Reinforcement Learning China Summer School

RLChina 2020

Model-based Reinforcement Learning

Prof. Weinan Zhang

John Hopcroft Center, Shanghai Jiao Tong University

July 30, 2020

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

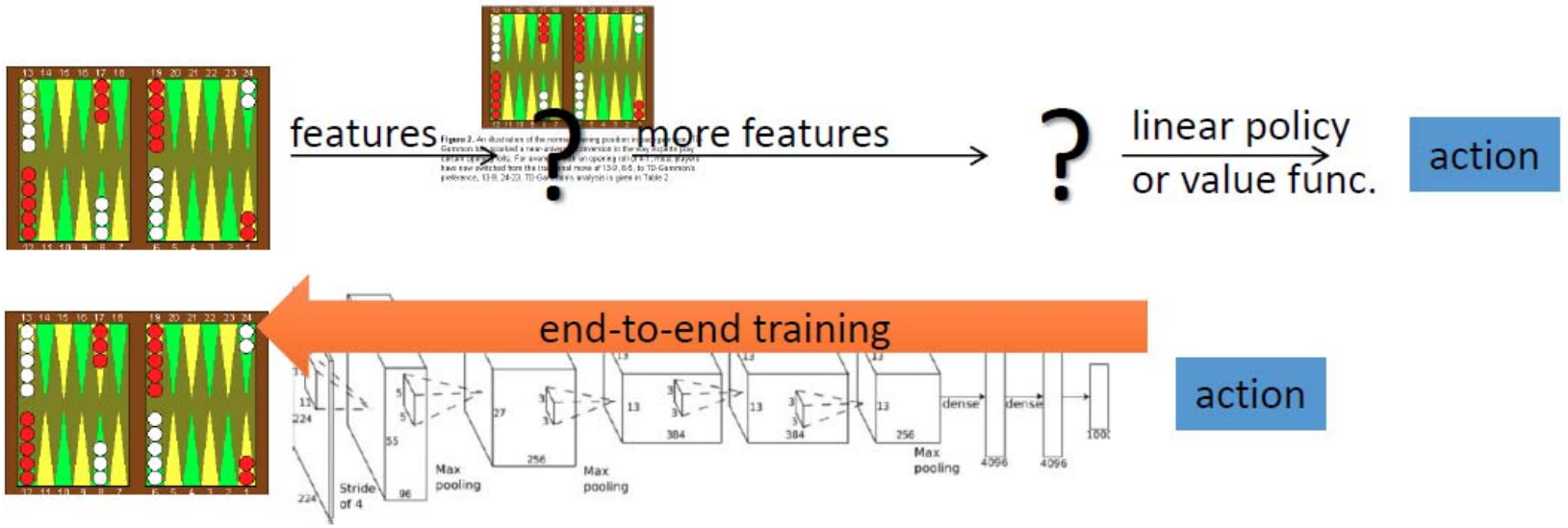
JOHN HOPCROFT CENTER FOR COMPUTER SCIENCE

APEX DATA & KNOWLEDGE MANAGEMENT LAB

A Perspective of

Overall Pathway of DRL

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



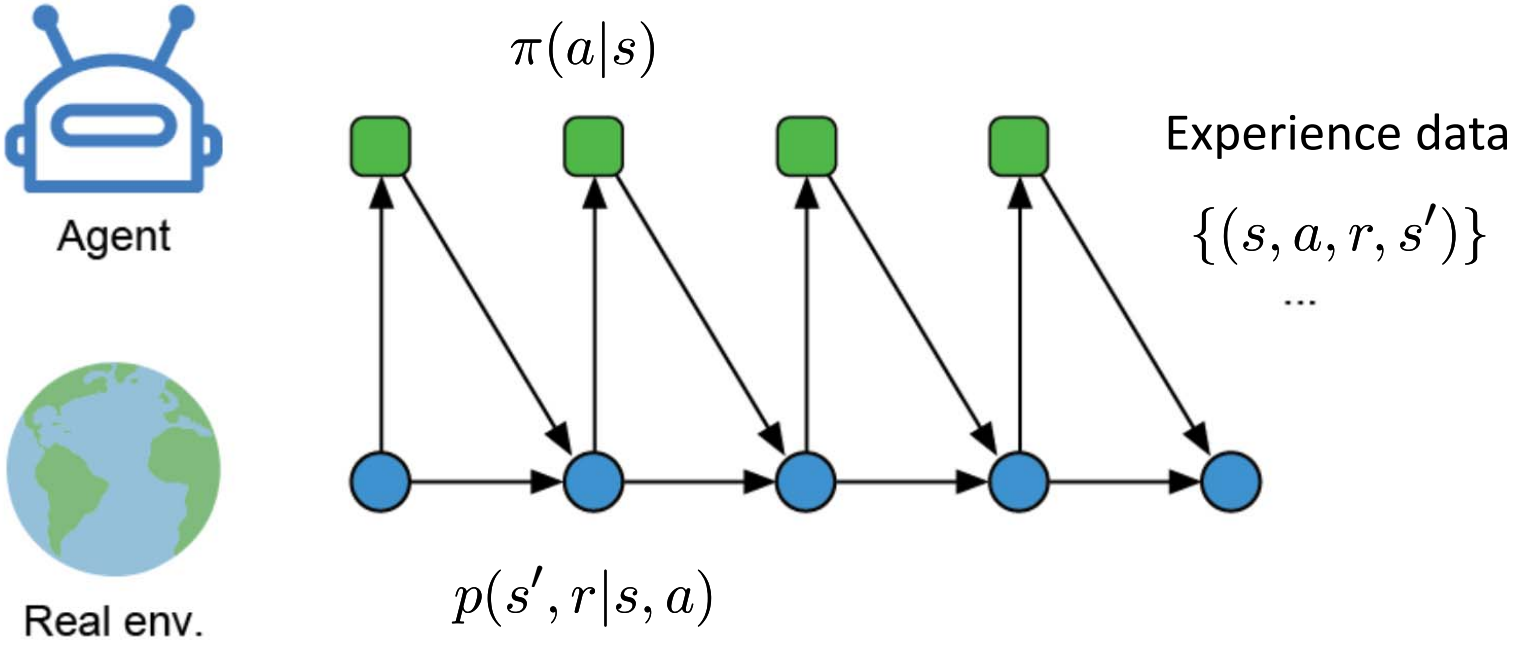
Slide from Sergey Levine. http://rail.eecs.berkeley.edu/deeprlcourse/static/slides/lec-1.pdf

A Perspective of

Overall Pathway of DRL

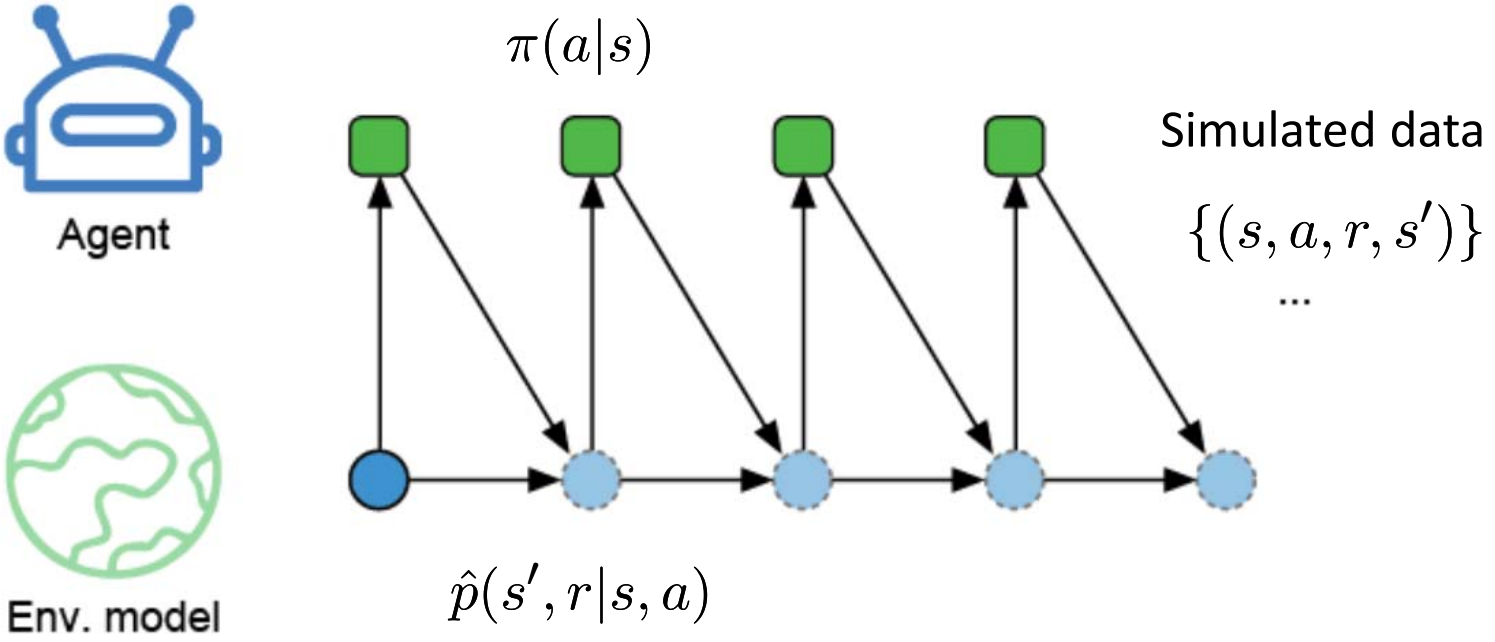
- Deep reinforcement learning gets appealing success
  - Atari, AlphaGo, DOTA 2, AlphaStar
- But DRL has very low data efficiency
  - Trial-and-error learning for deep networks
- A recent popular direction is model-based RL
  - Build a model $p(s', r|s, a)$
  - Based on the model to train the policy
  - 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 (1)

- 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

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

Model as a Whitebox

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

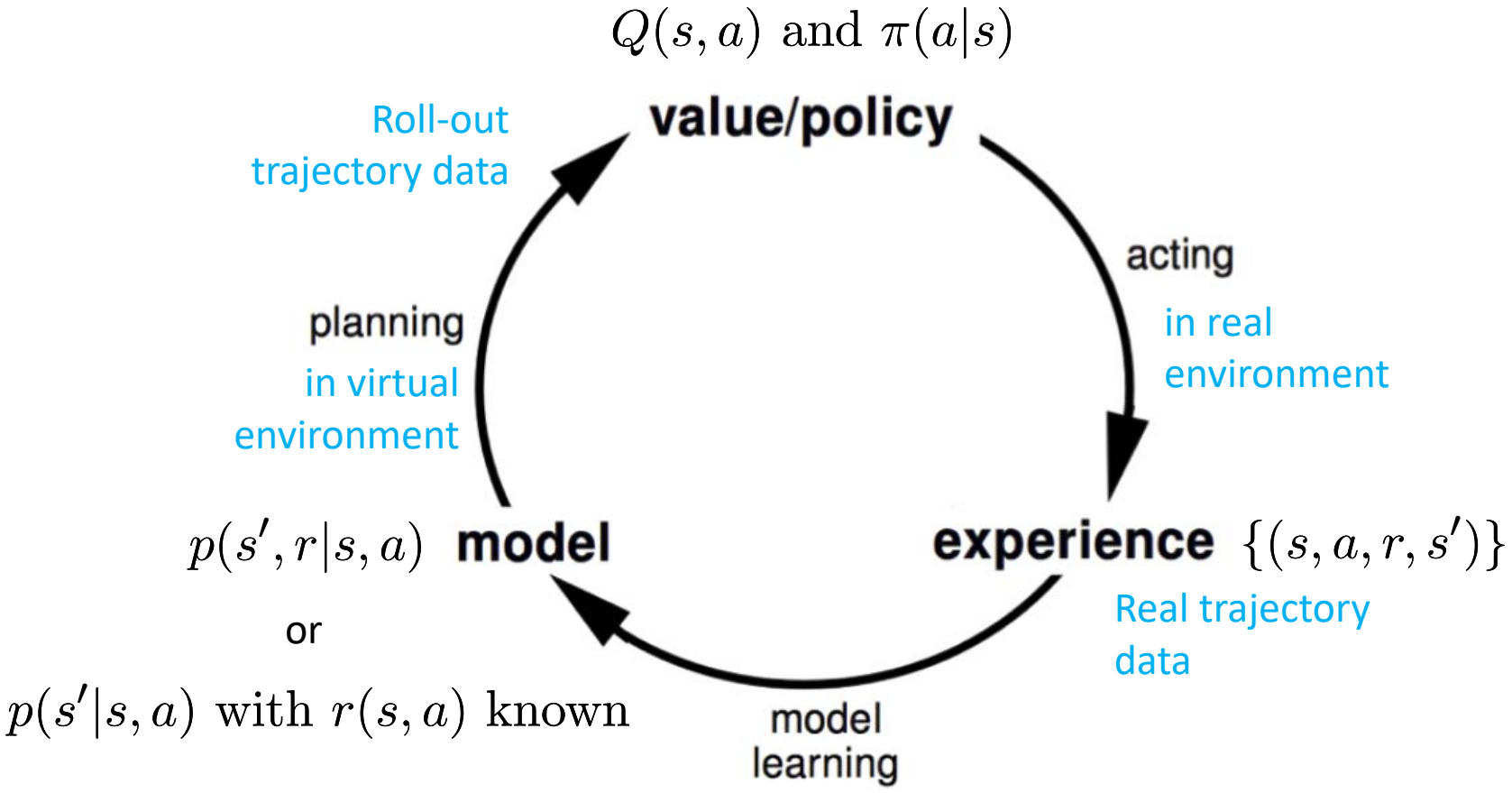
$$\rightarrow \frac{\partial V(s')}{\partial s'} \frac{\partial s'}{\partial a} \frac{\partial a}{\partial \theta}\Big|_{s' \sim f_\phi(s,a)}$$

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

Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS, PETS, POPLIN
3. Theoretic bounds and methods: SLBO, MBPO & BMPO
4. Backpropagation through paths: SVG and MAAC

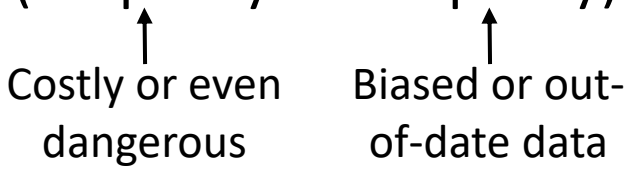
Model-based RL



Annotations based on Rich Sutton's figure

Model-free RL v.s. Model-based RL (2)

- Model-based RL
  - Can efficiently learn model by supervised learning methods
  - Can reason about model uncertainty
  - Empirically less interactions with real environment
  - First learn a model, then construct a value function
    - two sources of approximation error
- Model-free RL
  - No need to learn the model
  - Learn value function (and/or policy) from real experience (on-policy or off-policy)



Q-Planning

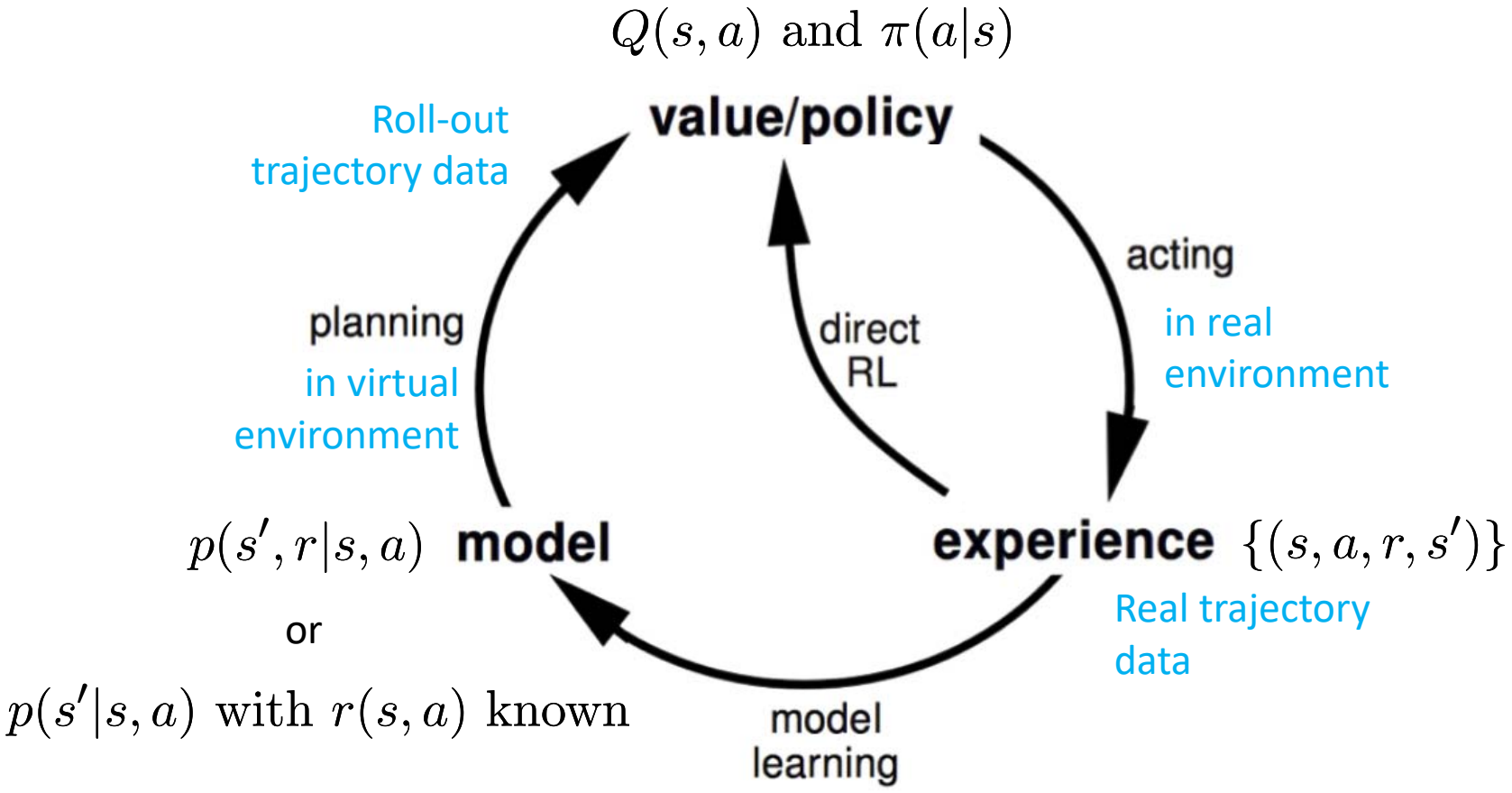
- Random-sample one-step tabular Q-planning
  - First, learn a model $p(s',r|s,a)$ from experience data
  - Then perform one-step sampling by the model to learn the Q function

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

- 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) $A \leftarrow \varepsilon$-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

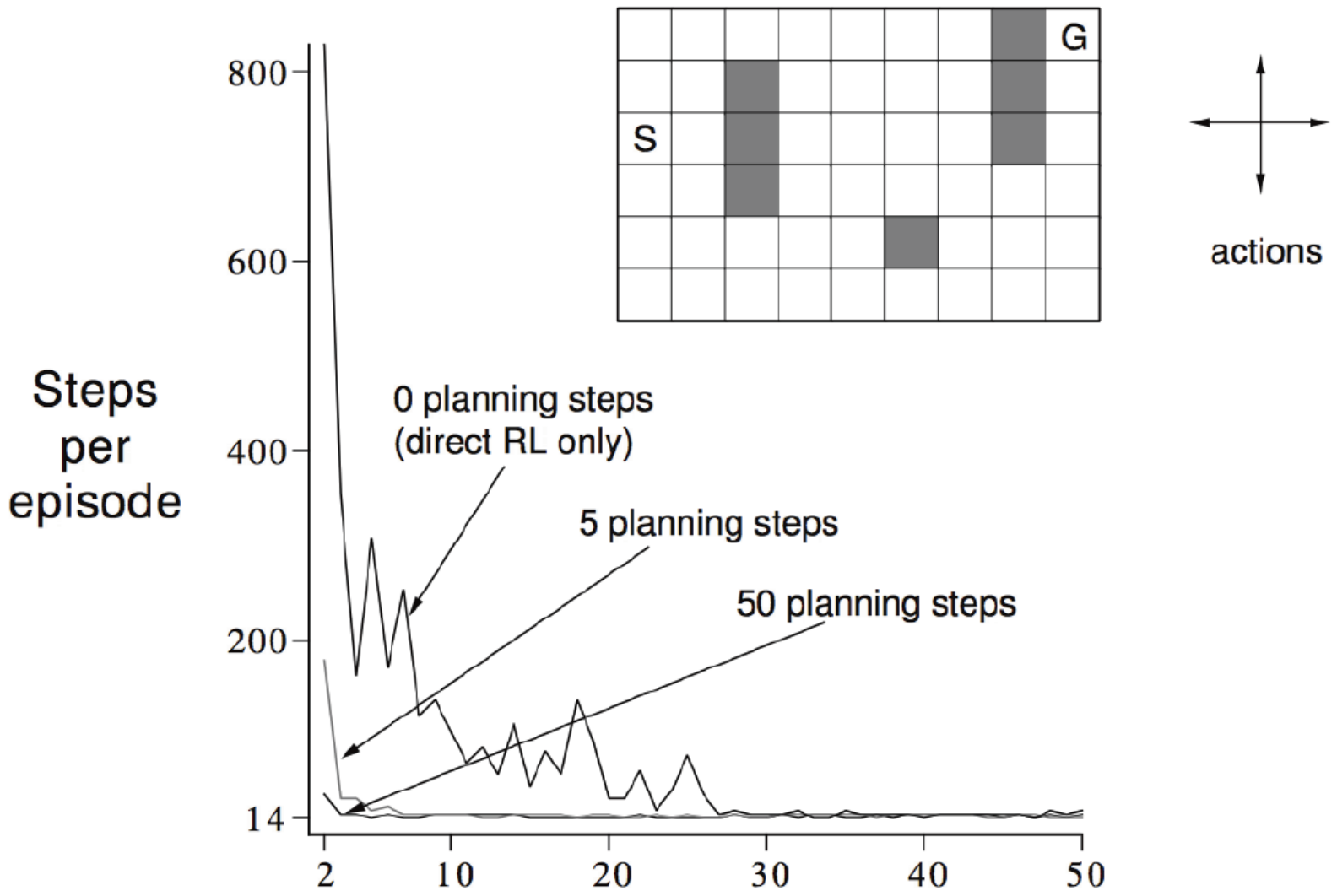
$A \leftarrow$ random action previously taken in S

$R, S' \leftarrow Model(S,A)$

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

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 Model-based RL

- Does the model really help improve the data efficiency?
- Inevitably, the model is to-some-extent inaccurate. When to trust the model?
- How to properly leverage the model to better train our policy?

Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS, PETS, POPLIN
3. Theoretic bounds and methods: SLBO, MBPO & BMPO
4. Backpropagation through paths: SVG and MAAC

Shooting Methods

- Based on a model $p(s',r|s,a)$, we can sample N trajectories

$$\{[s_1^{(k)}, a_1, r_1^{(k)}, s_2^{(k)}, a_2, r_2^{(k)}, \ldots, s_T^{(k)}, a_T, r_T^{(k)}]\}_{k=1,\ldots,N}$$

  given an action sequence to take

$$[a_1, a_2, \ldots, a_T]$$

- For the current state s and each candidate action a, build the action sequence as

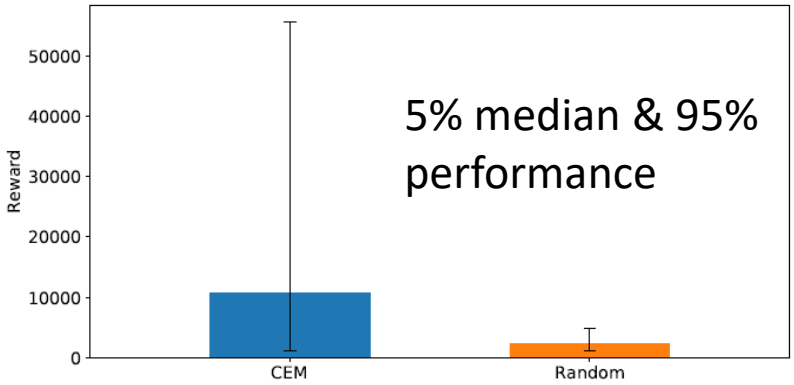
$$[a, a_1, a_2, \ldots, a_T]$$

  to sample N trajectories from the model.

$$\hat{Q}(s,a) = \frac{1}{N}\sum_{k=1}^{N}\sum_{t=0}^{T}\gamma^t r_t^{(k)} \qquad \pi(s) = \arg\max_a \hat{Q}(s,a)$$

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.



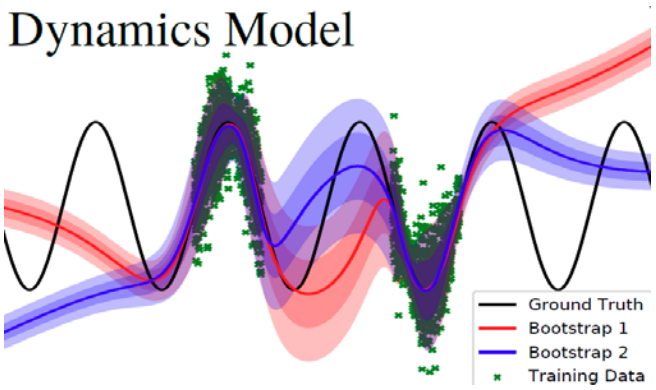
Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." *NIPS* 2018.

PETS: Probabilistic Ensembles with Trajectory Sampling

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

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

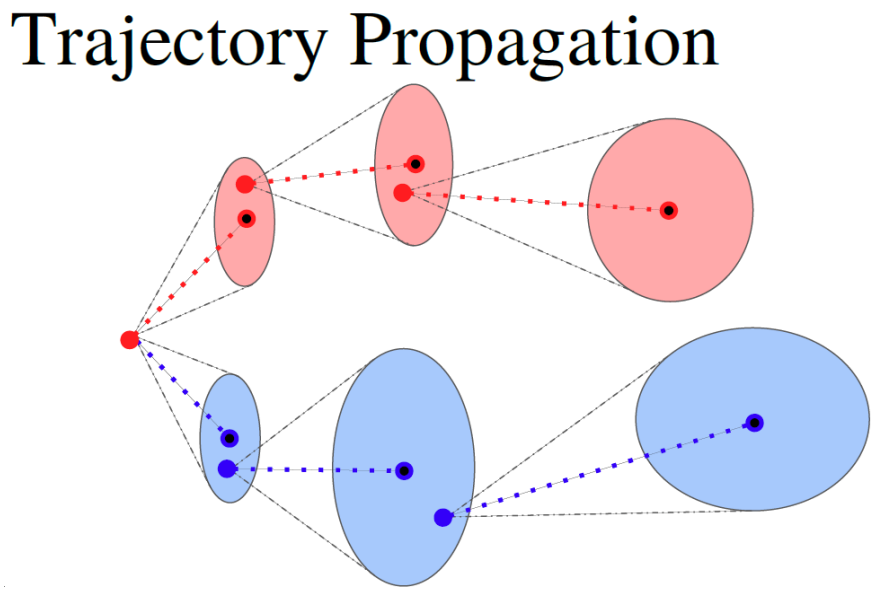
- Probabilistic ensemble (PE) dynamics model is shown as an ensemble of two bootstraps
  - Bootstrap disagreement far from data captures epistemic uncertainty: our subjective uncertainty due to a lack of data (model variance)
  - Each a probabilistic neural network that captures aleatoric uncertainty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

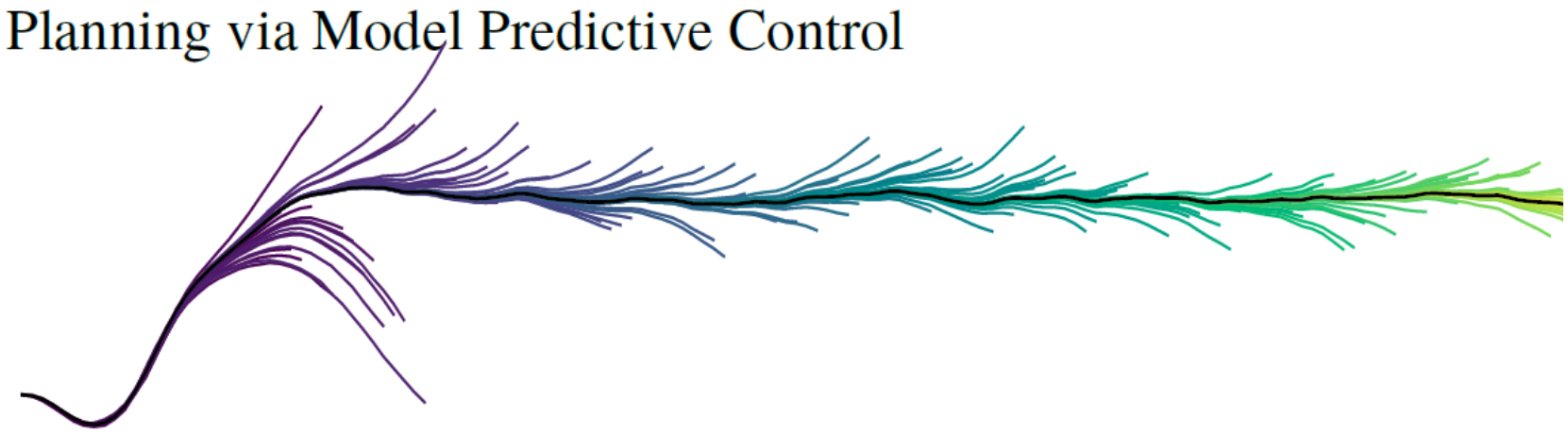


Chua, Kurtland, et al. "Deep reinforcement learning in a handful of trials using probabilistic dynamics models." *NIPS* 2018.

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*':

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

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

POPLIN: Model-based Policy Planning

- PETS uses CEM to maintain a distribution of actions yielding high reward at each time step
- Can we better choose the action sequence?
- POPLIN: maintain a policy to sample actions given the current simulated state

Wang, Tingwu, and Jimmy Ba. "Exploring Model-based Planning with Policy Networks." *arXiv preprint arXiv:1906.08649* (2019).

POPLIN: Model-based Policy Planning

- POPLIN: maintain a policy to sample actions given the current simulated state

$$\mathcal{R}(s_i, \mathbf{a}_i) = \mathbb{E}\left[\sum_{t=i}^{i+\tau} r(s_t, a_t)\right], \text{ where } s_{t+1} \sim f_\phi(s_{t+1}|s_t, a_t)$$

- Action level

$$\mathcal{R}(s_i, \boldsymbol{\delta}_i) = \mathbb{E}\left[\sum_{t=i}^{i+\tau} r(s_t, \hat{a}_t + \delta_t)\right]$$

- Policy parameter level

$$\mathcal{R}(s_i, \boldsymbol{\omega}_i) = \mathbb{E}\left[\sum_{t=i}^{i+\tau} r\left(s_t, \pi_{\theta+\omega_t}(s_t)\right)\right]$$

Wang, Tingwu, and Jimmy Ba. "Exploring Model-based Planning with Policy Networks." *arXiv preprint arXiv:1906.08649* (2019).

POPLIN Policy Distillation Schemes

- With action sequences ranked & selected by total reward
- Behavior cloning (BC) for POPLIN-A and -P

$$\min_{\theta} \mathbb{E}_{s,\, a\in\mathcal{D}} ||\pi_\theta(s) - a||^2$$

- Generative adversarial network training (GAN) for POPLIN-P

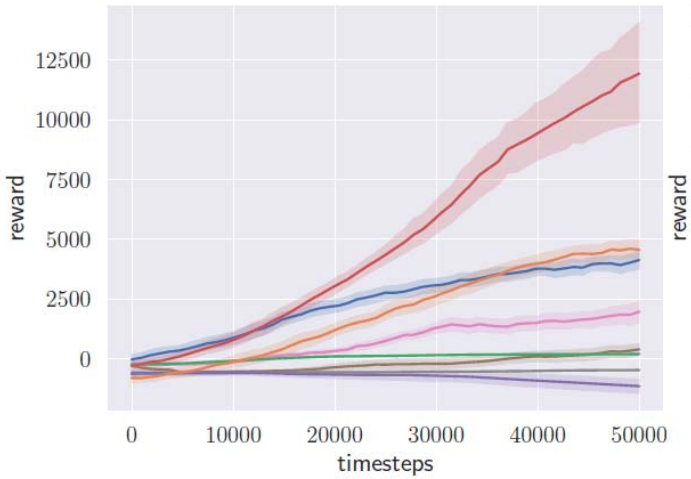
$$\min_{\pi_\theta} \max_{\psi} \mathbb{E}_{s,\, a\in\mathcal{D}} \log(D_\psi(s,a)) + \mathbb{E}_{s\in\mathcal{D},\, z\sim\mathcal{N}(\mathbf{0},\sigma_0\boldsymbol{I})} \log(1 - D_\psi(s, \pi_{\theta+z}(s)))$$

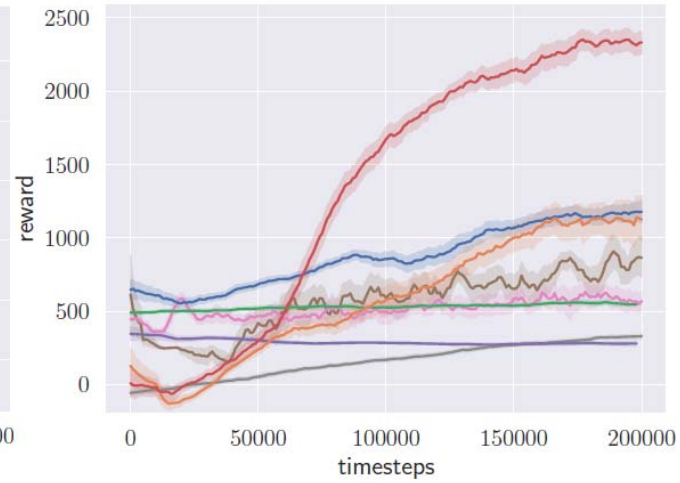
- Setting parameter average (AVG) for POPLIN-P

$$\theta = \theta + \frac{1}{|\mathcal{D}|} \sum_{\omega\in\mathcal{D}} \omega$$

POPLIN Experiments

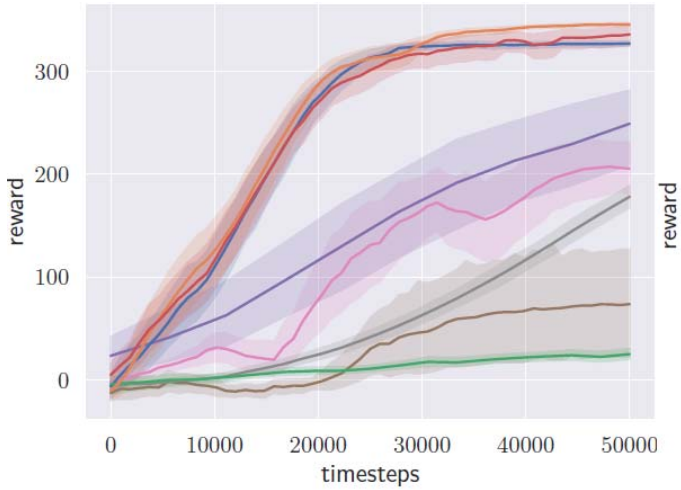
Cheetah

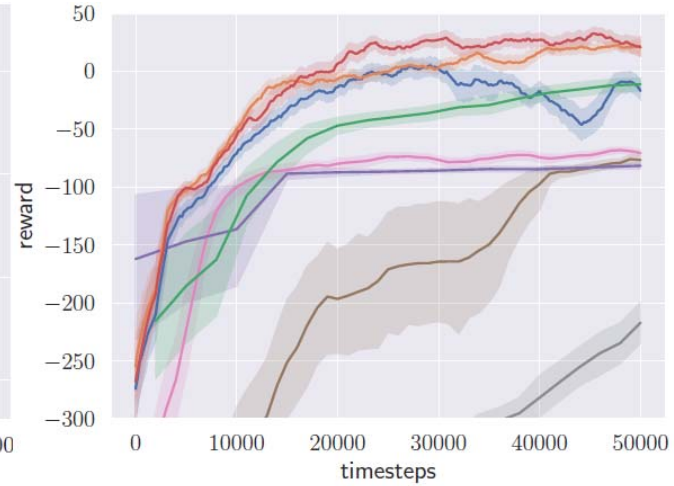




Ant

Swimmer



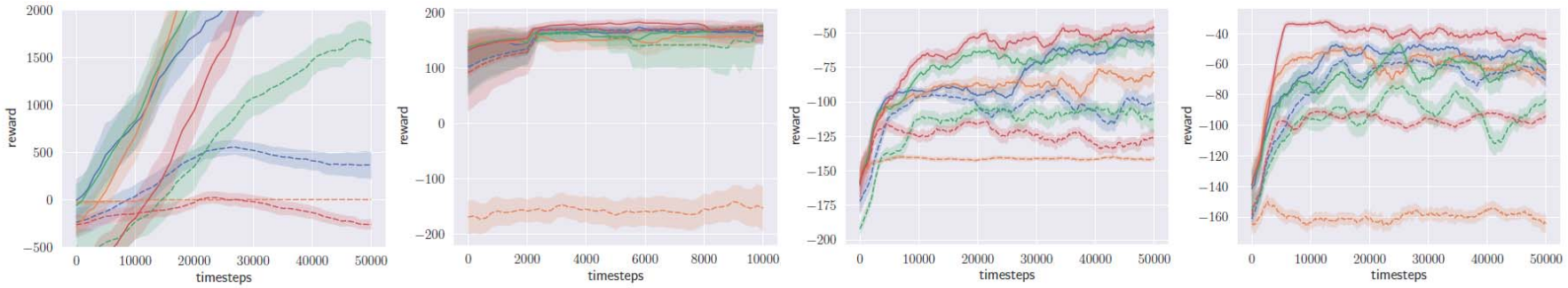


Acrobot

POPLIN-P
POPLIN-A
SAC
METRPO
RS
TD3
PETS
PPO

POPLIN Experiments

	Cheetah	Ant	Hopper	Swimmer	Cheetah-v0	Walker2d
POPLIN-P (ours)	**12227.9 ± 5652.8**	**2330.1 ± 320.9**	**2055.2 ± 613.8**	**334.4 ± 34.2**	**4235.0 ± 1133.0**	**597.0 ± 478.8**
POPLIN-A (ours)	4651.1 ± 1088.5	1148.4 ± 438.3	202.5 ± 962.5	**344.9 ± 7.1**	1562.8 ± 1136.7	-105.0 ± 249.8
PETS [5]	4204.5 ± 789.0	1165.5 ± 226.9	114.9 ± 621.0	326.2 ± 12.6	2288.4 ± 1019.0	282.5 ± 501.6
METRPO [20]	-744.8 ± 707.1	282.2 ± 18.0	1272.5 ± 500.9	225.5 ± 104.6	2283.7 ± 900.4	-1609.3 ± 657.5
TD3 [9]	218.9 ± 593.3	870.1 ± 283.8	1816.6 ± 994.8	72.1 ± 130.9	3015.7 ± 969.8	-516.4 ± 812.2
SAC [14]	1745.9 ± 839.2	548.1 ± 146.6	788.3 ± 738.2	204.6 ± 69.3	3459.8 ± 1326.6	164.5 ± 1318.6
Training Time-step	50000	200000	200000	50000	200000	200000
	Reacher3D	Pusher	Pendulum	InvertedPendulum	Acrobot	Cartpole
POPLIN-P (ours)	**-29.0** ± 25.2	**-55.8 ± 23.1**	167.9 ± 45.9	**-0.0 ± 0.0**	**23.2 ± 27.2**	**200.8 ± 0.3**
POPLIN-A (ours)	**-27.7** ± 25.2	**-56.0 ± 24.3**	**178.3 ± 19.3**	**-0.0 ± 0.0**	20.5 ± 20.1	**200.6 ± 1.3**
PETS [5]	-47.7 ± 43.6	**-52.7 ± 23.5**	155.7 ± 79.3	-29.5 ± 37.8	-18.4 ± 46.3	199.6 ± 4.6
METRPO [20]	-43.5 ± 3.7	-98.5 ± 12.6	174.8 ± 6.2	-29.3 ± 29.5	-78.7 ± 5.0	138.5 ± 63.2
TD3 [9]	-331.6 ± 134.6	-216.4 ± 39.6	168.6 ± 12.7	-102.9 ± 101.0	-76.5 ± 10.2	-409.2 ± 928.8
SAC [14]	-161.6 ± 43.7	-227.6 ± 42.2	159.5 ± 12.1	-0.2 ± 0.1	-69.4 ± 7.0	195.5 ± 8.7
Training Time-step	50000	50000	50000	50000	50000	50000



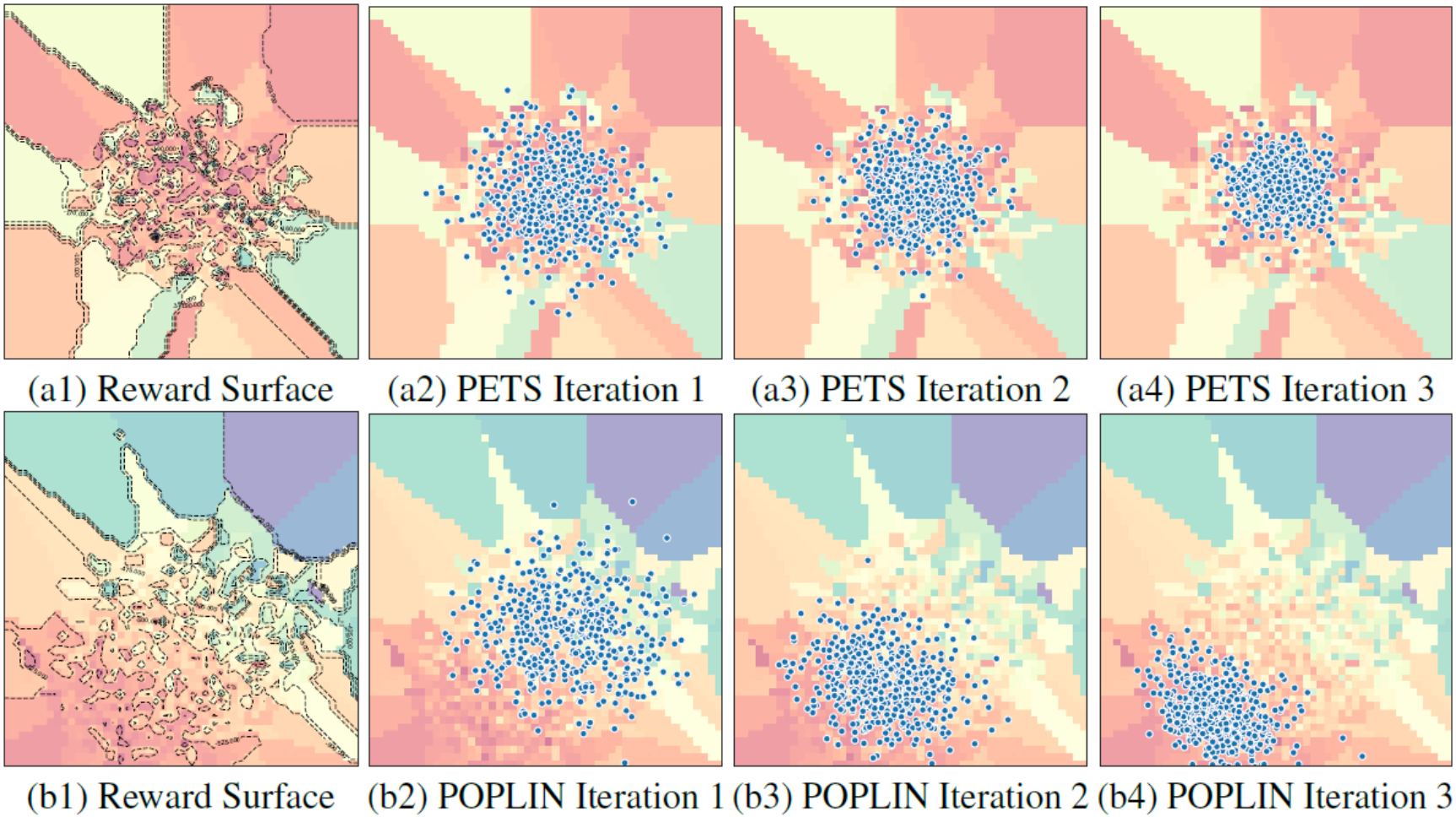
(a) Cheetah (b) Pendulum (c) Pusher (e) Reacher3D

MPC Control — POPLIN-A-BC — POPLIN-P-GAN — POPLIN-P-AVG — POPLIN-P-BC

Policy Control — POPLIN-A-BC — POPLIN-P-GAN — POPLIN-P-AVG — POPLIN-P-BC

POPLIN vs. PETS

Transform each planned candidate action trajectory with PCA into a 2D blue scatter. Red areas are with high reward. Blue points are actions taken.



Content

1. Introduction to MBRL from Dyna
2. Shooting methods: RS, PETS, POPLIN
3. Theoretic bounds and methods: SLBO, MBPO & BMPO
4. Backpropagation through paths: SVG and MAAC

Any Theoretic Bound for MBRL?

$$V^{\pi, M^\star} \geq V^{\pi, \widehat{M}} - D(\widehat{M}, \pi)$$

Value in real env

Value in the model

Discrepancy of model and real env when playing with π

- A set of practical requirement of theoretic analysis

$$V^{\pi, M^\star} \geq V^{\pi, \widehat{M}} - D_{\pi_{\text{ref}}, \delta}(\widehat{M}, \pi), \qquad \forall \pi \text{ s.t. } d(\pi, \pi_{\text{ref}}) \leq \delta \quad \text{(R1)}$$

$$\widehat{M} = M^\star \Longrightarrow D_{\pi_{\text{ref}}}(\widehat{M}, \pi) = 0, \quad \forall \pi, \pi_{\text{ref}} \quad \text{(R2)}$$

A very strong assumption

$$D_{\pi_{\text{ref}}}(\widehat{M}, \pi) \text{ is of the form } \mathop{\mathbb{E}}_{\tau \sim \pi_{\text{ref}}, M^\star} [f(\widehat{M}, \pi, \tau)] \quad \text{(R3)}$$

e.g. $D_{\pi_{\text{ref}}}(\widehat{M}, \pi) = L \cdot \mathop{\mathbb{E}}_{S_0, \ldots, S_t, \sim \pi_{\text{ref}}, M^\star} \left[\| \widehat{M}(S_t) - S_{t+1} \| \right]$

Luo, Yuping, et al. "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees." ICLR 2019.

Meta Algorithm of MBRL

Algorithm 1 Meta-Algorithm for Model-based RL

Inputs: Initial policy π_0. Discrepancy bound D and distance function d that satisfy equation (R1) and (R2).

For $k = 0$ to T:

$$\pi_{k+1}, M_{k+1} = \underset{\pi \in \Pi,\ M \in \mathcal{M}}{\operatorname{argmax}} \quad V^{\pi, M} - D_{\pi_k, \delta}(M, \pi) \tag{3.3}$$

$$\text{s.t. } d(\pi, \pi_k) \leq \delta \tag{3.4}$$

- Remarks
  1. Optimize the lower bound directly (with TRPO)
  2. Jointly optimizing *M* and π encourages the algorithm to choose the most optimistic model among those that can be used to accurately estimate the value function.
  3. Fixing *M* to optimize is π just model-free RL

Luo, Yuping, et al. "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees." ICLR 2019.

Convergence Theorem & Proof

Theorem 3.1. *Suppose that $M^\star \in \mathcal{M}$, that D and d satisfy equation (R1) and (R2), and the optimization problem in equation (3.3) is solvable at each iteration. Then, Algorithm 1 produces a sequence of policies $\pi_0, \ldots, \pi_T$ with monotonically increasing values:*

$$V^{\pi_0, M^\star} \leq V^{\pi_1, M^\star} \leq \cdots \leq V^{\pi_T, M^\star} \tag{3.5}$$

Moreover, as $k \to \infty$, the value $V^{\pi_k, M^\star}$ converges to some $V^{\bar{\pi}, M^\star}$, where $\bar{\pi}$ is a local maximum of $V^{\pi, M^\star}$ in domain Π.

Proof:

$$\pi_{k+1}, M_{k+1} = \underset{\pi \in \Pi,\ M \in \mathcal{M}}{\operatorname{argmax}} \quad V^{\pi, M} - D_{\pi_k, \delta}(M, \pi)$$
$$\text{s.t. } d(\pi, \pi_k) \leq \delta$$

$$V^{\pi_{k+1}, M^\star} \geq V^{\pi_{k+1}, M_{k+1}} - D_{\pi_k}(M_{k+1}, \pi_{k+1}) \quad \text{(by R1)}$$

$$\geq V^{\pi_k, M^\star} - D_{\pi_k}(M^\star, \pi_k) = V^{\pi_k, M^\star} \quad \text{(by R2)}$$

= 0

SLBO: Stochastic Lower Bound Optimization

Algorithm 2 Stochastic Lower Bound Optimization (SLBO)

Initialize model network parameters ϕ and policy network parameters θ
Initialize dataset $\mathcal{D} \leftarrow \emptyset$
for n_{outer} iterations **do**
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{$ collect n_{collect} samples from real environment using π_θ with noises $\}$
  for n_{inner} iterations **do** ▷ optimize (6.2) with stochastic alternating updates
    for n_{model} iterations **do**
      optimize (6.1) over ϕ with sampled data from $\mathcal{D}$ by one step of Adam
    for n_{policy} iterations **do**
      $\mathcal{D}' \leftarrow \{$ collect n_{trpo} samples using $\widehat{M}_\phi$ as dynamics $\}$
      optimize π_θ by running TRPO on $\mathcal{D}'$

- Model optimization loss

$$\mathcal{L}_\phi^{(H)}((s_{t:t+h}, a_{t:t+h}); \phi) = \frac{1}{H}\sum_{i=1}^{H} \|(\hat{s}_{t+i} - \hat{s}_{t+i-1}) - (s_{t+i} - s_{t+i-1})\|_2.$$

- Model & policy optimization target

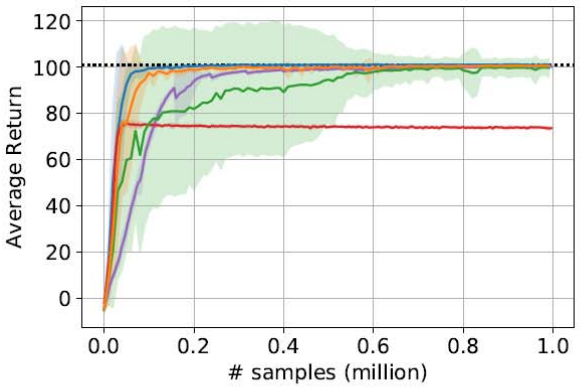
sg: stop gradient

$$\max_{\phi,\theta} \; V^{\pi_\theta, \text{sg}(\widehat{M}_\phi)} - \lambda \mathop{\mathbb{E}}_{(s_{t:t+h}, a_{t:t+h}) \sim \pi_k, M^\star} \left[\mathcal{L}_\phi^{(H)}((s_{t:t+h}, a_{t:t+h}); \phi)\right]$$

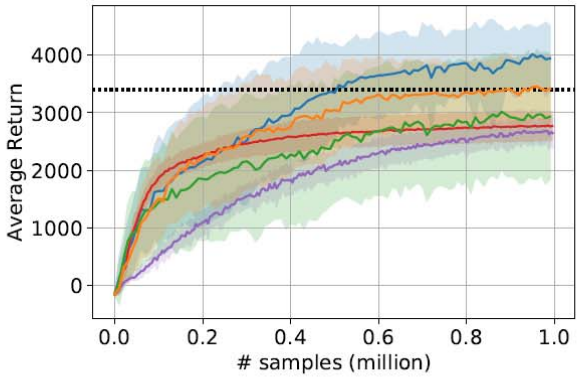
Luo, Yuping, et al. "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees." ICLR 2019.

SLBO Experiments

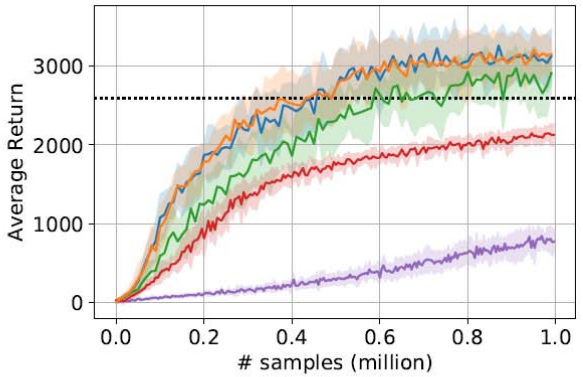
All envs with maximum horizon of 500



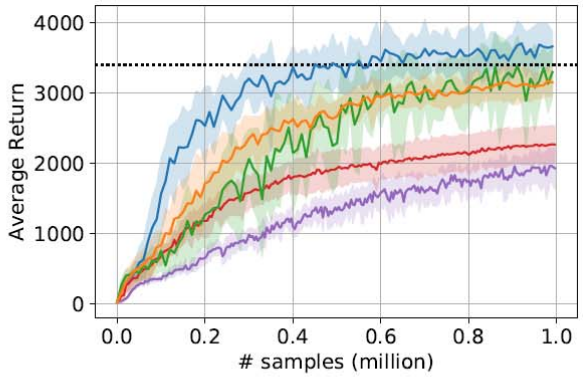
(a) Swimmer



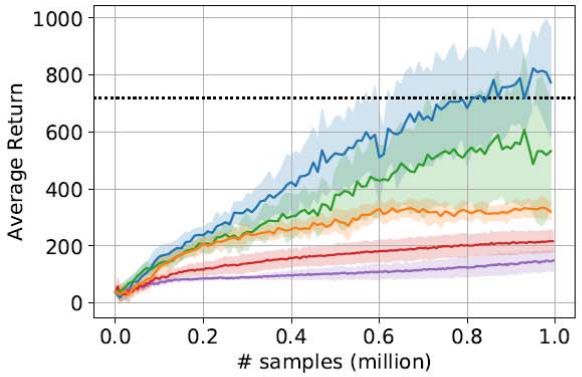
(b) Half Cheetah



(c) Ant



(d) Walker



(e) Humanoid

SLBO SLBO-MSE MB-TRPO SAC MF-TRPO

Luo, Yuping, et al. "Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees." ICLR 2019.

Problems of SLBO

- All environments have restricted 500 steps horizon
- Use the model to roll-out whole trajectories from the start state
- Practically, we may not roll-out so long horizon due to the compounding error of the model
  - Particularly when the model is inaccurate
  - We need to know whether and to-what-extent the model is inaccurate
  - And decide how to adaptively perform roll-out

Bound based on Model & Policy Error

Theorem 4.1. *Let the expected TV-distance between two transition distributions be bounded at each timestep by ϵ_m and the policy divergence be bounded by ϵ_π. Then the true returns and model returns of the policy are bounded as:*

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

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

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

based on old policy sampled data

- The discrepancy of policy value in real environment and environment model is linearly proportional to the model error ϵ_m

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

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

Theorem 4.2. *Given returns* $\eta^{\text{branch}}[\pi]$ *from the* k*-branched rollout method,*

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

- This lower bound is maximized when *k* = 0, corresponding to not using the model at all.

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

Bound based on Model & Policy Error

Theorem 4.2. *Given returns* $\eta^{\text{branch}}[\pi]$ *from the* k*-branched rollout method,*

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

- With a new model error (first-order) approximation

$$\hat{\epsilon}_{m'}(\epsilon_\pi) \approx \epsilon_m + \epsilon_\pi \frac{\mathrm{d}\epsilon_{m'}}{\mathrm{d}\epsilon_\pi} \qquad \epsilon_{m'} = \max_t \mathbb{E}_{s\sim\pi_t}[D_{TV}(p(s', r|s, a)\|p_\theta(s', r|s, a))]$$

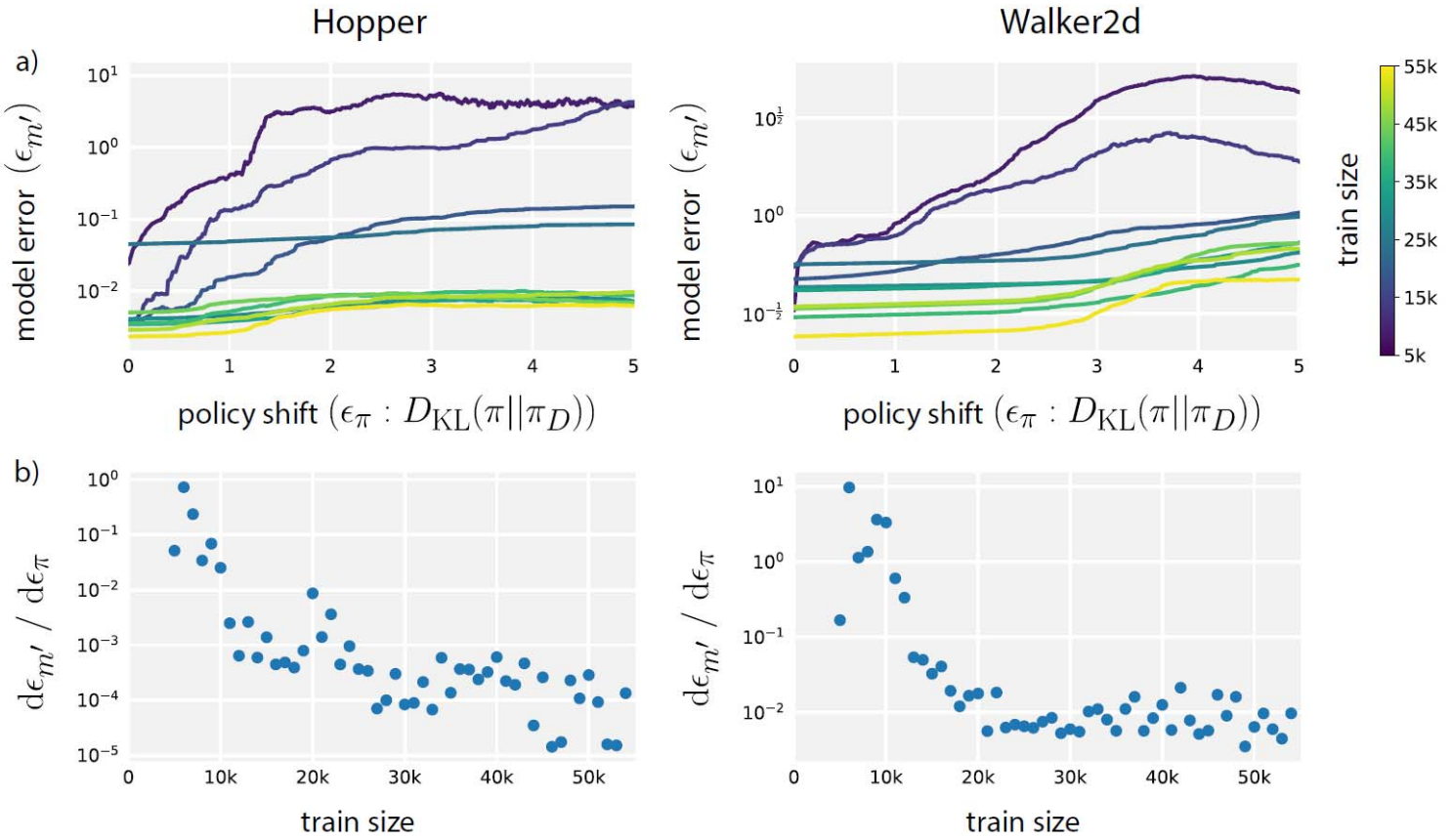
- The bound is rewritten as

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

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

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

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



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

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

MBPO Algorithm

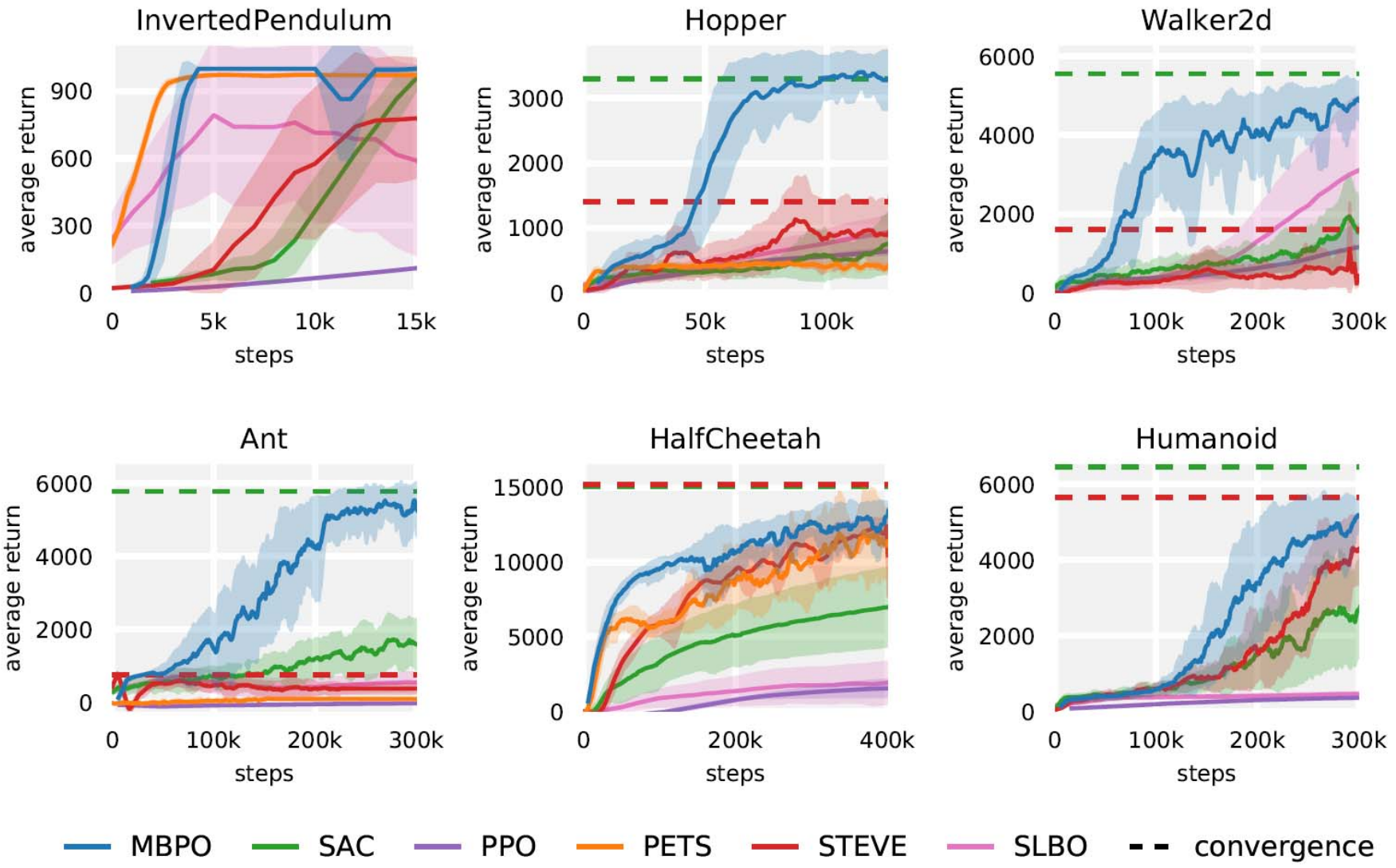
Algorithm 2 Model-Based Policy Optimization with Deep Reinforcement Learning

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

MBPO Experiments

1000-step horizon



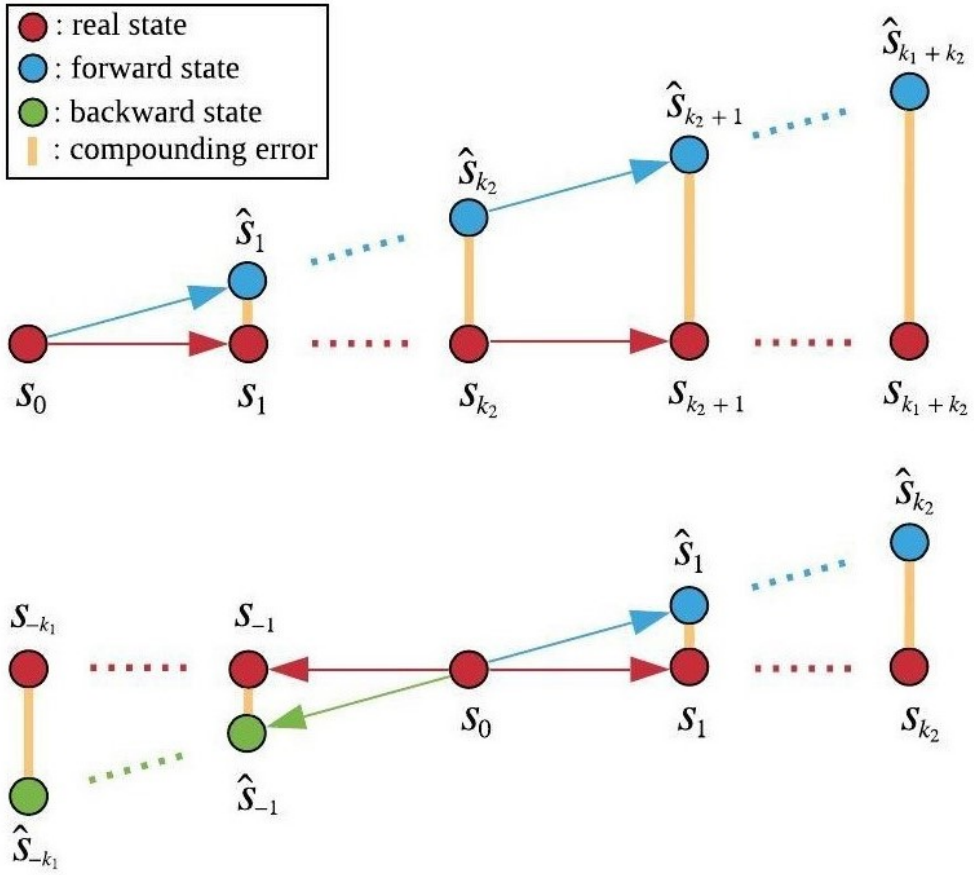
One Step Further based on MBPO

- Idea: Bidirectional models for branched rollout to reduce compounding error

- Paper: Hang Lai, Jian Shen, Weinan Zhang, Yong Yu. Bidirectional Model-based Policy Optimization. ICML 2020.
- https://arxiv.org/abs/2007.01995

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

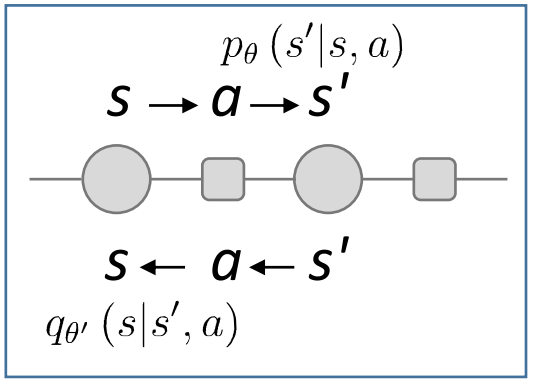
bidirectional model

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:



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

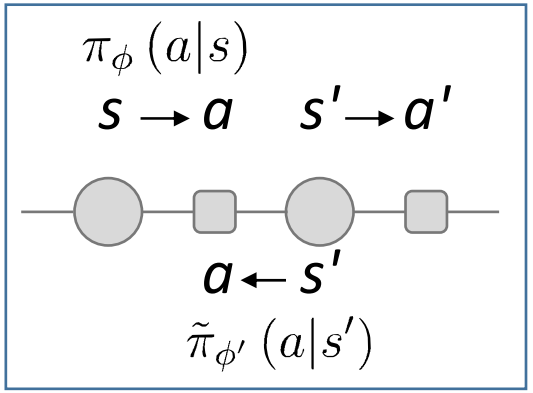
$$\mathcal{L}_b(\theta') = \sum_{t=1}^{N} \left[\mu_{\theta'}\left(s_{t+1},a_t\right) - s_t\right]^\top \Sigma_{\theta'}^{-1}\left(s_{t+1},a_t\right)\left[\mu_{\theta'}\left(s_{t+1},a_t\right) - s_t\right] + \log\det\Sigma_{\theta'}\left(s_{t+1},a_t\right)$$

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 lean 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):

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

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.

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

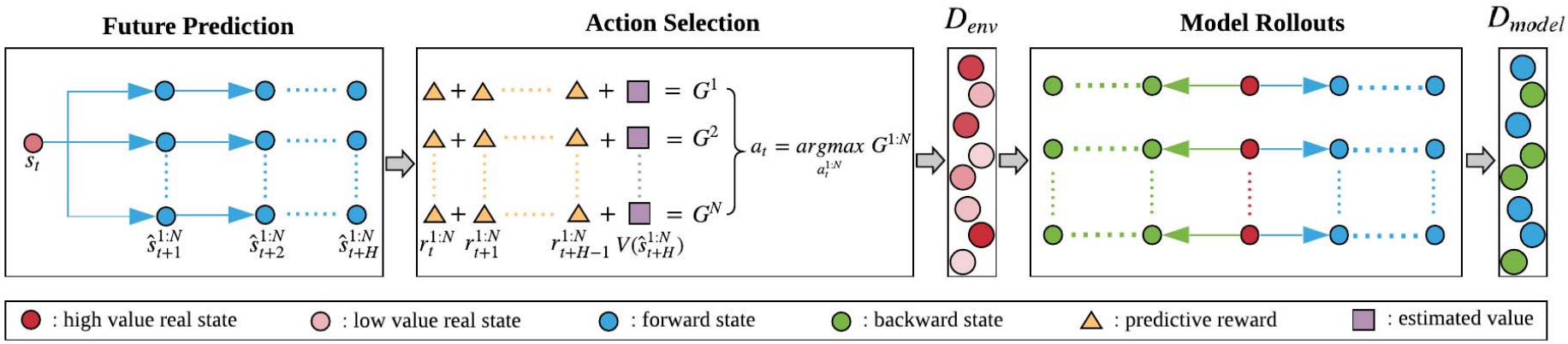
  - Thus, the agent could learn to reach high value states through backward rollouts, and also learn to act better after these states through forward rollouts

- Incorporating MPC 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:

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

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

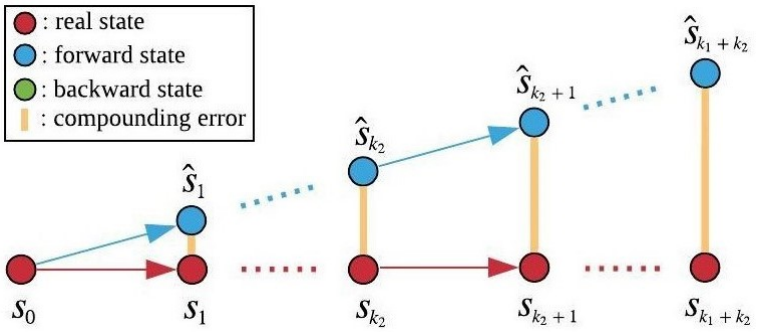
Overall Algorithm of BMPO



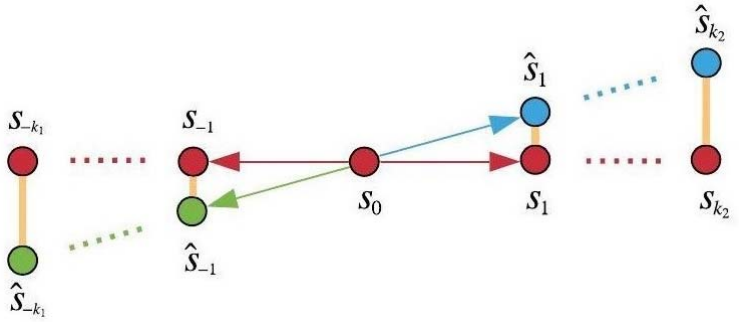
1. When interacting with the environment, the agent uses the model to perform MPC-based action selection
2. Data is stored in D_{env}, where the value of state increases from light red to dark red
3. High value states are then sampled from D_{env} to perform bidirectional model rollouts, which are stored in D_{model}
4. Model-free method (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:

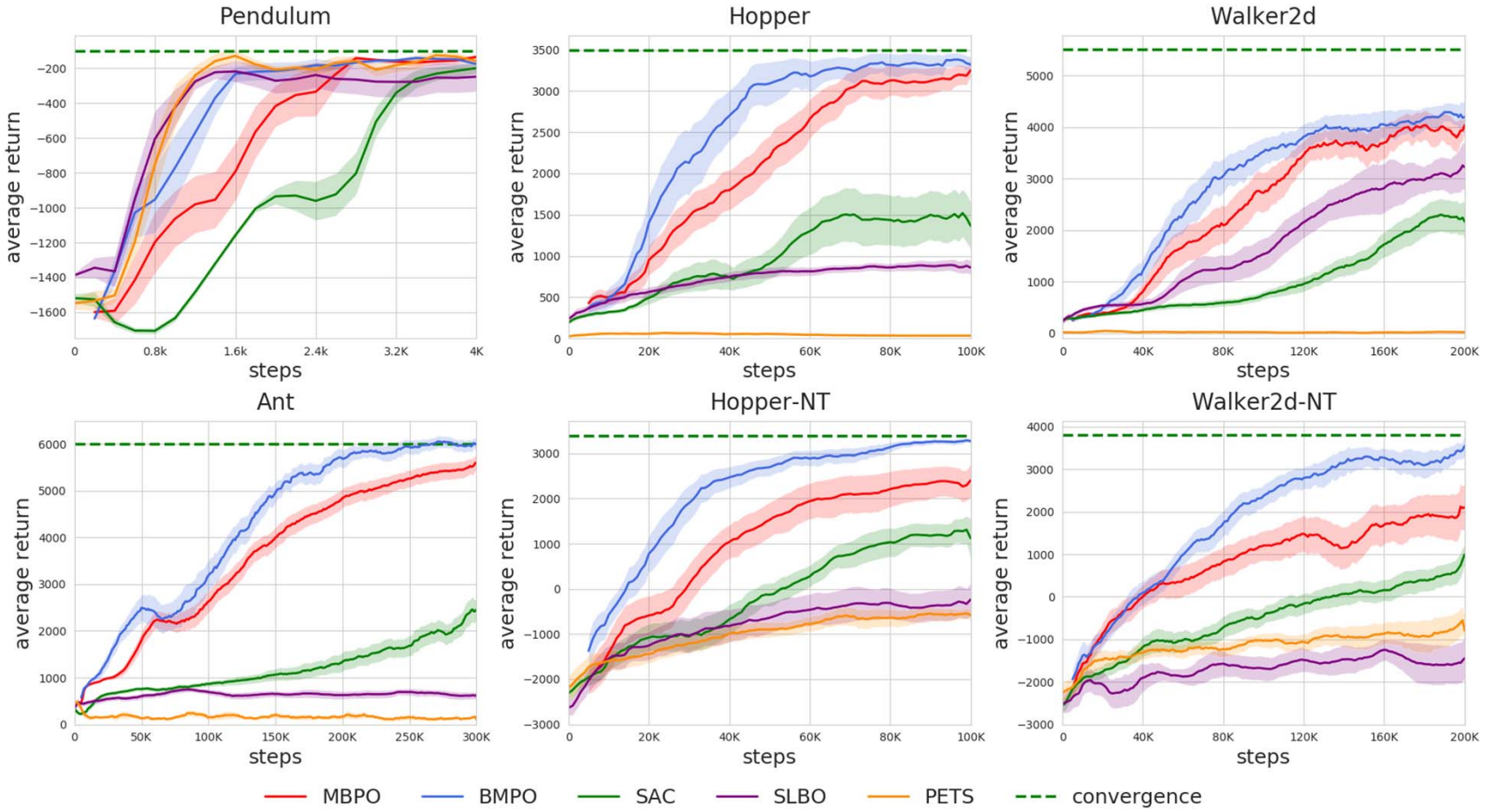
$$\left|\eta[\pi] - \eta^{\text{branch}}[\pi]\right| \leq 2r_{\max}\left[\frac{\gamma^{k_1+k_2+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^{k_1+k_2}\epsilon_\pi}{(1-\gamma)} + \frac{\max(k_1,k_2)\epsilon_m}{1-\gamma}\right].$$

Forward model:

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

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

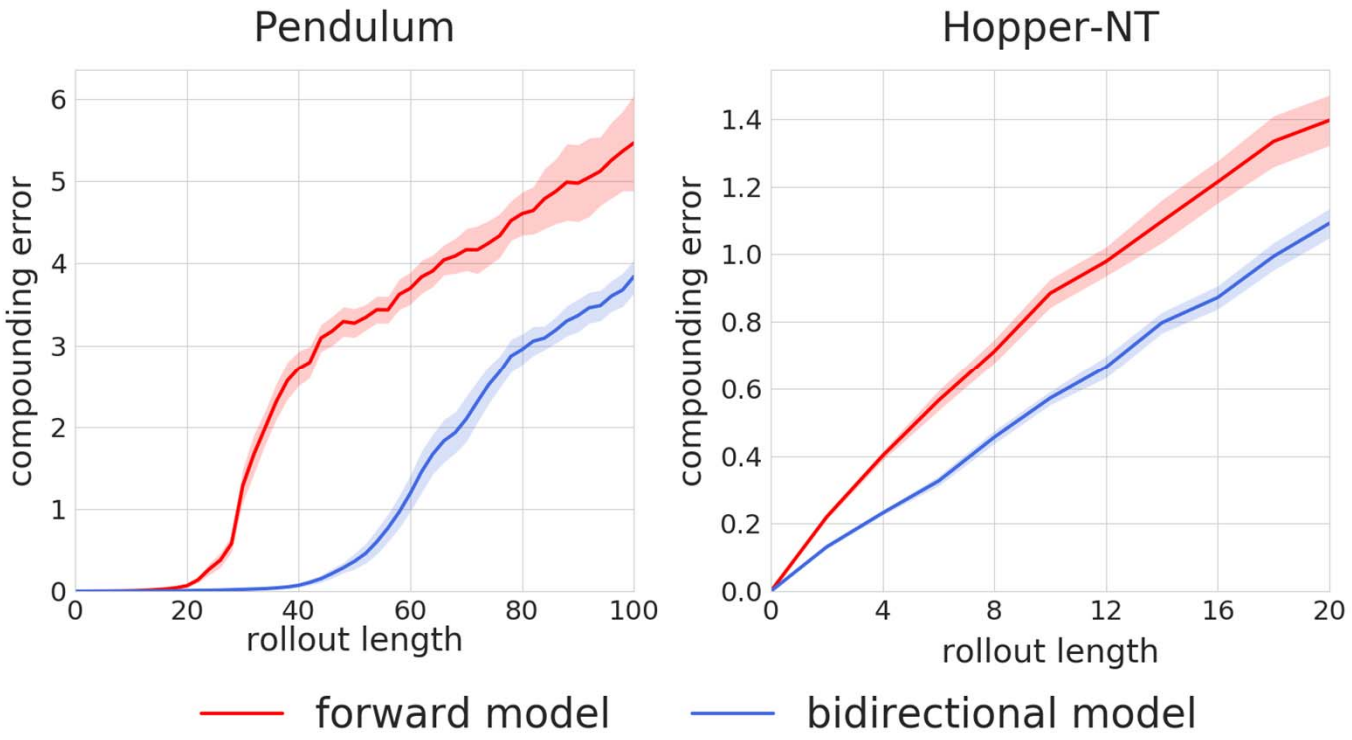
Comparison with State-of-the-Arts



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



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

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

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, POPLIN
3. Theoretic bounds and methods: SLBO, MBPO & BMPO
4. Backpropagation through paths: SVG and MAAC

REVIEW

Deterministic Policy Gradient

- A critic module for state-action value estimation

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

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

- 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

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

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

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

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

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

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

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

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

➡ $y = \mu(x) + \sigma(x)\xi$, 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 $$\nabla_{\mathbf{x}}\mathbb{E}_{p(\mathbf{y}|\mathbf{x})}\mathbf{g}(\mathbf{y}) = \mathbb{E}_{\rho(\xi)}\mathbf{g}_{\mathbf{y}}\mathbf{f}_{\mathbf{x}} \approx \frac{1}{M}\sum_{i=1}^{M}\mathbf{g}_{\mathbf{y}}\mathbf{f}_{\mathbf{x}}\Big|_{\xi=\xi_i}$$

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

From Deterministic to Stochastic

- Deterministic version for reference

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

- Stochastic environment, policy, value and gradients

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

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

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

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

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

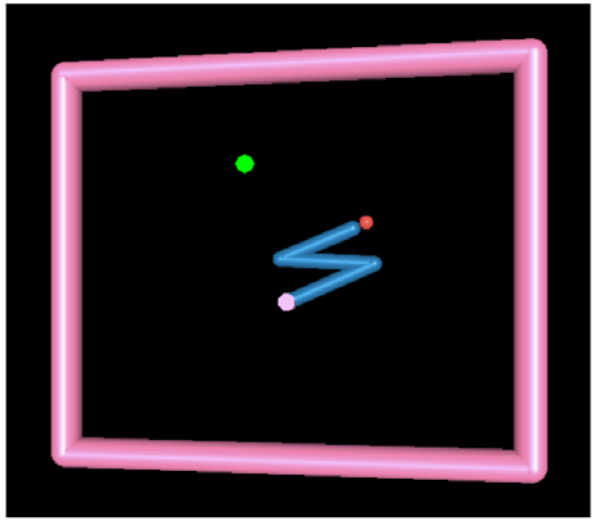
$$g_x \triangleq \partial g(x,y)/\partial x$$

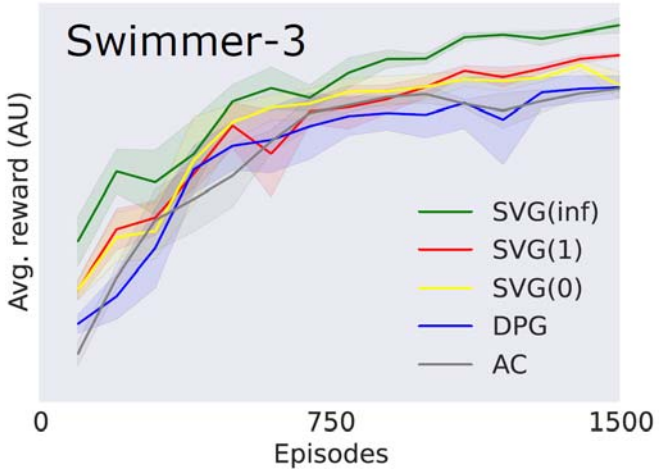
Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.

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





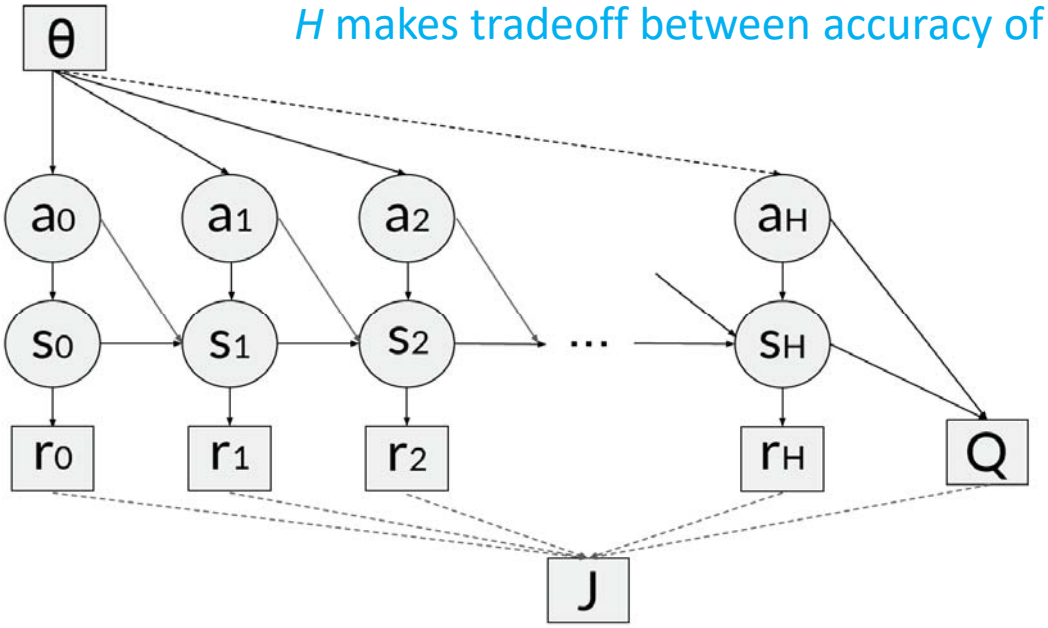
Heess, Nicolas, et al. "Learning continuous control policies by stochastic value gradients." NIPS 2015.

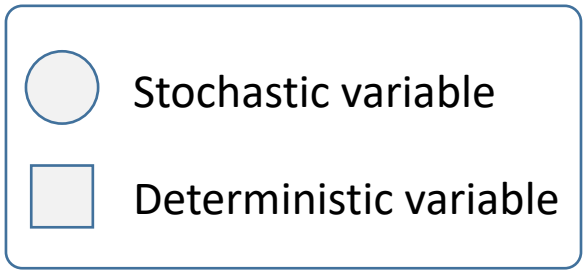
Model-Augmented Actor Critic

- 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

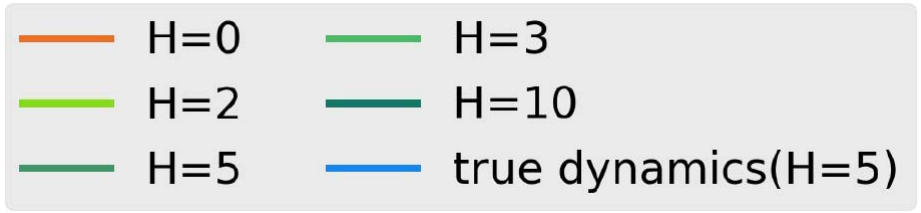


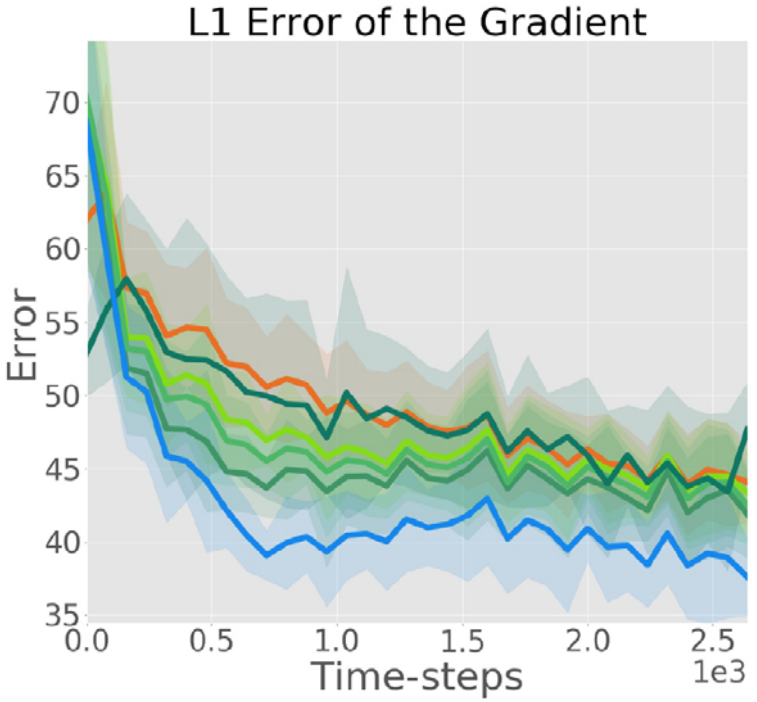


Ignasi Clavera, Yao Fu, Pieter Abbeel. Model-Augmented Actor Critic: Backpropagation through paths. ICLR 2020.

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

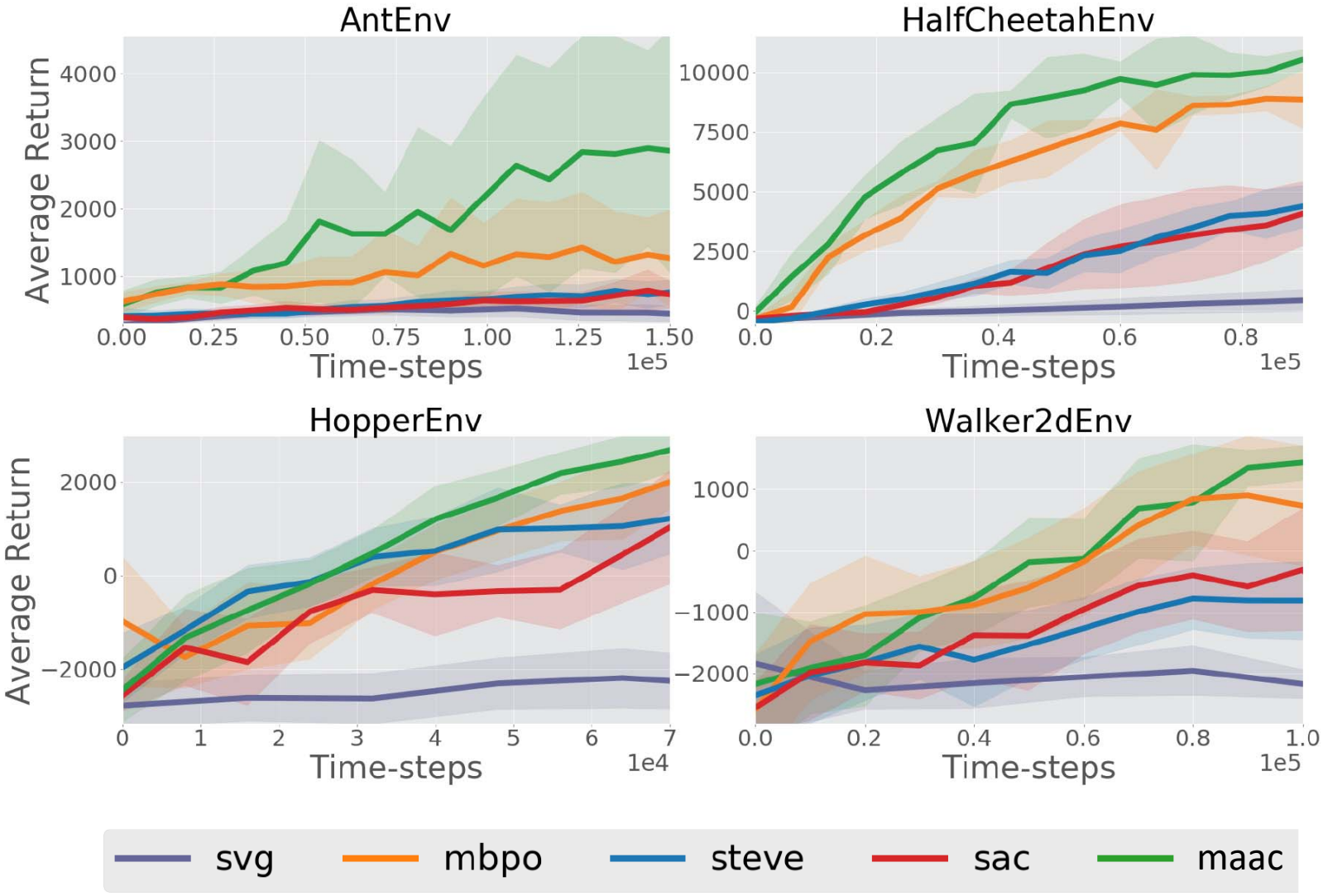
Algorithm 1 MAAC

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.

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

Thank You!
Questions?

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Weinan Zhang

Associate Professor

APEX Data & Knowledge Management Lab

John Hopcroft Center for Computer Science

Shanghai Jiao Tong University

http://wnzhang.net

RLChina 2020