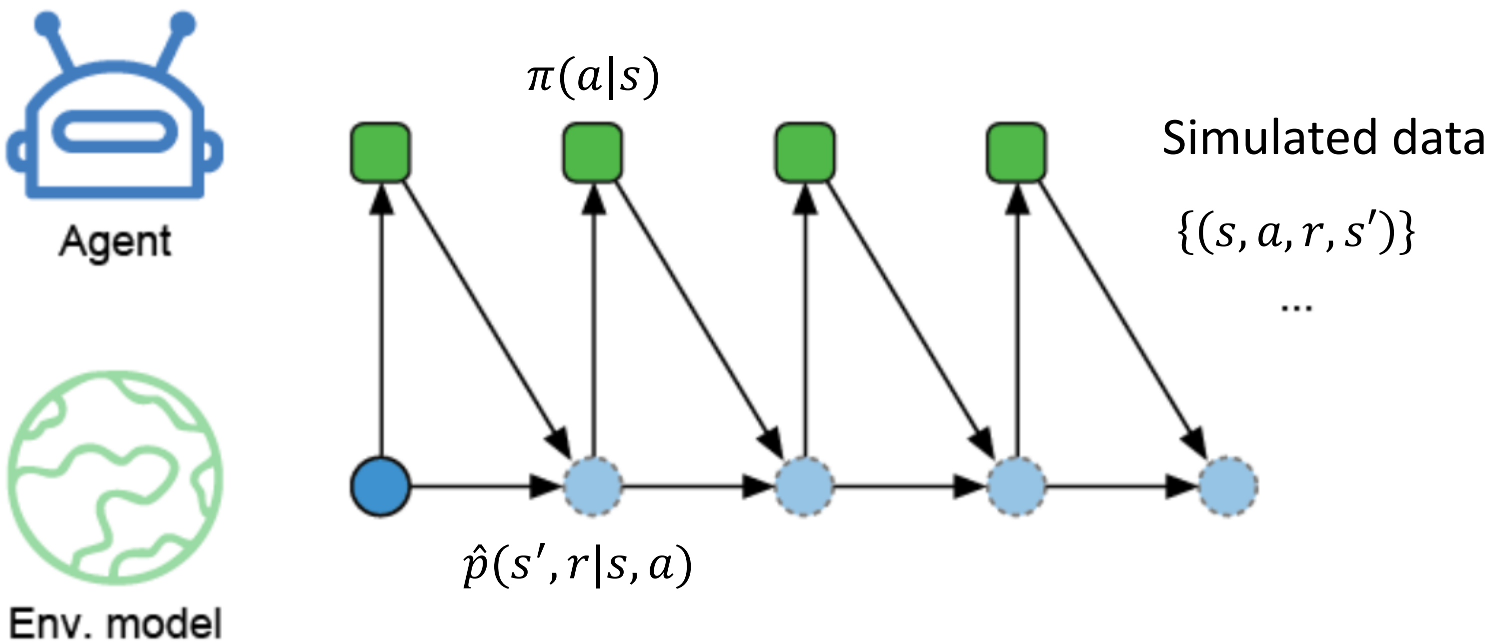
- But DRL has very low data efficiency
  - Trial-and-error learning for deep networks
- A recent popular direction is model-based RL
  - Build a model $p(s', r|s, a)$
  - Based on the model to train the policy
  - So that the data efficiency could be improved

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

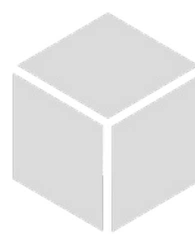
- Seamless to policy training algorithms
- The simulation data efficiency may still be low
- E.g., Dyna-Q, MPC, MBPO

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



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

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

Content

1. Introduction to MBRL from Dyna

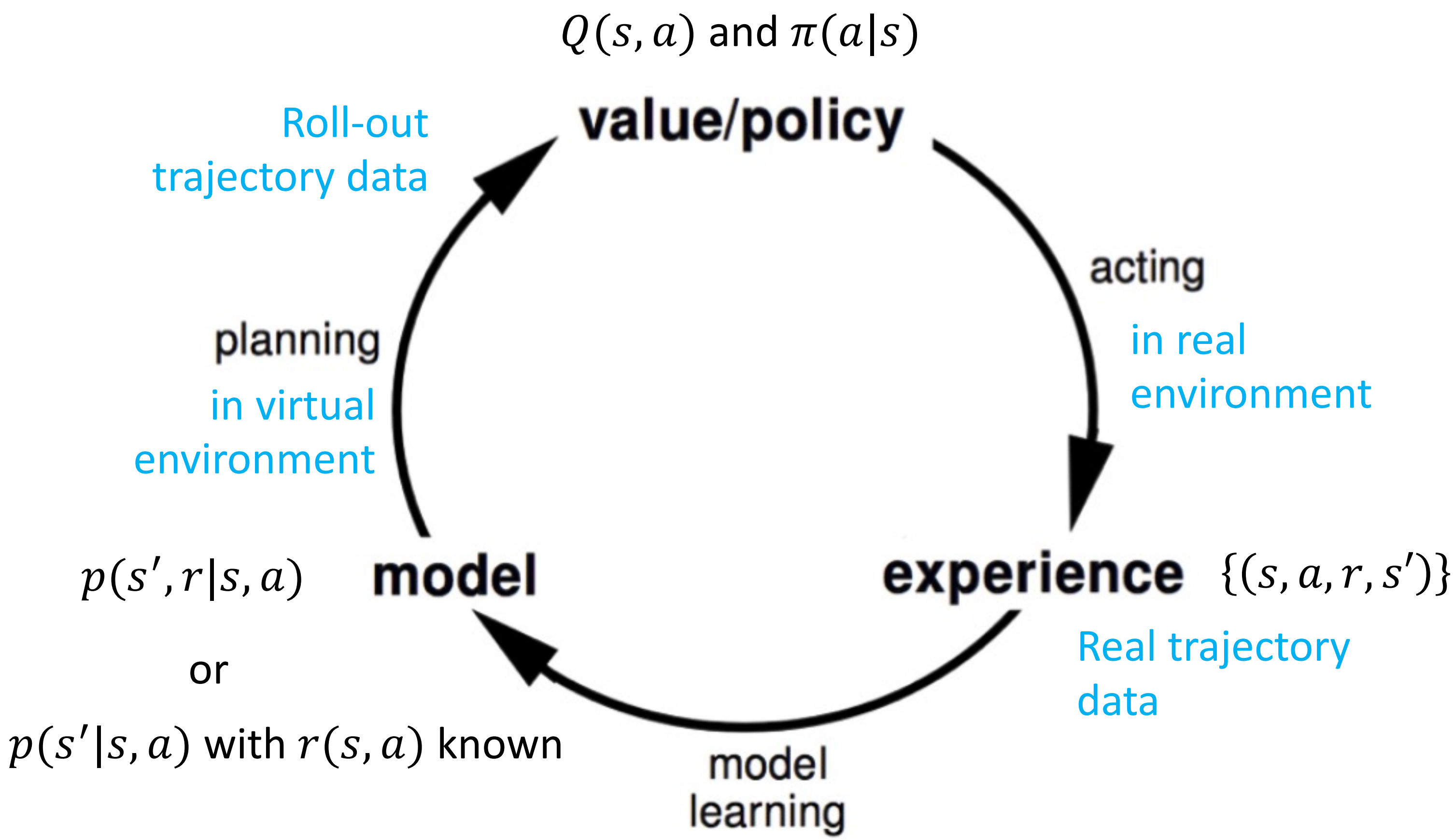
2. Shooting methods: RS & PETS

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

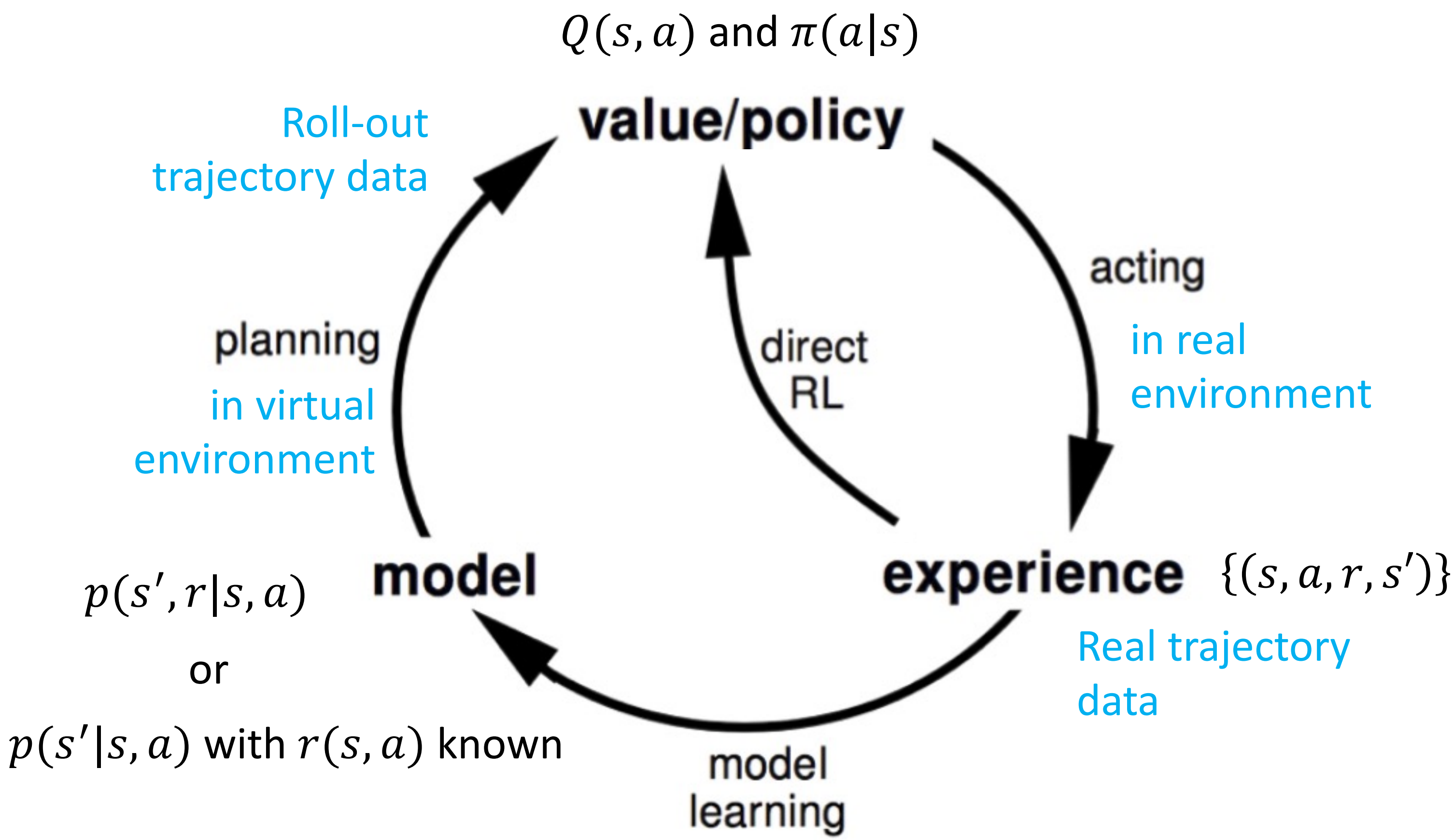
- Random-sample one-step tabular Q-planning
  - First, learn a model $p(s',r|s,a)$ from experience data
  - Then perform one-step sampling by the model to learn the Q function

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':
   $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

- Here model learning and reinforcement learning are separate

Dyna



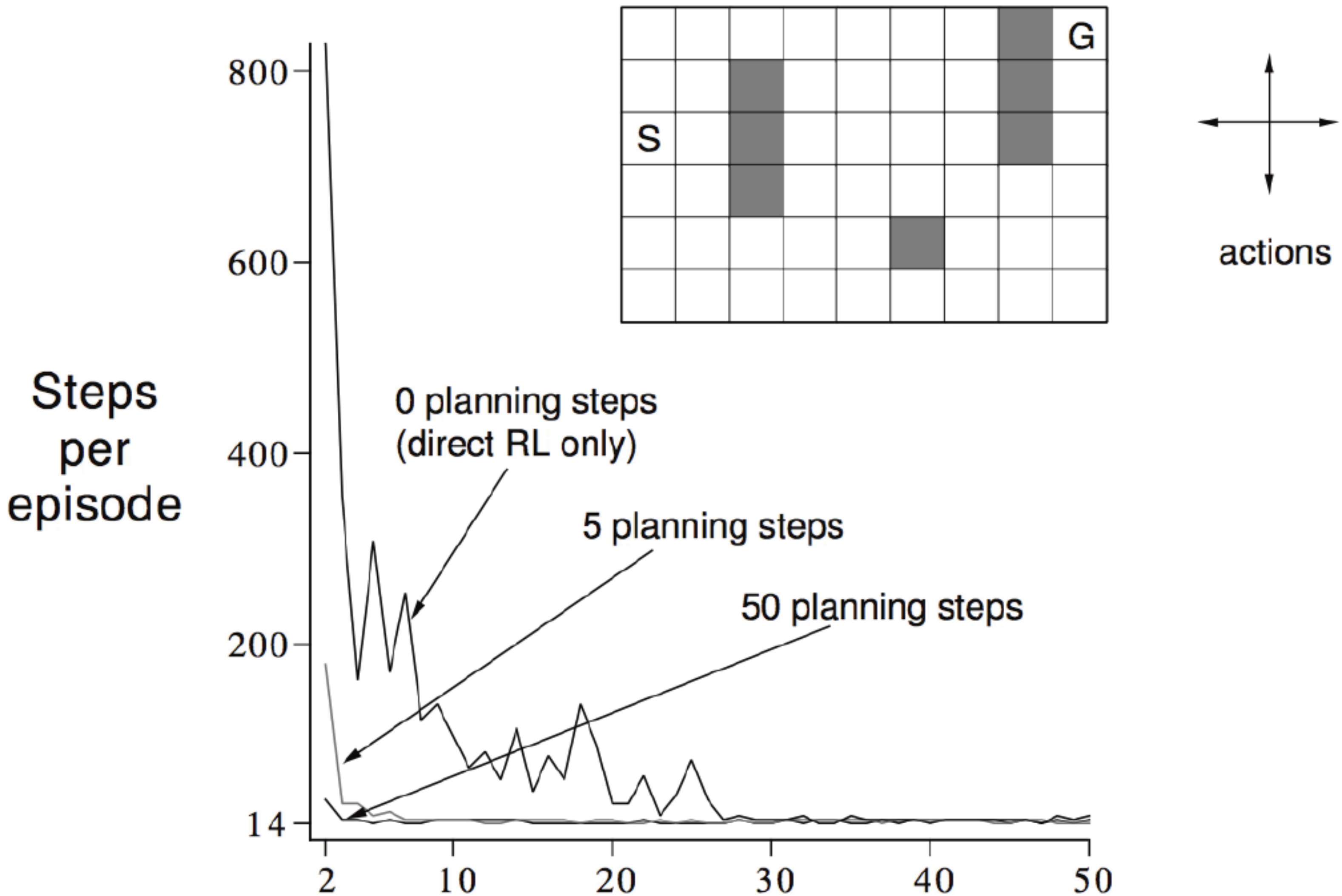
Annotations based on Rich Sutton’s figure

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) $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
(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
  $R, S' \leftarrow Model(S, A)$
  $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$

Sutton, Richard S. "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming." *Machine Learning Proceedings 1990*. Morgan Kaufmann, 1990. 216-224.

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:

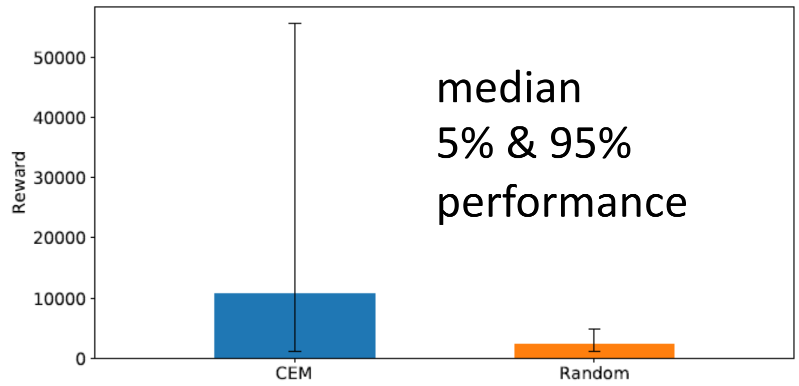
$$[s_0, a_0, \hat{r}_0, \hat{s}_1, a_1, \hat{r}_1, \hat{s}_2, a_2, \hat{r}_2, \dots, \hat{s}_T, a_T, \hat{r}_T]$$

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

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

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



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

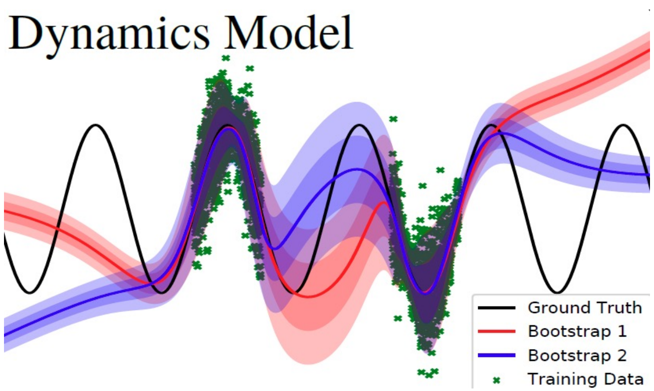
PETS: Probabilistic Ensembles with Trajectory Sampling

Model	Aleatoric uncertainty	Epistemic uncertainty
Baseline Models		
Deterministic NN (D)	No	No
Probabilistic NN (P)	Yes	No
Deterministic ensemble NN (DE)	No	Yes
Gaussian process baseline (GP)	Homoscedastic	Yes
Our Model		
Probabilistic ensemble NN (PE)	**Yes**	**Yes**

$$\text{loss}_{\text{P}}(\boldsymbol{\theta}) = -\sum_{n=1}^{N} \log \widetilde{f}_{\boldsymbol{\theta}}(\boldsymbol{s}_{n+1} | \boldsymbol{s}_n, \boldsymbol{a}_n)$$

Gaussian NN $\widetilde{f} = \Pr(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t) = \mathcal{N}\left(\boldsymbol{\mu}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{\Sigma}_\theta(\boldsymbol{s}_t, \boldsymbol{a}_t)\right)$

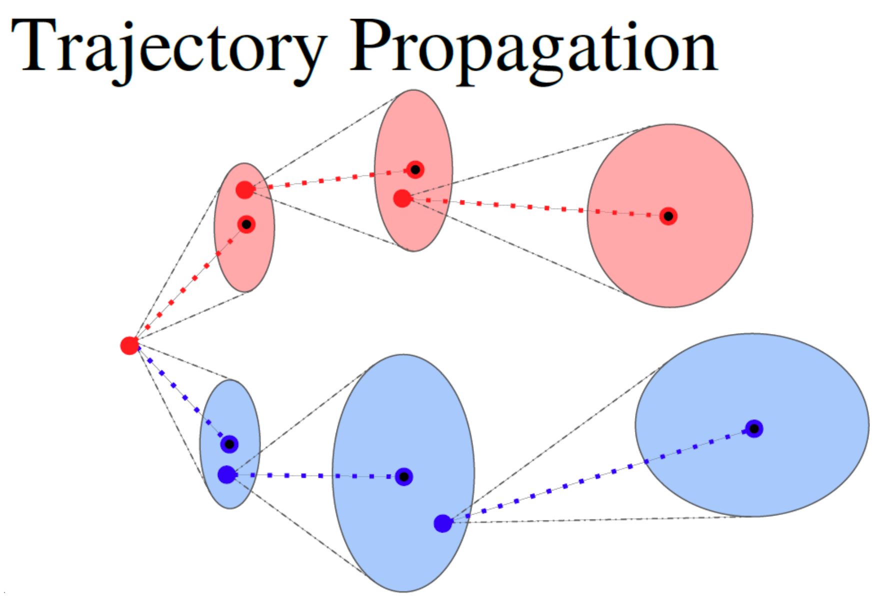
- Probabilistic ensemble (PE) dynamics model is shown as an ensemble of two bootstraps
  - Bootstrap disagreement far from data captures epistemic 认知的 uncertainty: our subjective uncertainty due to a lack of data (model variance)
  - Each probabilistic neural network captures aleatoric 偶然的 uncertainty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



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

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*

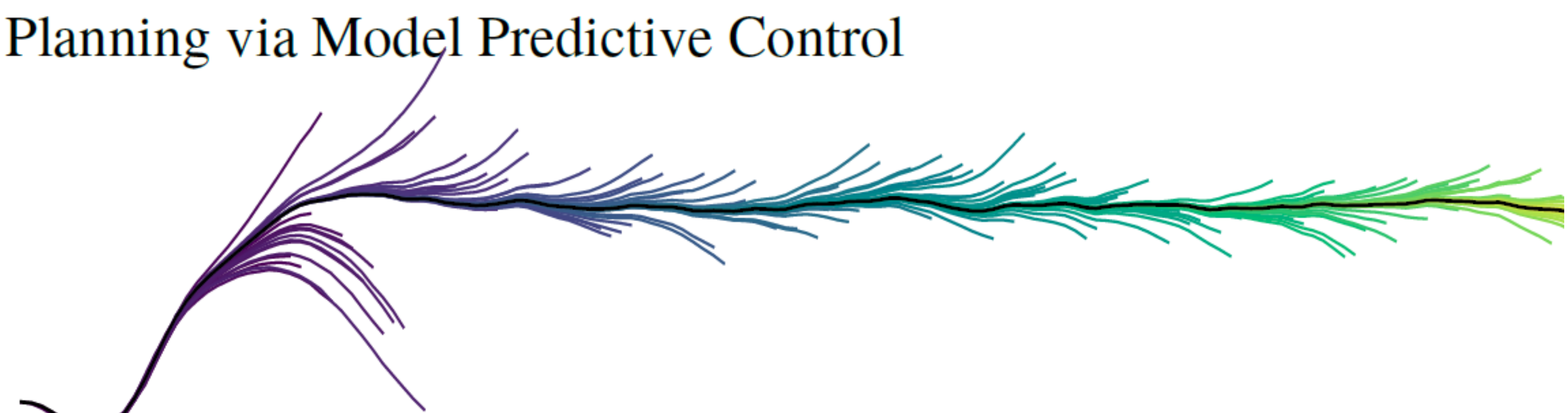


Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." *NIPS* 2018.

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.



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

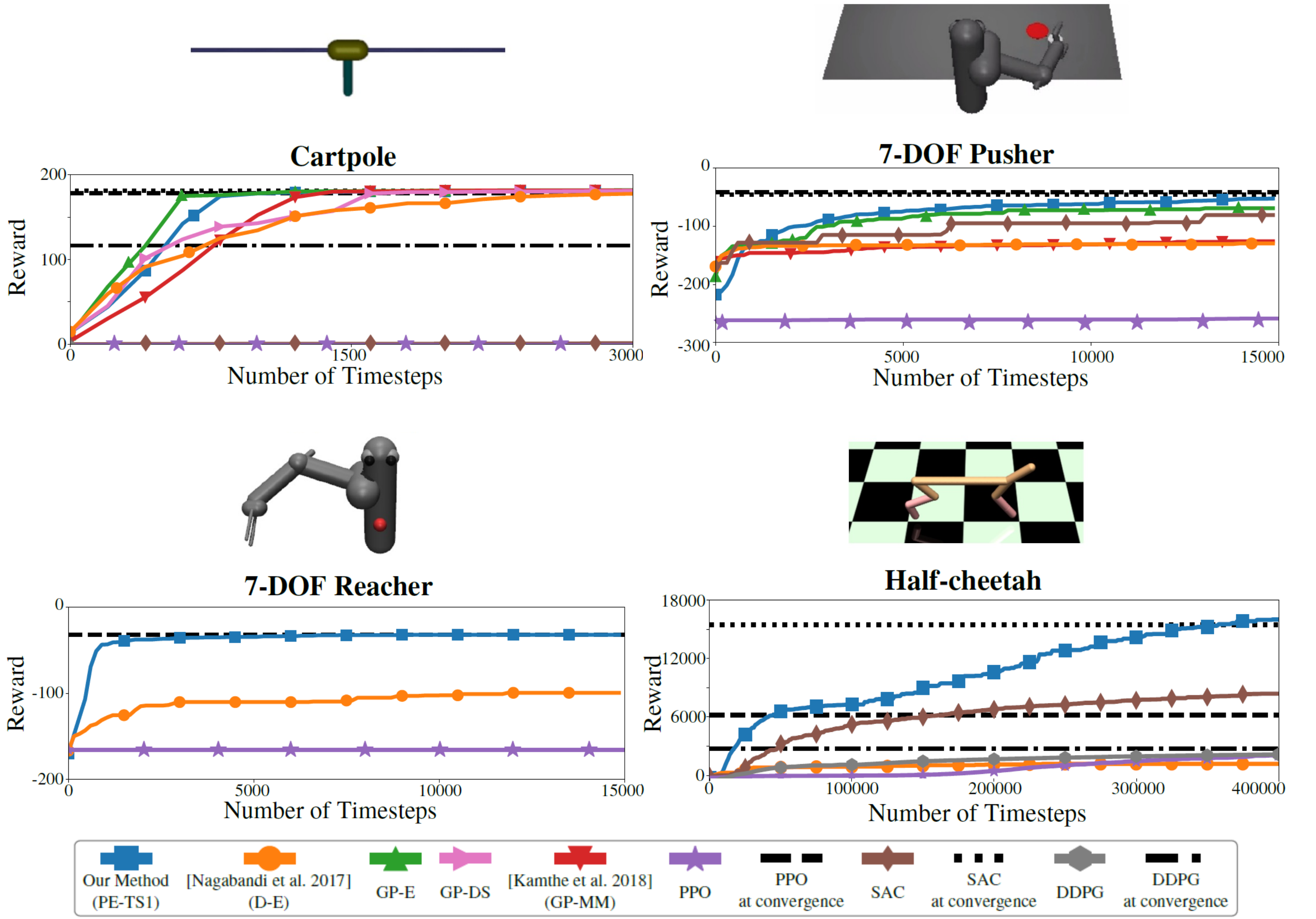
PETS Algorithm

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 $\widetilde{f}$ given $\mathbb{D}$.
4: **for** Time $t = 0$ to TaskHorizon **do**
5: **for** Actions sampled $\boldsymbol{a}_{t:t+T} \sim \text{CEM}(\cdot)$, 1 to NSamples **do**
6: Propagate state particles $\boldsymbol{s}_\tau^p$ using *TS* and $\widetilde{f}|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
7: Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_\tau^p, \boldsymbol{a}_\tau)$
8: Update $\text{CEM}(\cdot)$ distribution.
9: Execute first action $\boldsymbol{a}_t^*$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^*$.
10: 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.

PETS Experiments



Content

1. Introduction to MBRL from Dyna

2. Shooting methods: RS & PETS

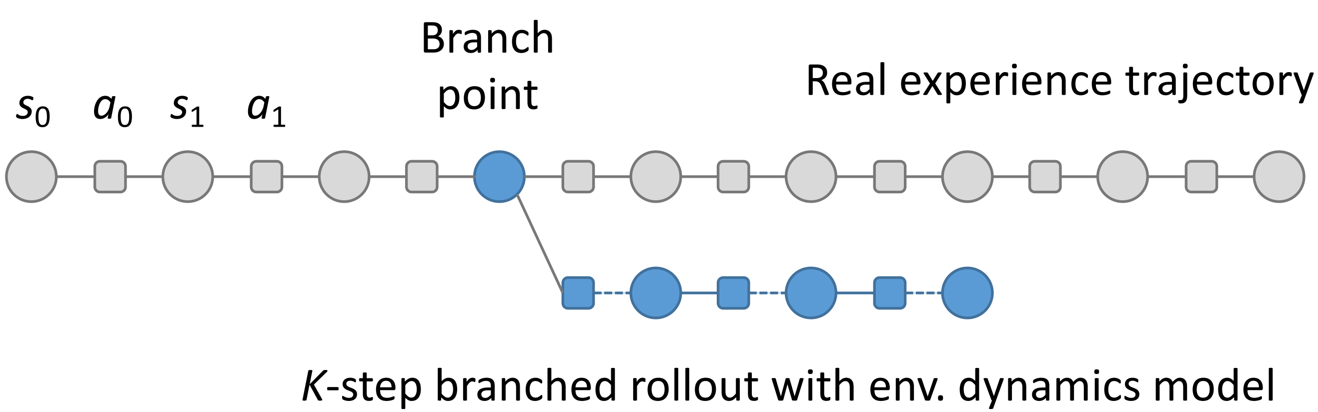
3. Branched rollout method: MBPO

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 π under the learned model p_θ
- Dyna can be viewed as a special case of *k* = 1 branched rollout



Janner, Michael, et al. "When to Trust Your Model: Model-Based Policy Optimization." NIPS 2019.

Bound based on Model & Policy Error

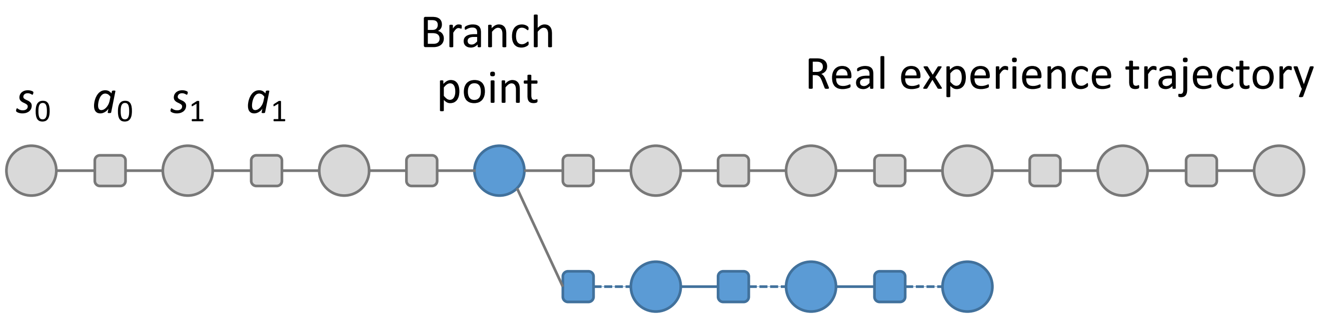
- Quantify model error and policy shift as

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

- The policy value discrepancy bound is written as

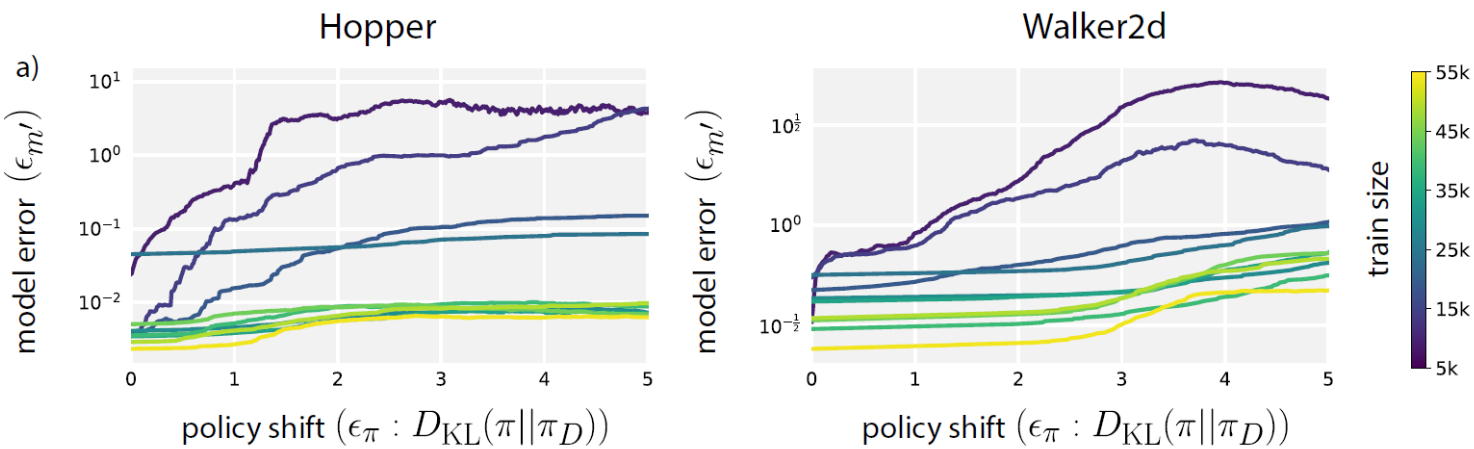
$$\eta[\pi] \geq \eta^{\text{branch}}[\pi] - 2r_{\max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k\epsilon_\pi}{(1-\gamma)} + \frac{k}{1-\gamma}(\epsilon_{m'})\right]$$

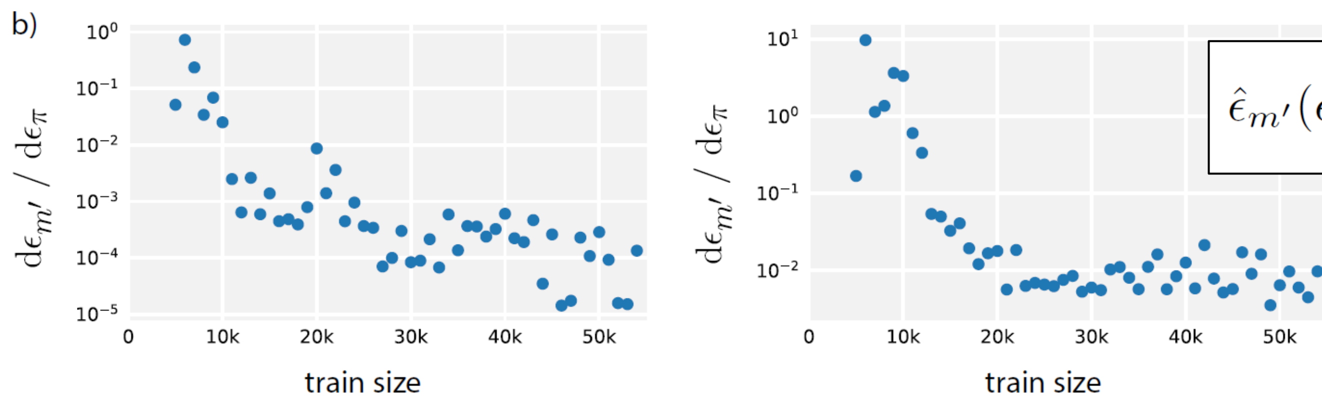
where the optimal $k > 0$ if $\frac{\mathrm{d}\epsilon_{m'}}{\mathrm{d}\epsilon_\pi}$ is sufficiently small



K-step branched rollout with env. dynamics model

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





$$\hat{\epsilon}_{m'}(\epsilon_\pi) \approx \epsilon_m + \epsilon_\pi \frac{d\epsilon_{m'}}{d\epsilon_\pi}$$

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

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

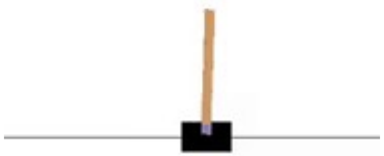
MBPO Algorithm

Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

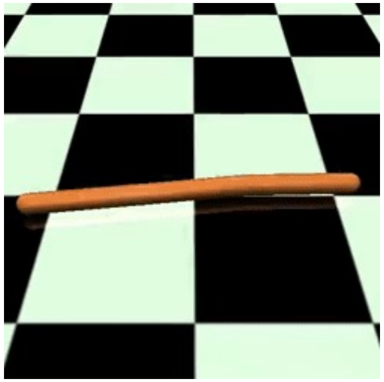
Initialize policy π_ϕ, predictive model p_θ, environment dataset $\mathcal{D}_{\text{env}}$, model dataset $\mathcal{D}_{\text{model}}$
for N epochs **do**
    Train model p_θ on $\mathcal{D}_{\text{env}}$ via maximum likelihood
    for E steps **do**
        Take action in environment according to π_ϕ; 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: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$

- Remarks
  - Branch out from the real trajectories (instead from s_0)
  - Branch rollout k steps depends on model & policy
  - Soft AC to update policy

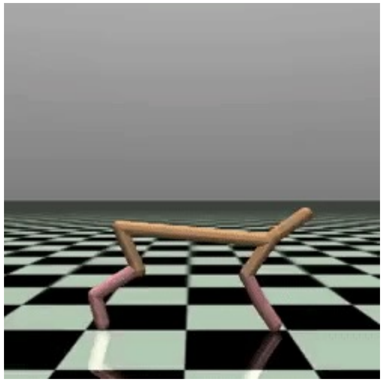
Experiment Environments



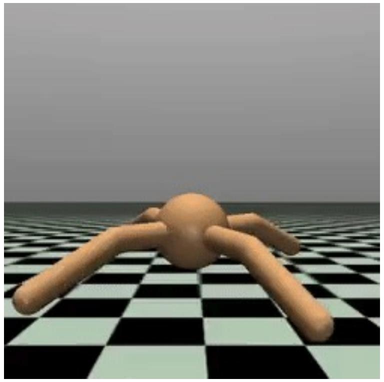
Cartpole



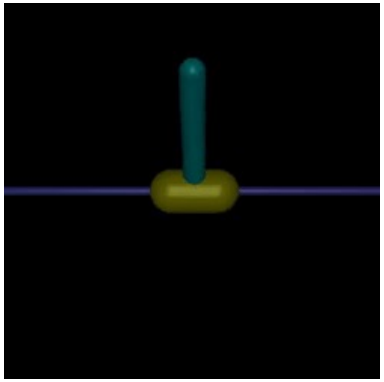
Swimmer



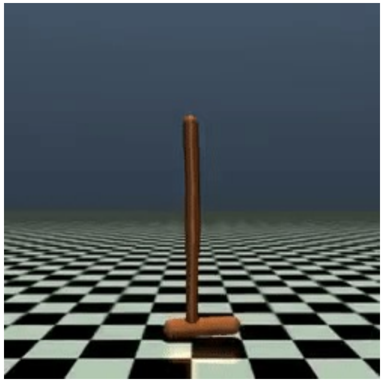
HalfCheetah



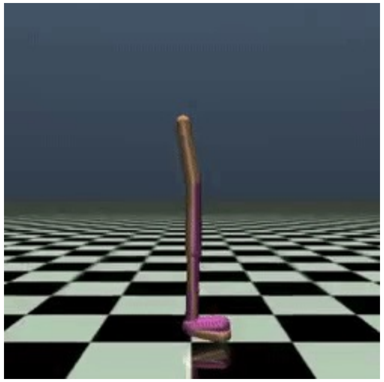
Ant



InvertedPendulum



Hopper

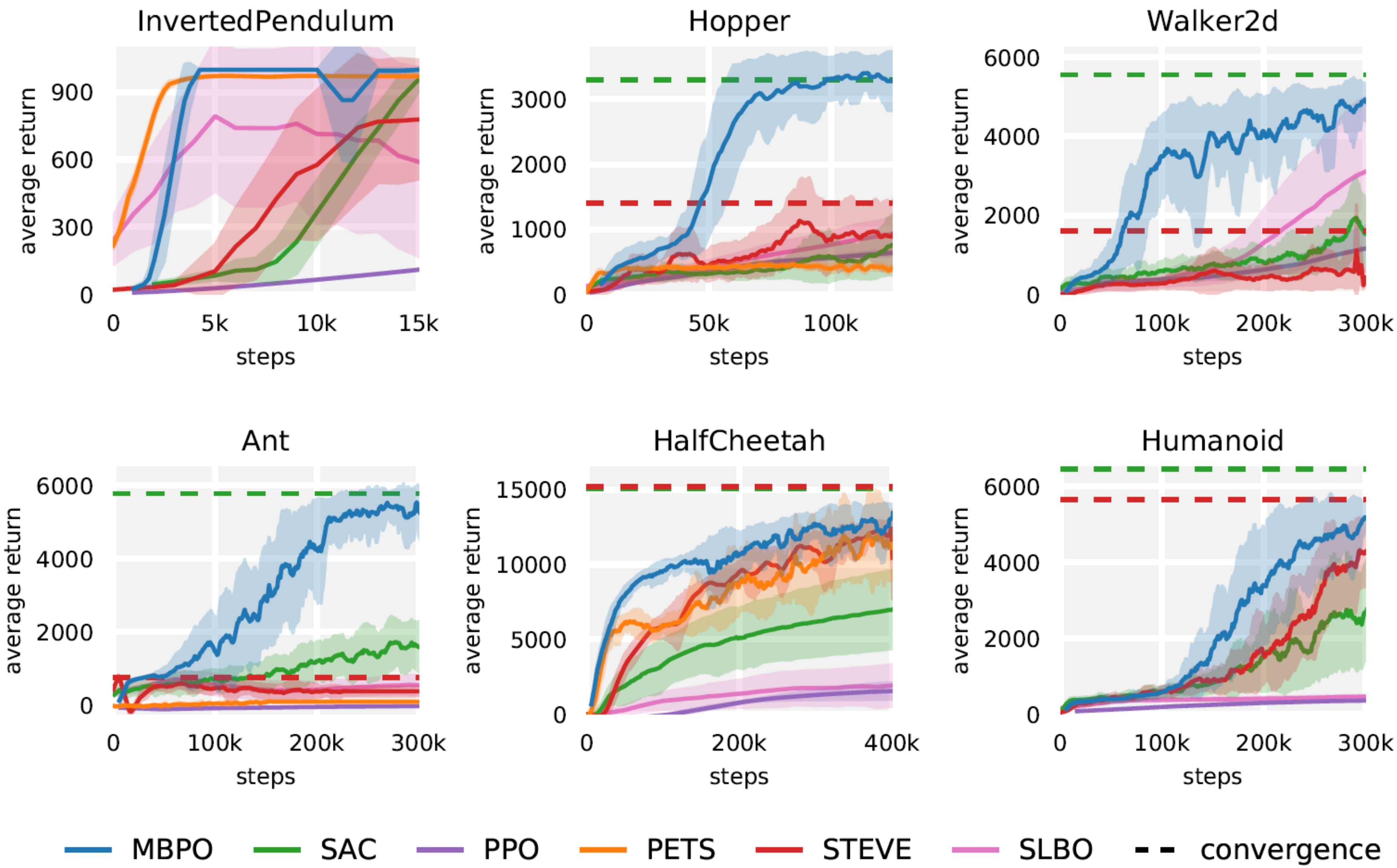


Walker2d

Humanoid

MBPO Experiments

1000-step horizon



Summary of Theoretic Analysis in MBRL

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

lower $|\eta(\pi) - \hat{\eta}(\pi)|$ ➡ more use of model ➡ higher sample efficiency

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

$$|\eta(\pi) - \hat{\eta}(\pi)| \leq C(\underset{\substack{\uparrow \\ \text{Generalization error of the model}}}{\varepsilon_m}, \underset{\substack{\uparrow \\ \text{Policy shift during training}}}{\varepsilon_\pi})$$

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

Content

1. Introduction to MBRL from Dyna

2. Shooting methods: RS & PETS

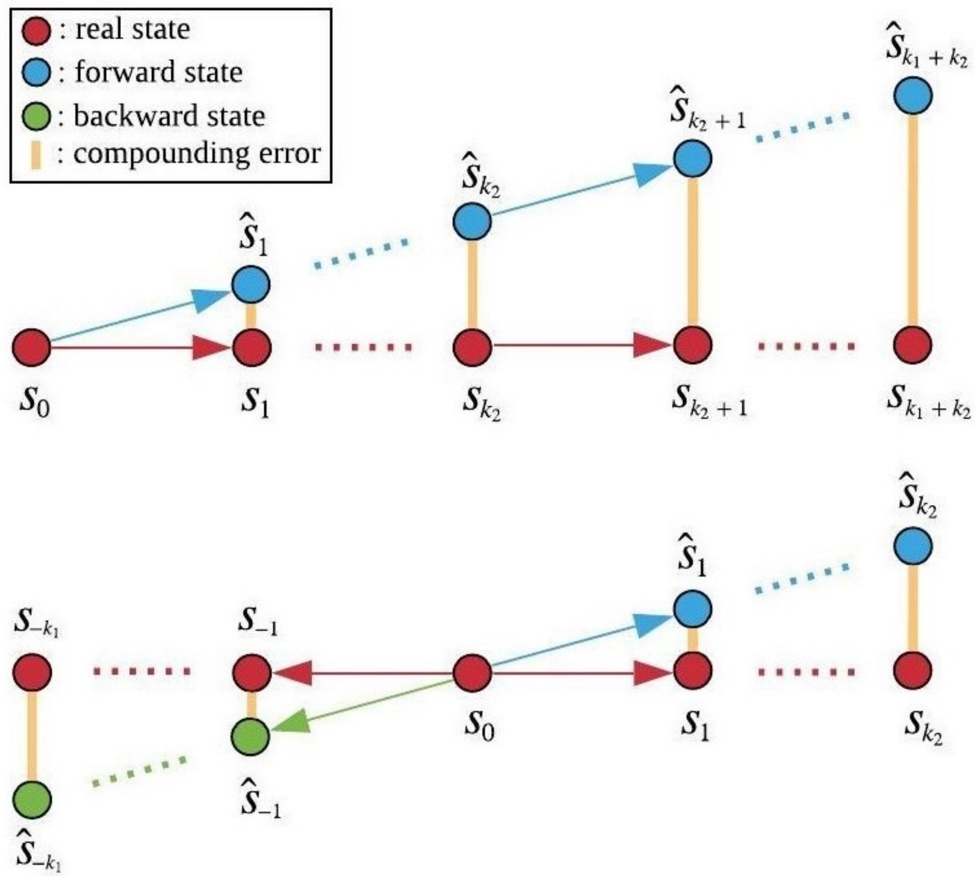
3. Branched rollout method: MBPO

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



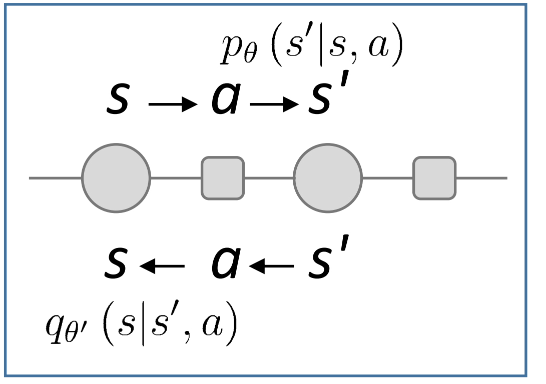
forward model

bidirectional model

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

Dynamics Model Learning

- An ensemble of bootstrapped probabilistic networks are used to parameterize both the forward model $p_\theta\left(s'|s,a\right)$ and the backward model $q_{\theta'}\left(s|s',a\right)$



- Each probabilistic neural network outputs a Gaussian distribution with diagonal covariance and is trained via maximum likelihood. The corresponding loss functions are:

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

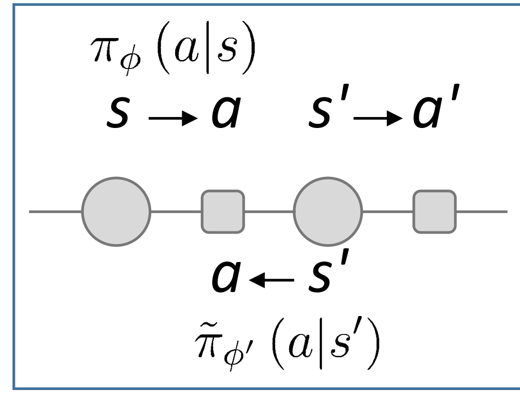
$$\mathcal{L}_b(\theta')=\sum_{t=1}^{N}\left[\mu_{\theta'}\left(s_{t+1},a_t\right)-s_t\right]^\top\Sigma_{\theta'}^{-1}\left(s_{t+1},a_t\right)\left[\mu_{\theta'}\left(s_{t+1},a_t\right)-s_t\right]+\log\det\Sigma_{\theta'}\left(s_{t+1},a_t\right)$$

where μ and Σ are the mean and covariance respectively, and N denotes the total number of real transition data

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

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:

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

- or conditional GAN (GAIL):

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

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

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

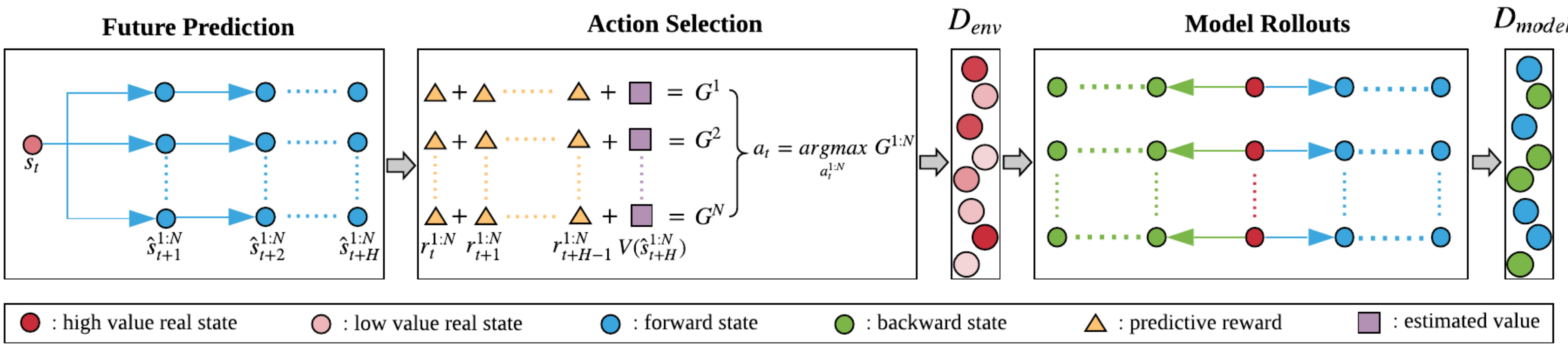
- Incorporating MPC (Model Predictive Control) to refine the action taken in real environment.
  - At each time-step, candidate action sequences are generated from current policy and the corresponding trajectories are simulated by the learned model
  - Then the first action of the sequence that yields the highest accumulated rewards is selected:

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

Truncate the rollout with value function

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

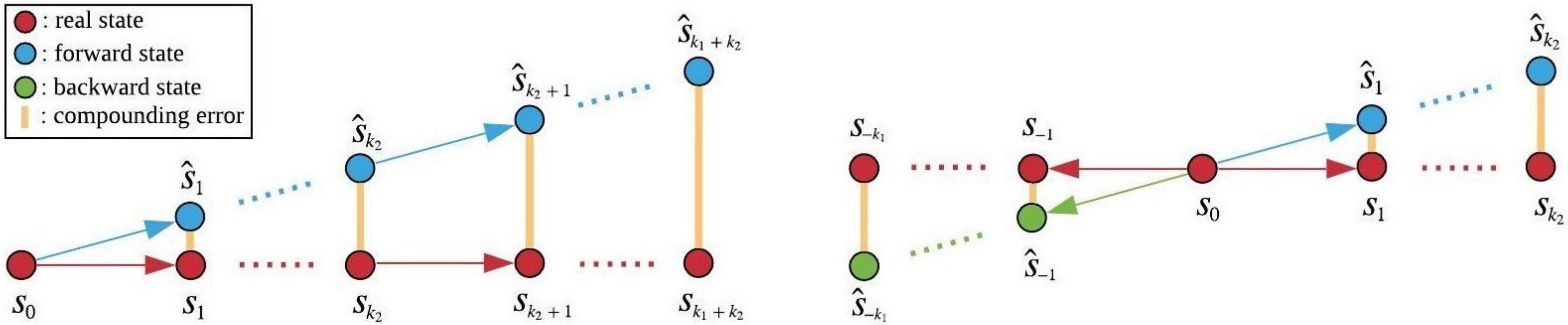
Overall Algorithm of BMPO



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}

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

Theoretical Analysis



forward model　　　　bidirectional model

Bidirectional model:

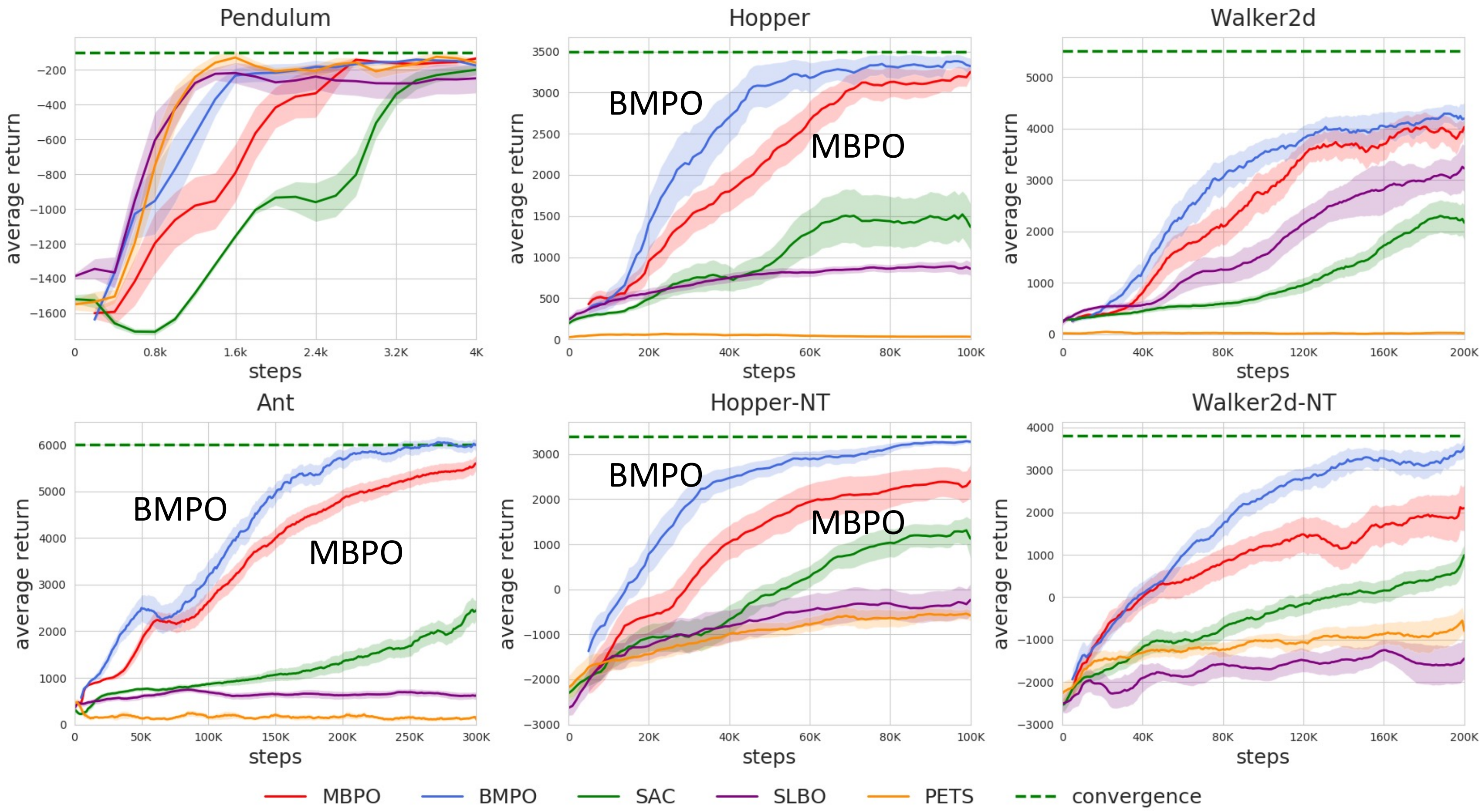
$$\left|\eta[\pi] - \eta^{\text{branch}}[\pi]\right| \leq 2r_{\max}\left[\frac{\gamma^{k_1+k_2+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k_1+k_2}\epsilon_\pi}{(1-\gamma)} + \frac{\boxed{\max(k_1,k_2)}\epsilon_m}{1-\gamma}\right].$$

Forward model:

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

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

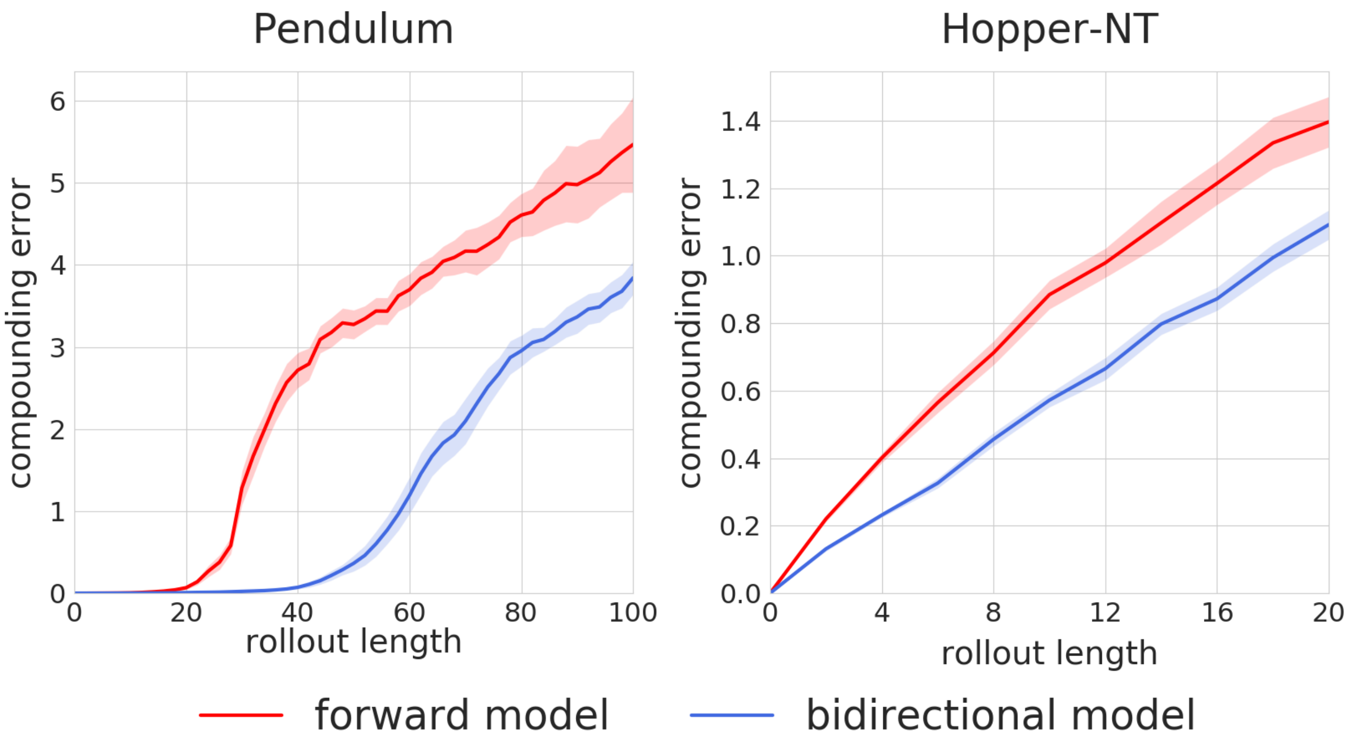
Comparison with State-of-the-Arts



- 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

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

Model Compounding Error



$$\mathrm{Error}_{\mathrm{for}} = \frac{1}{2h}\sum_{i=1}^{2h} \|\hat{s}_i - s_i\|_2^2$$

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

Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.

Content

1. Introduction to MBRL from Dyna

2. Shooting methods: RS & PETS

3. Branched rollout method: MBPO

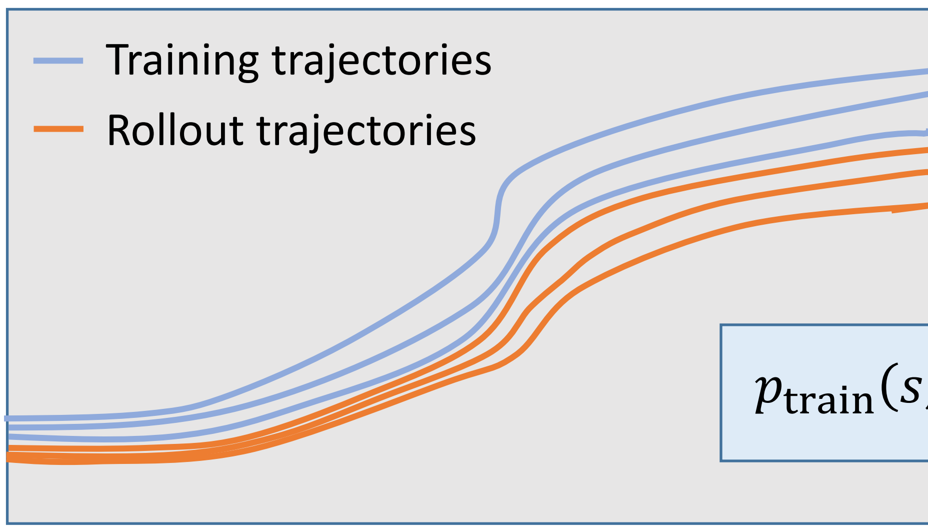
4. Recent work: BMPO, AMPO and AutoMBPO

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

State



Time

State transition
$T(s'|s,a)$

$$p_{\text{train}}(s,a) \neq p_{\text{rollout}}(s,a)$$

Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

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

Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

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 (IPM)

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

- IPM measures 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 $\epsilon_\pi := 2 d_{\mathrm{TV}}(\nu_T^\pi, \nu_T^{\pi_D})$. Under the assumption of Lemma 3.1, the expected return $\eta[\pi]$ admits the following bound:

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

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

policy learning in model + model learning = basic MBRL

Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

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 $\epsilon_\pi := 2 d_{\mathrm{TV}}(\nu_T^\pi, \nu_T^{\pi_D})$. Under the assumption of Lemma 3.1, the expected return $\eta[\pi]$ admits the following bound:

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

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

reliable exploitation in batch RL

distribution distance

(violate the rule of exploration)

Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

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*

$$\boxed{\epsilon_t(h)} \leq \boxed{\epsilon_s(h)} + \boxed{2KW_1(\mu_s, \mu_t)} + \boxed{\lambda} \tag{11}$$

error in target domain

error in source domain

Wasserstein distance

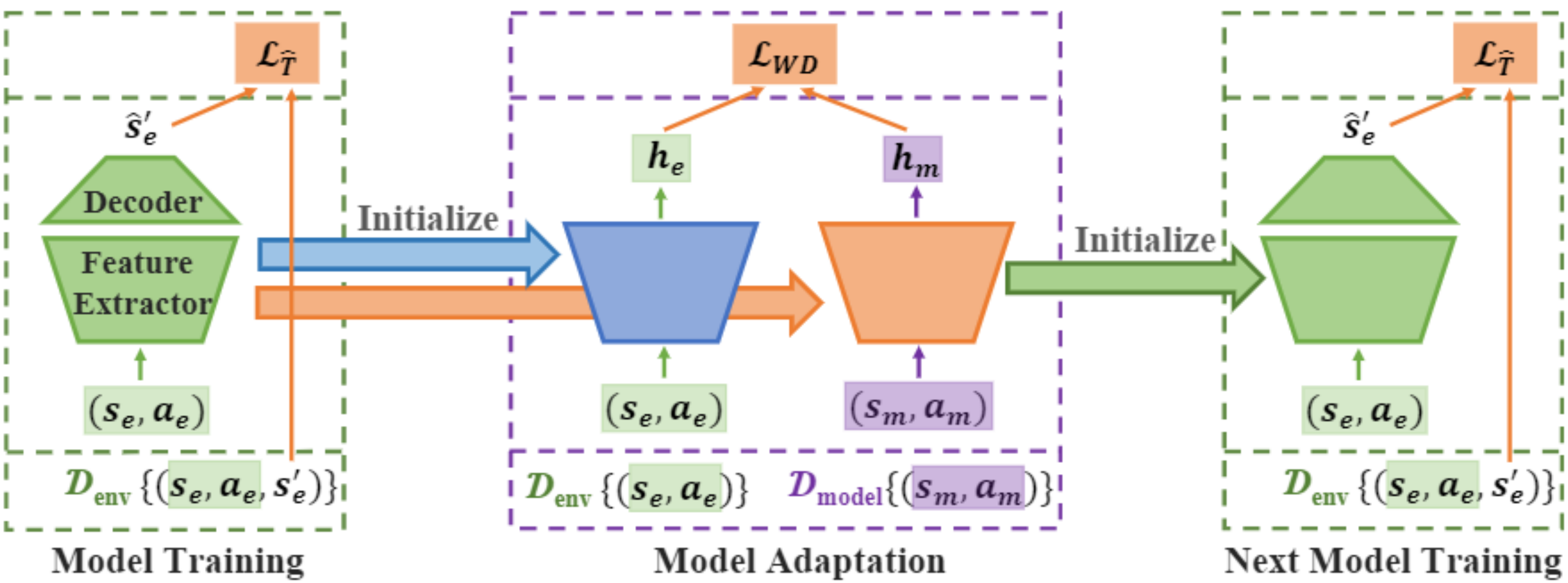
Constant

Inspiration: aligning the two feature distributions in MBRL

Shen, Jian, et al. "Wasserstein distance guided representation learning for domain adaptation." AAAI 2018.

Unsupervised Model Adaptation

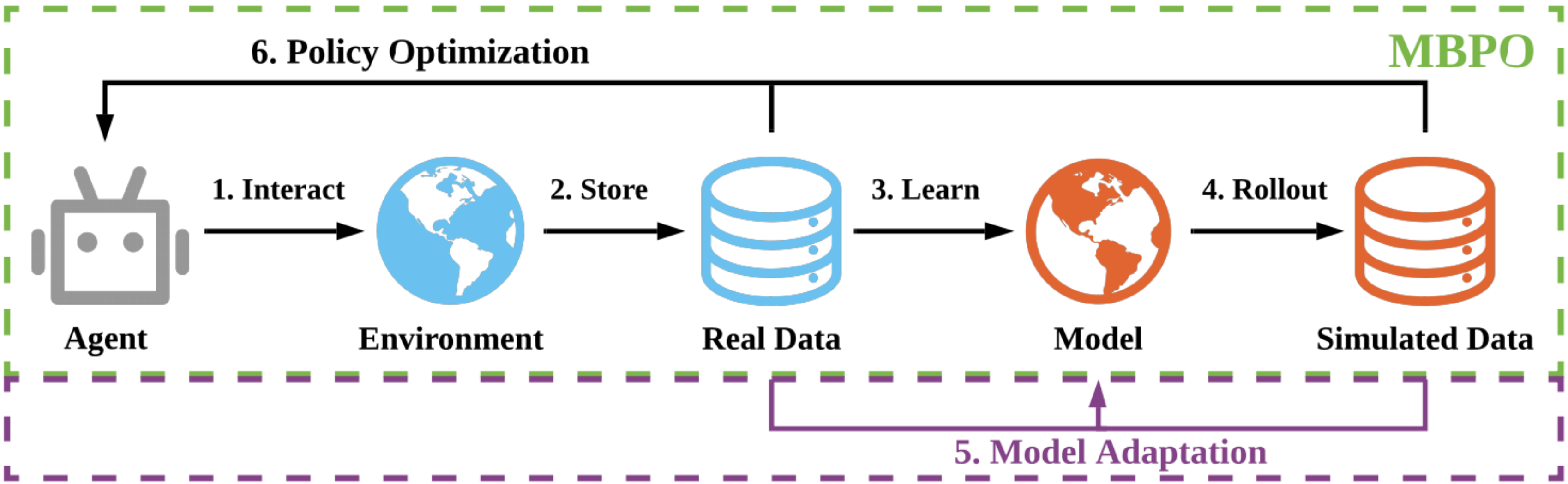
- Source domain: model training data
- Target domain: model rollout data
- Aligning the latent feature distributions by minimizing IPMs according to the lower bound



Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

AMPO

- Adaptation augmented Model-based Policy Optimization



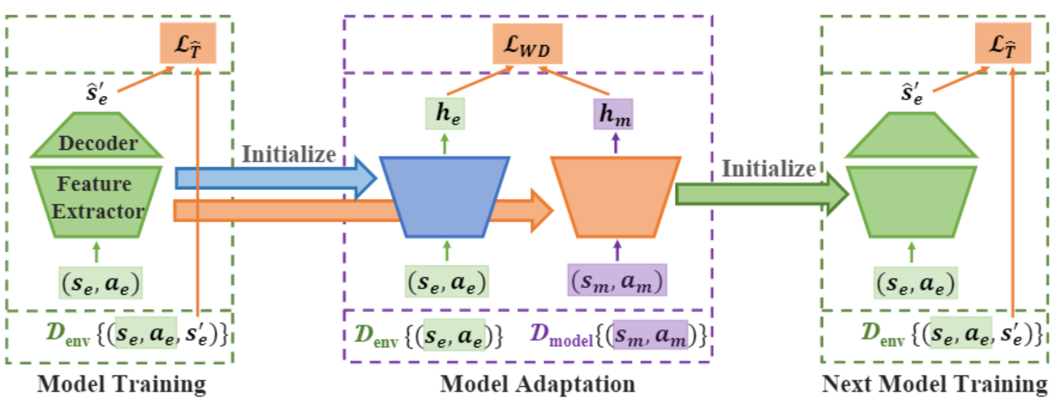
- AMPO: Wasserstein-1 distance (WGAN)
- Variants: use other distribution divergence, e.g. MMD (Maximum Mean Discrepancy)

Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

Overall Algorithm

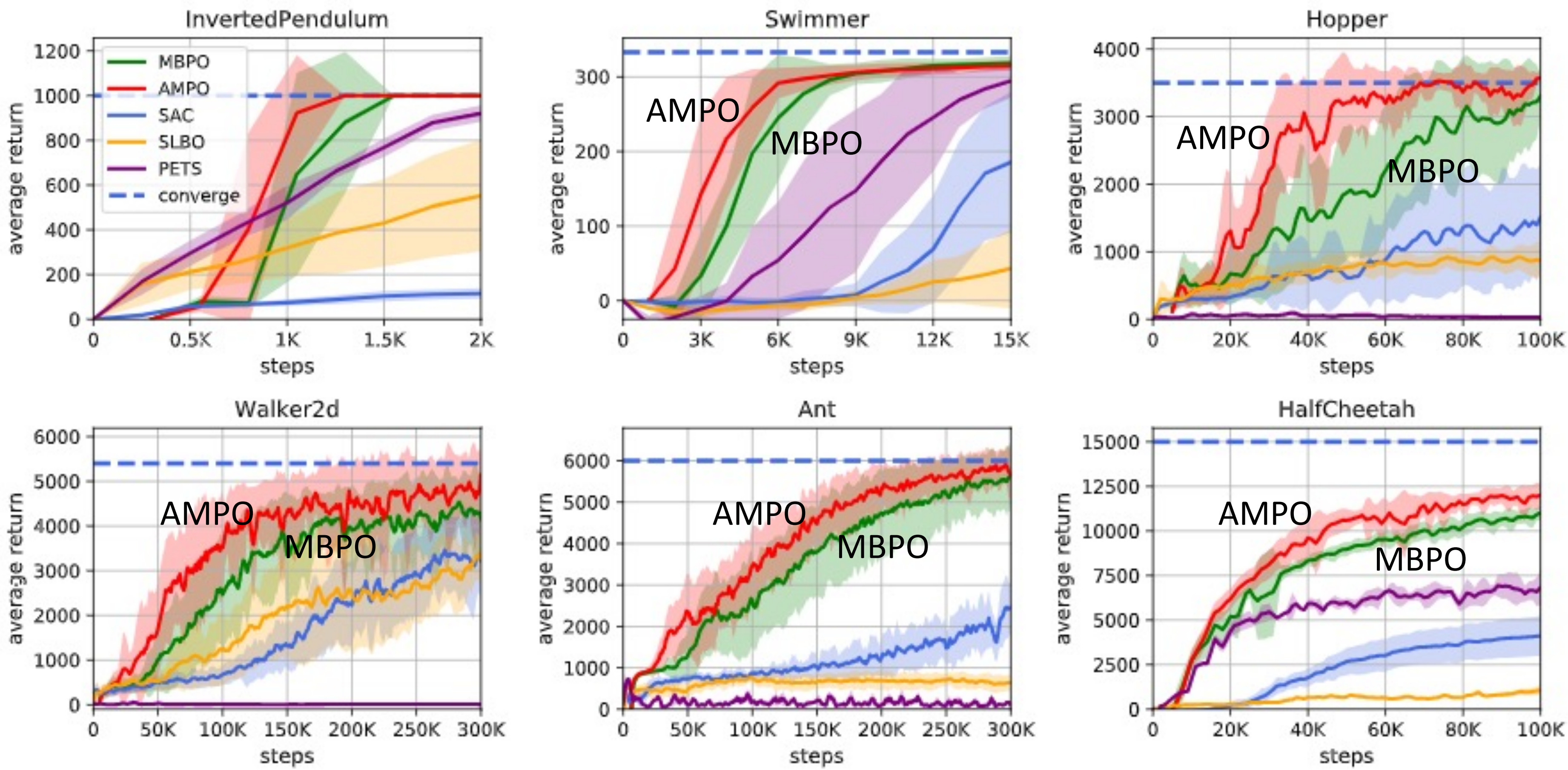
Algorithm 1 AMPO

Initialize policy π_ϕ, dynamics model $\hat{T}_\theta$, environment buffer $\mathcal{D}_e$, model buffer $\mathcal{D}_m$
repeat
  Take an action in the environment using the policy π_ϕ; add the sample(s, a, s', r) to $\mathcal{D}_e$
  if every E real timesteps are finished **then**
    Perform G_1 gradient steps to train the model $\hat{T}_\theta$ with samples from $\mathcal{D}_e$
    for F model rollouts **do**
      Sample a state s uniformly from $\mathcal{D}_e$
      Use policy π_ϕ to perform a k-step model rollout starting from s; add to $\mathcal{D}_m$
    end for
    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 $\mathcal{L}_{\text{WD}}$
  end if
  Perform G_3 gradient steps to train the policy π_ϕ with samples (s, a, s', r) from $\mathcal{D}_e \cup \mathcal{D}_m$
until certain number of real samples



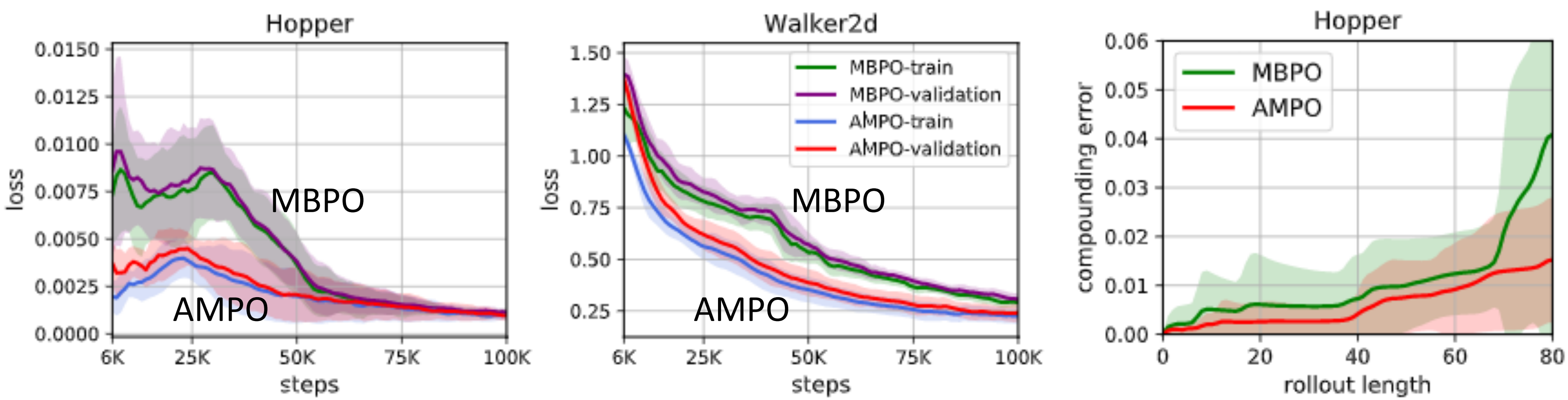
Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

Comparison with State-of-the-Arts



Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

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

Jian Shen, Han Zhao, Weinan Zhang, Yong Yu. Model-based Policy Optimization with Unsupervised Model Adaptation. NeurIPS 2020.

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

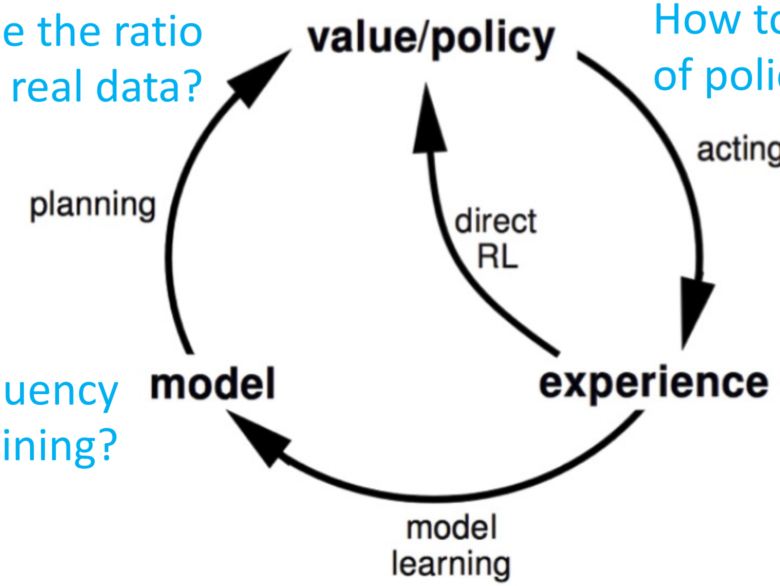
How to schedule the ratio of real data?

How to tune the frequency of policy training?

What is the rollout length k?

How much to interact with real data to make the best balance of performance and sample efficiency?

How to tune the frequency of model training?



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

Hang Lai, et al. "On Effective Scheduling of Model-based Reinforcement Learning." NIPS2021

Analysis of real ratio schedule

Return discrepancy upper bound:

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

Real Ratio

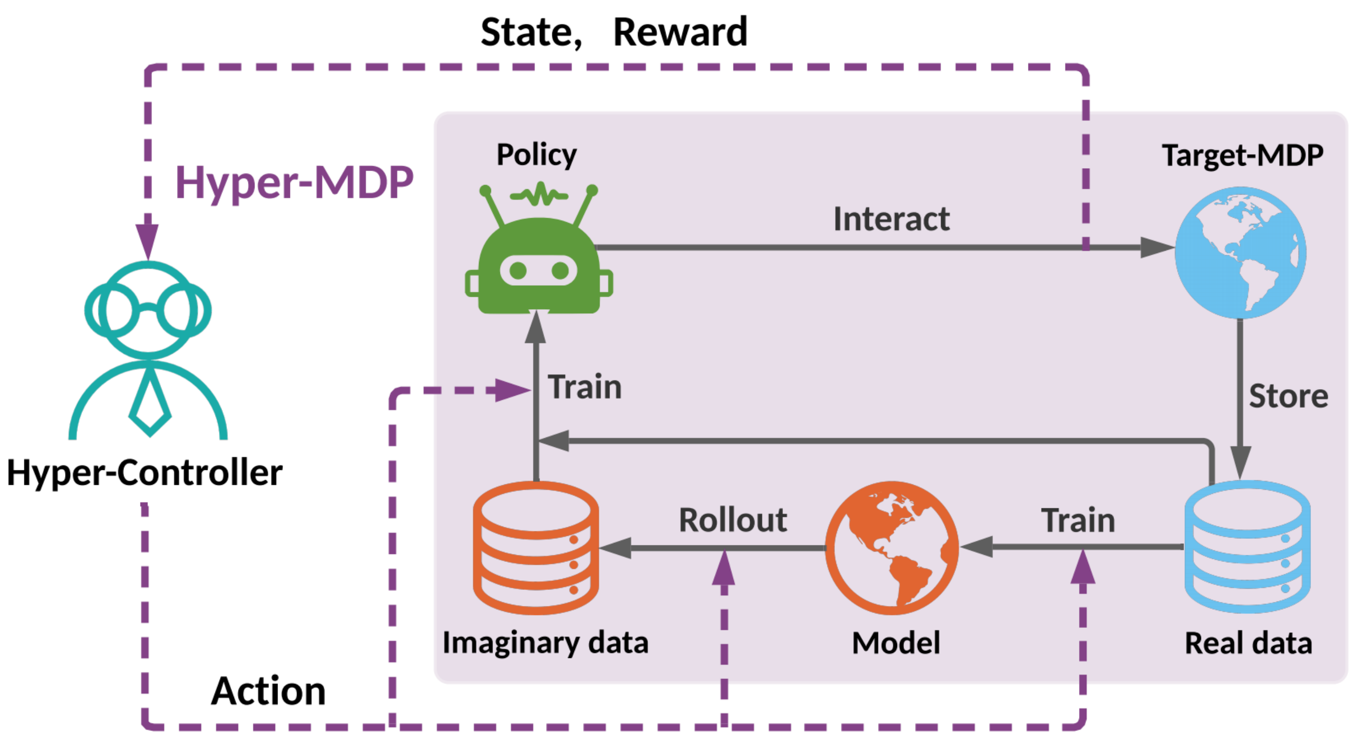
Real data number

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

Hang Lai, et al. "On Effective Scheduling of Model-based Reinforcement Learning." NIPS2021

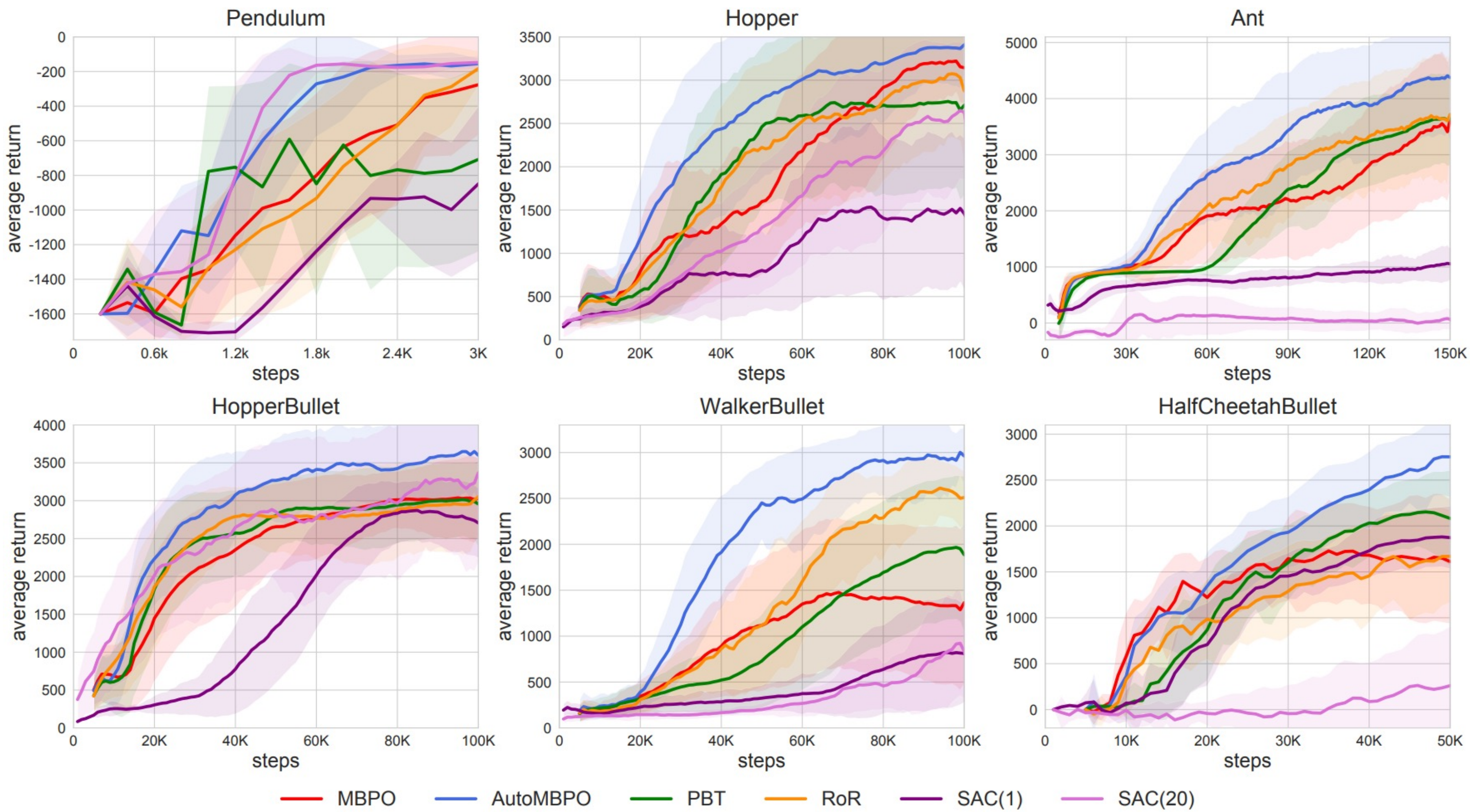
AutoMBPO Framework

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



Hang Lai, et al. "On Effective Scheduling of Model-based Reinforcement Learning." NIPS2021

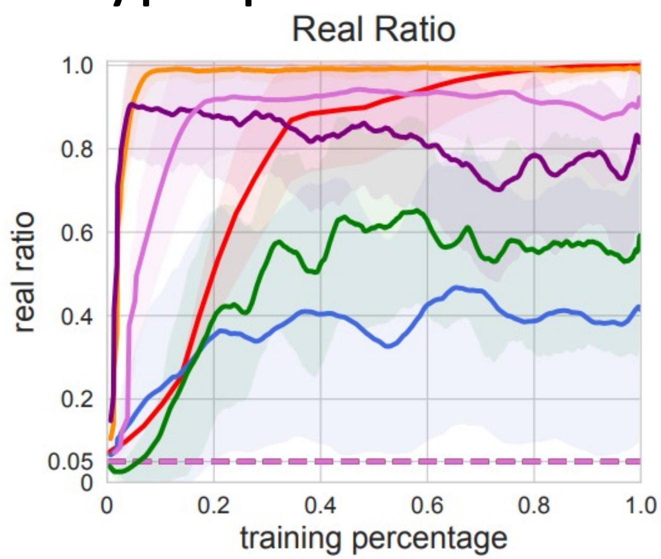
AutoMBPO Experiments



Hang Lai, et al. "On Effective Scheduling of Model-based Reinforcement Learning." NIPS2021

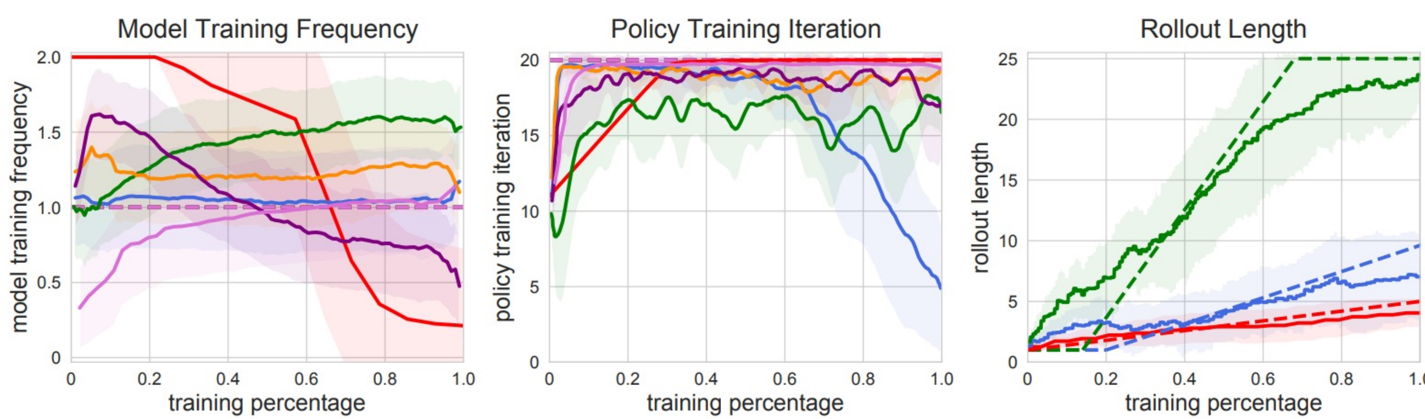
AutoMBPO Experiments

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.

Pendulum Hopper Ant HopperBullet
WalkerBullet HalfCheetahBullet MBPO

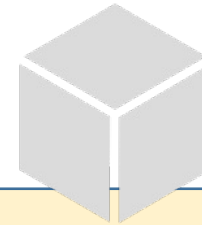


Hang Lai, et al. "On Effective Scheduling of Model-based Reinforcement Learning." NIPS2021

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

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

REVIEW

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

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

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

On-policy Chain rule

D. Silver et al. Deterministic Policy Gradient Algorithms. ICML 2014.

From Deterministic to Stochastic

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

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

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

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

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

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

Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.

From Deterministic to Stochastic

- Math: reparameterization of distributions
  - Conditional Gaussian distribution

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

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

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

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

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

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

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

Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.

From Deterministic to Stochastic

- Deterministic version for reference

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

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

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

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

- Stochastic environment, policy, value and gradients

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

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

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

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

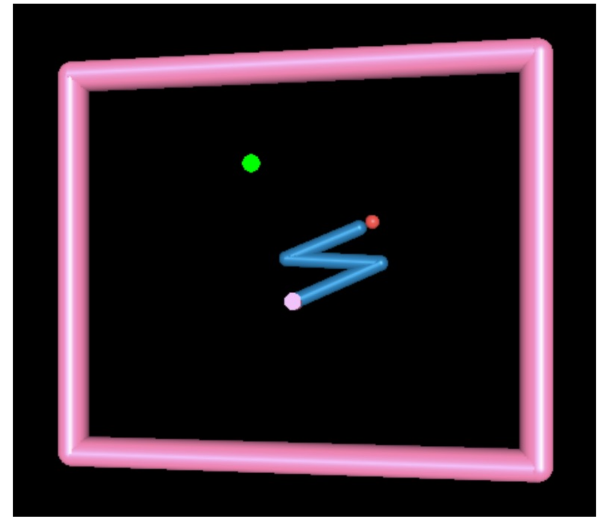
$$\boxed{g_x \triangleq \partial g(x,y)/\partial x}$$

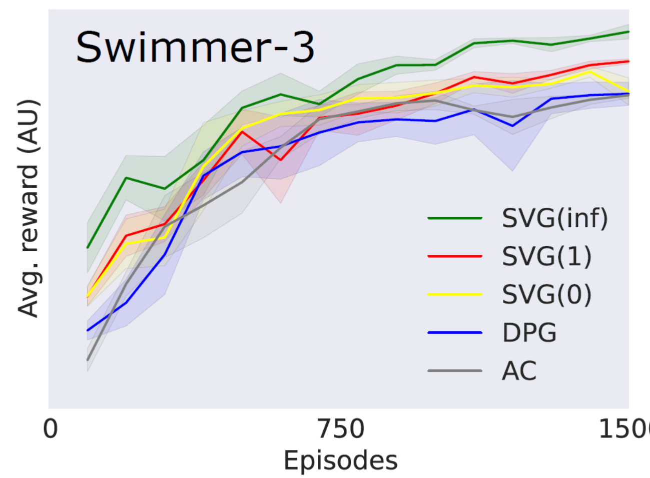
Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.

SVG Algorithm and Experiments

Algorithm 1 SVG(∞)

Given empty experience database $\mathcal{D}$
for trajectory $= 0$ **to** ∞ **do**
  for $t = 0$ **to** T **do**
    Apply control $\mathbf{a} = \pi(\mathbf{s}, \eta; \theta)$, $\eta \sim \rho(\eta)$
    Insert $(\mathbf{s}, \mathbf{a}, r, \mathbf{s}')$ into $\mathcal{D}$
  end for
  Train generative model $\hat{\mathbf{f}}$ using $\mathcal{D}$
  $v'_{\mathbf{s}} = 0$ (finite-horizon)
  $v'_{\theta} = 0$ (finite-horizon)
  for $t = T$ **down to** 0 **do**
    Infer $\xi|(\mathbf{s}, \mathbf{a}, \mathbf{s}')$ and $\eta|(\mathbf{s}, \mathbf{a})$
    $v_{\theta} = [r_{\mathbf{a}}\pi_{\theta} + \gamma(v'_{\mathbf{s}'}\hat{\mathbf{f}}_{\mathbf{a}}\pi_{\theta} + v'_{\theta})]\big|_{\eta,\xi}$
    $v_{\mathbf{s}} = [r_{\mathbf{s}} + r_{\mathbf{a}}\pi_{\mathbf{s}} + \gamma v'_{\mathbf{s}'}(\hat{\mathbf{f}}_{\mathbf{s}} + \hat{\mathbf{f}}_{\mathbf{a}}\pi_{\mathbf{s}})]\big|_{\eta,\xi}$
  end for
  Apply gradient-based update using v^0_{θ}
end for





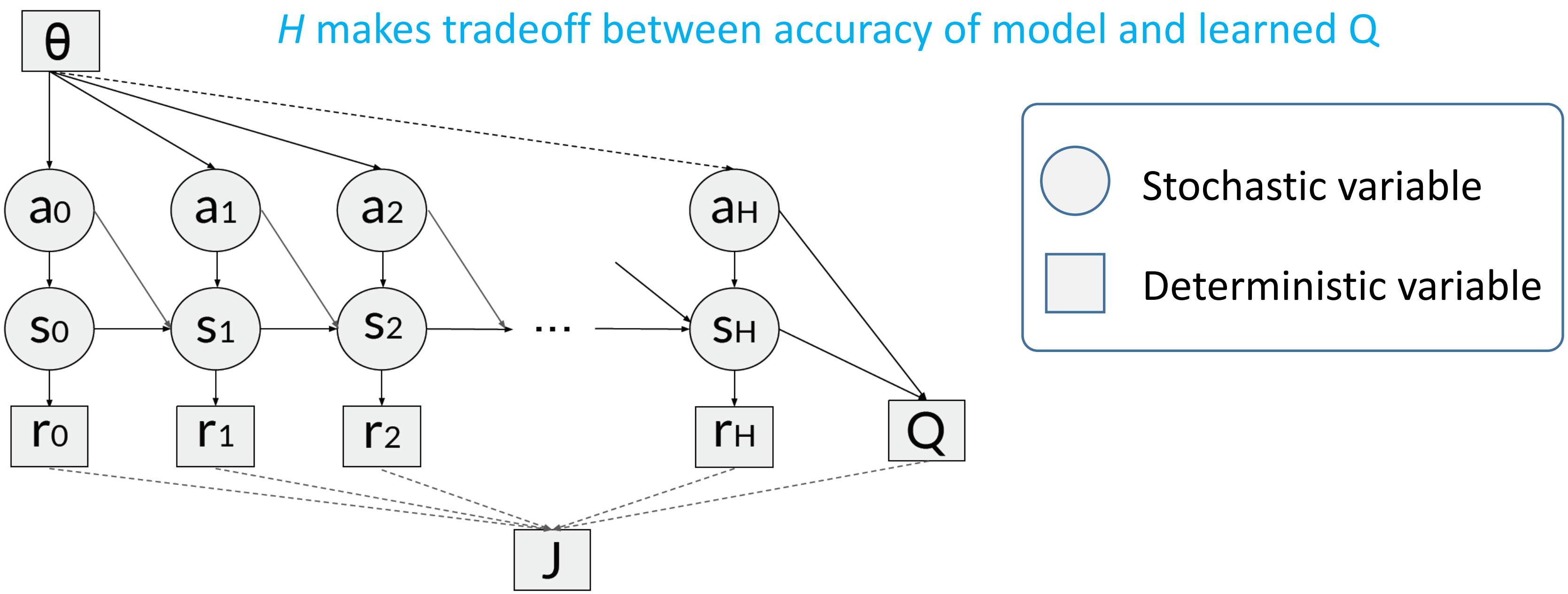
Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.

Model-Augmented Actor Critic

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

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

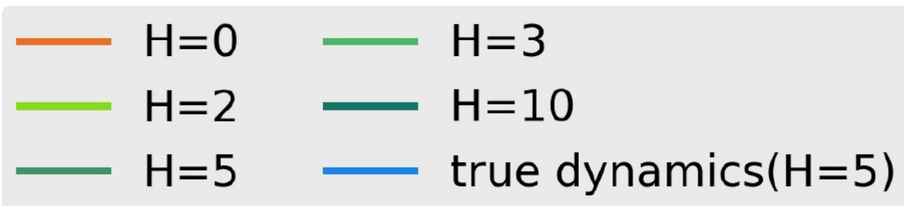
H makes tradeoff between accuracy of model and learned Q

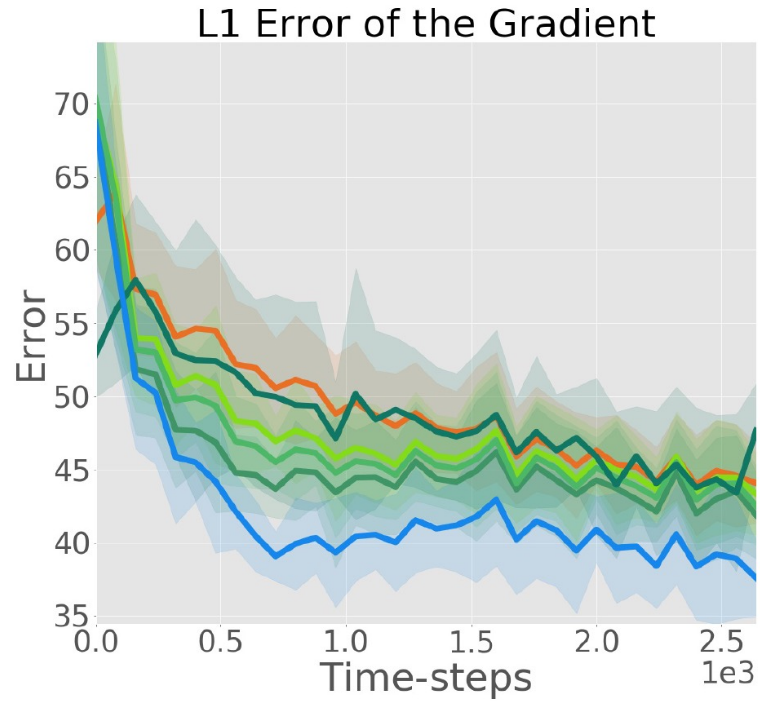


Ignasi Clavera, Yao Fu, Pieter Abbeel. Model-Augmented Actor Critic: Backpropagation through paths. ICLR 2020.

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:*

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

Ignasi Clavera, Yao Fu, Pieter Abbeel. Model-Augmented Actor Critic: Backpropagation through paths. ICLR 2020.

Theoretic Bounds

One-step gradient approximation error

Lemma 4.2 (Total Variation Bound). *Under the assumptions of the Lemma 4.1, let $\boldsymbol{\theta} = \boldsymbol{\theta}_o + \alpha \nabla_{\boldsymbol{\theta}} J_\pi$ be the parameters resulting from taking a gradient step on the exact objective, and $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_o + \alpha \nabla_{\boldsymbol{\theta}} \hat{J}_\pi$ 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 $\boldsymbol{\theta}'$ and $\hat{\boldsymbol{\theta}}$ as defined in Lemma 4.2, and assuming that the reward is bounded by $r_{\max}$. Then the average return of the $\pi_{\hat{\boldsymbol{\theta}}}$ satisfies*

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

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

Ignasi Clavera, Yao Fu, Pieter Abbeel. Model-Augmented Actor Critic: Backpropagation through paths. ICLR 2020.

MAAC Algorithm

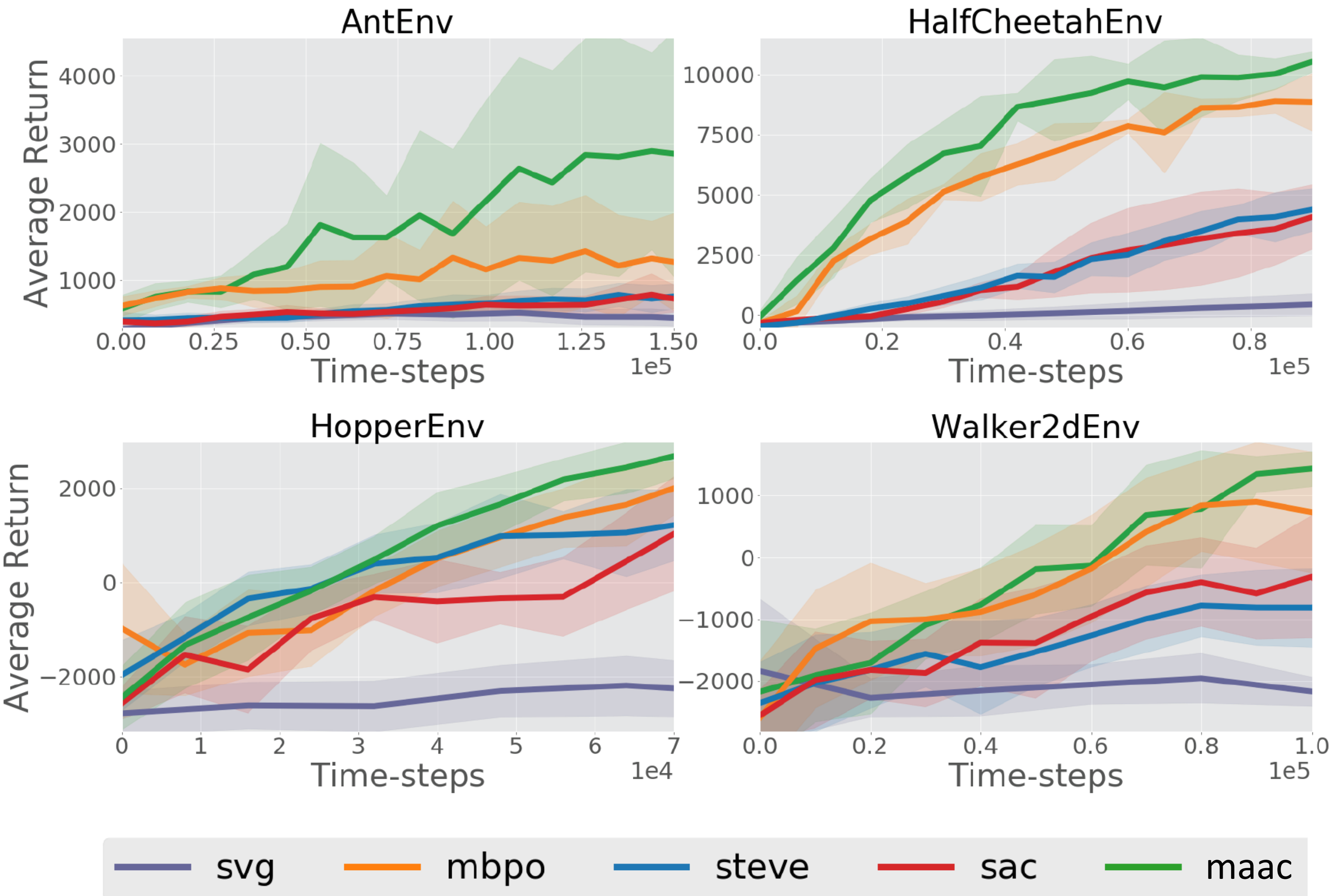
Algorithm 1 MAAC

Initialize the policy $\pi_{\boldsymbol{\theta}}$, model $\hat{f}_{\boldsymbol{\phi}}$, $\hat{Q}_{\boldsymbol{\psi}}$, $\mathcal{D}_{\text{env}} \leftarrow \emptyset$, and the model dataset $\mathcal{D}_{\text{model}} \leftarrow \emptyset$
repeat
    Sample trajectories from the real environment with policy π_θ. Add them to $\mathcal{D}_{\text{env}}$.
    for $i = 1 \ldots G_1$ **do**
        $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \beta_f \nabla_{\boldsymbol{\phi}} J_f(\boldsymbol{\phi})$ using data from $\mathcal{D}_{\text{env}}$.
    end for
    Sample trajectories $\mathcal{T}$ from $\hat{f}_{\boldsymbol{\phi}}$. Add them to $\mathcal{D}_{\text{model}}$.
    $\mathcal{D} \leftarrow \mathcal{D}_{\text{model}} \cup \mathcal{D}_{\text{env}}$
    for $i = 1 \ldots G_2$ **do**
        Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta_\pi \nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta})$ using data from $\mathcal{D}$
        Update $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \beta_Q \nabla_{\boldsymbol{\psi}} J_Q(\boldsymbol{\psi})$ using data from $\mathcal{D}$
    end for
until the policy performs well in the real environment
return Optimal parameters $\boldsymbol{\theta}^*$

- Model learning: PILCO - ensemble of GPs $\{\hat{f}_{\boldsymbol{\phi}_1}, \ldots, \hat{f}_{\boldsymbol{\phi}_M}\}$
- Policy Optimization $$J_\pi(\boldsymbol{\theta}) = \mathbb{E}\left[\sum_{t=0}^{H-1} \gamma^t r(\hat{s}_t) + \gamma^H Q_{\boldsymbol{\psi}}(\hat{s}_H, a_H)\right] + \beta \mathcal{H}(\pi_{\boldsymbol{\theta}})$$
- Q-function Learning $$J_Q(\boldsymbol{\psi}) = \mathbb{E}[(Q_{\boldsymbol{\psi}}(s_t, a_t) - (r(s_t, a_t) + \gamma Q_{\boldsymbol{\psi}}(s_{t+1}, a_{t+1})))^2]$$

Deisenroth, Marc, and Carl E. Rasmussen. "PILCO: A model-based and data-efficient approach to policy search." ICML 2011.

Experiments



Ignasi Clavera, Yao Fu, Pieter Abbeel. Model-Augmented Actor Critic: Backpropagation through paths. ICLR 2020.

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.