Model-based Reinforcement Learning

陈伟杰 2020.05.14

Outline

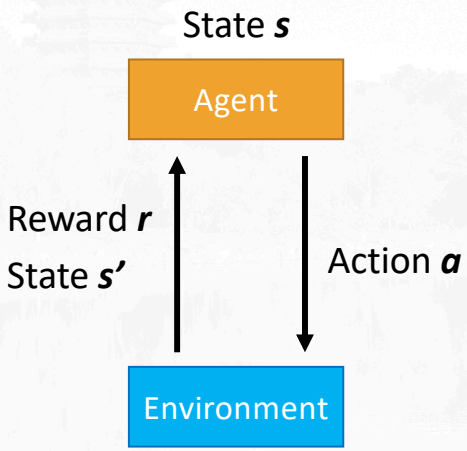
- Introduction

- Algorithms

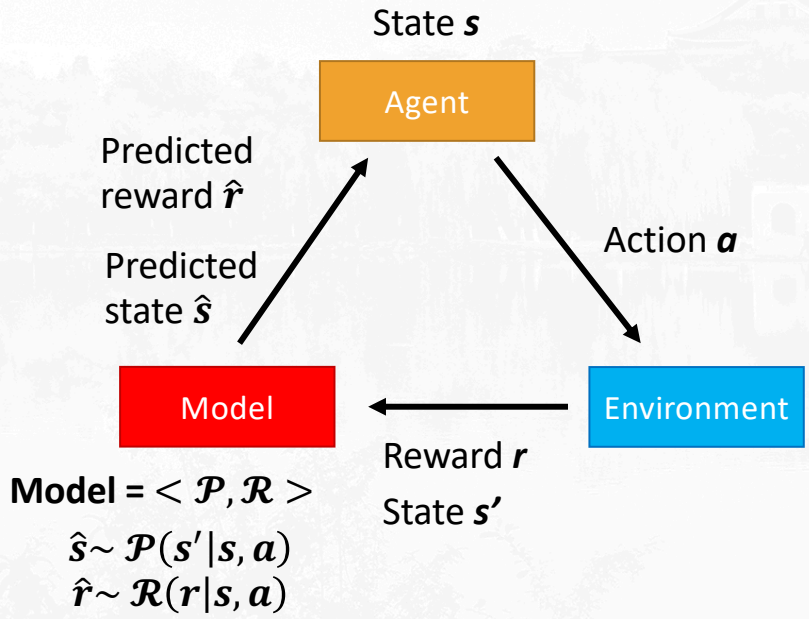
- Discussion & Conclusion

Introduction to model-based RL

Model-free RL



Model-based RL



Model = $< \mathcal{P}, \mathcal{R} >$

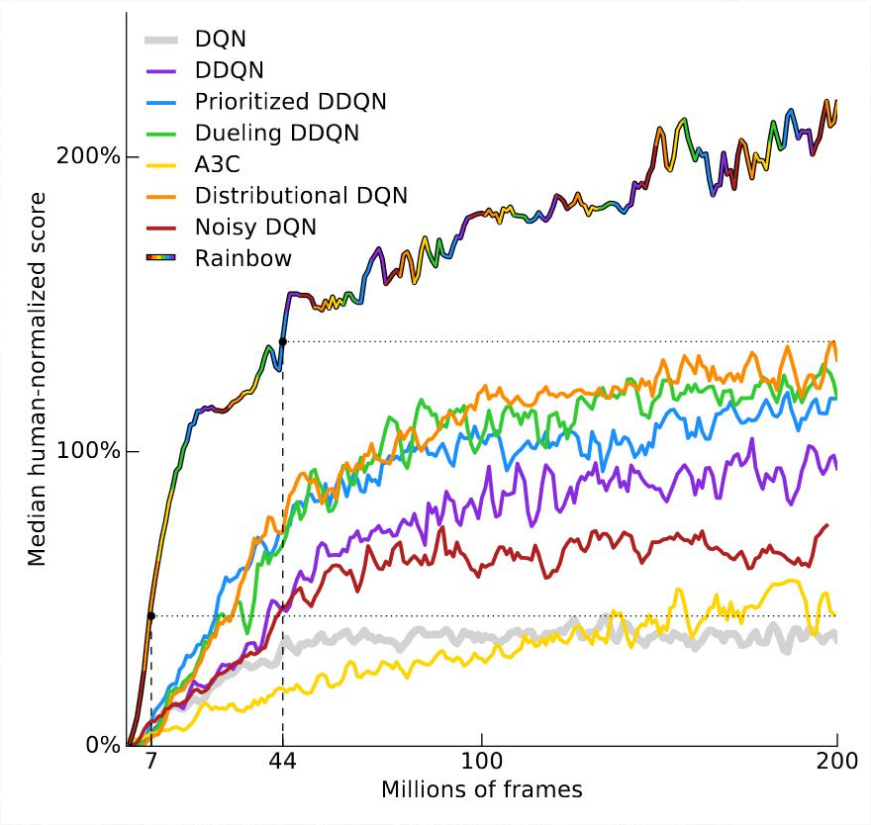
$\hat{s} \sim \mathcal{P}(s'|s,a)$

$\hat{r} \sim \mathcal{R}(r|s,a)$

From human intuition

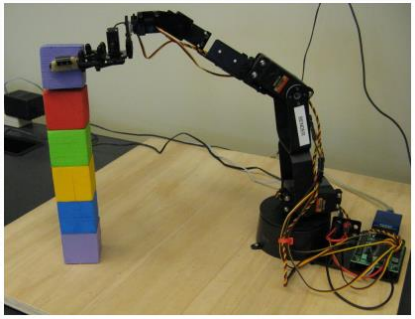
Human players can learn to play Atari games in **minutes** (Tsividis et al., 2017). However, some of the best model-free reinforcement learning algorithms require tens or hundreds of millions of time steps – the equivalent of **several weeks** of training in real time.

Human can predict the outcome of actions because of the some **knowledges or experiences**



Why not learn a model from data?

From sampling efficiency



1, Physics restriction
high cost or long time of interaction with environment

2, Device restriction
Interaction with environment can only be on CPU

3, Sparse restriction
Huge state space and sparse reward signal

From game theory

For policy-player $J(\boldsymbol{\pi}, \boldsymbol{W}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)\right]$

For model-player $\ell(\boldsymbol{M}, \mu) = \mathbb{E}_{(s,a)\sim\mu}\left[D_{KL}\left(P_{\boldsymbol{W}}(\cdot|s,a), P_{\boldsymbol{M}}(\cdot|s,a)\right)\right]$.

A two-player general sum game

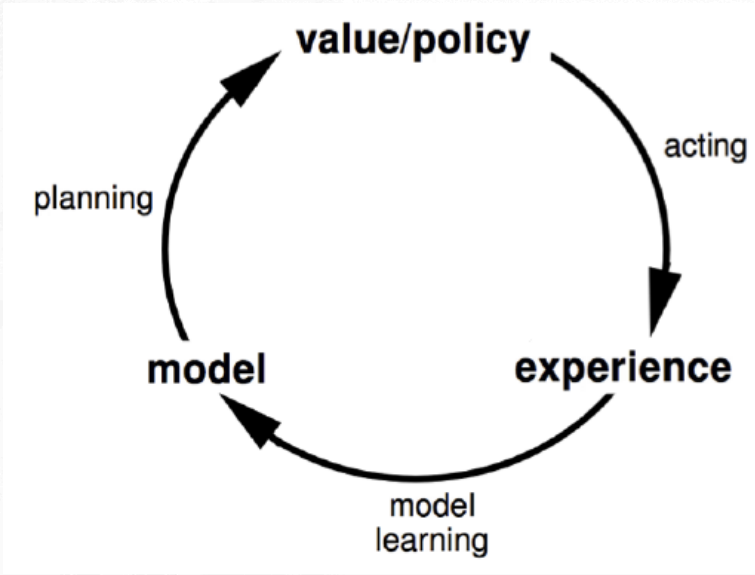
$$\overbrace{\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \boldsymbol{M})}^{\text{policy-player}} \quad , \quad \overbrace{\min_{\boldsymbol{M}} \ell(\boldsymbol{M}, \mu^{\boldsymbol{\pi}}_{\boldsymbol{W}})}^{\text{model-player}}$$

$$\mu^{\boldsymbol{\pi}}_{\boldsymbol{W}} = \frac{1}{T}\sum_{t=0}^{T} P(s_t = s, a_t = a)$$

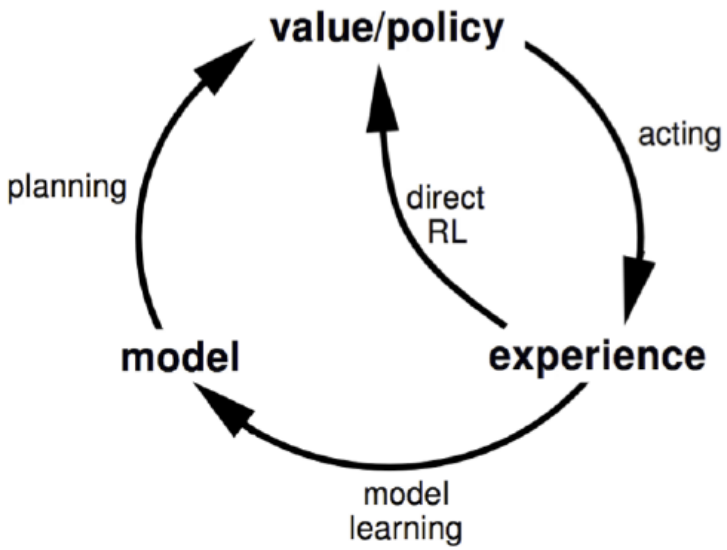
What's more, Imitation learning → Inverse reinforcement learning

Algorithms of model-based RL

Simple model-based RL framework



Dyna framework



Simple model-based RL framework

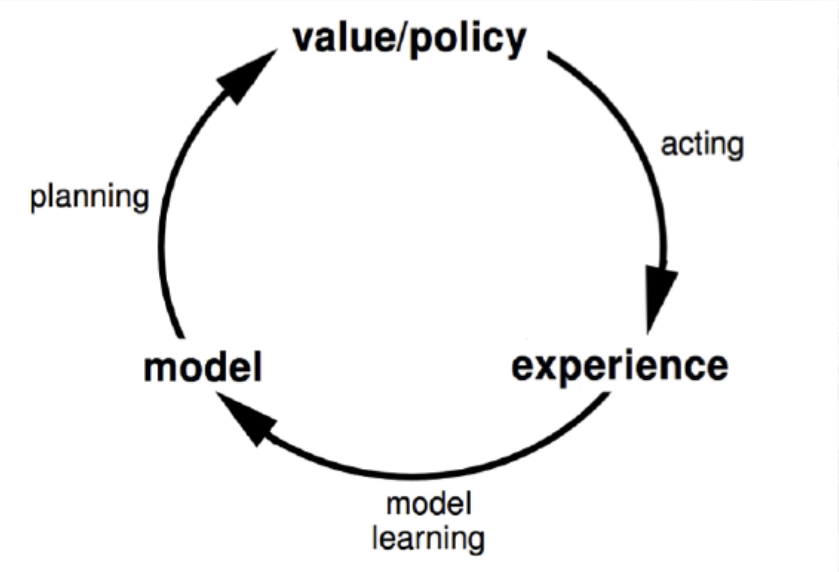
For each step:

agent take action with policy/value

interact with environment and add to data $\mathcal{D}_{env}$

learn a model with sample from $\mathcal{D}_{env}$

update value/policy with sample from $\mathcal{D}_{model}$



How to learn a model

Deterministic model

For model $f(\mathbf{s}, \mathbf{a})$ to minimize $\sum_i \| f(\mathbf{s}_i, \mathbf{a}_i) - \mathbf{s}'_i \|^2$

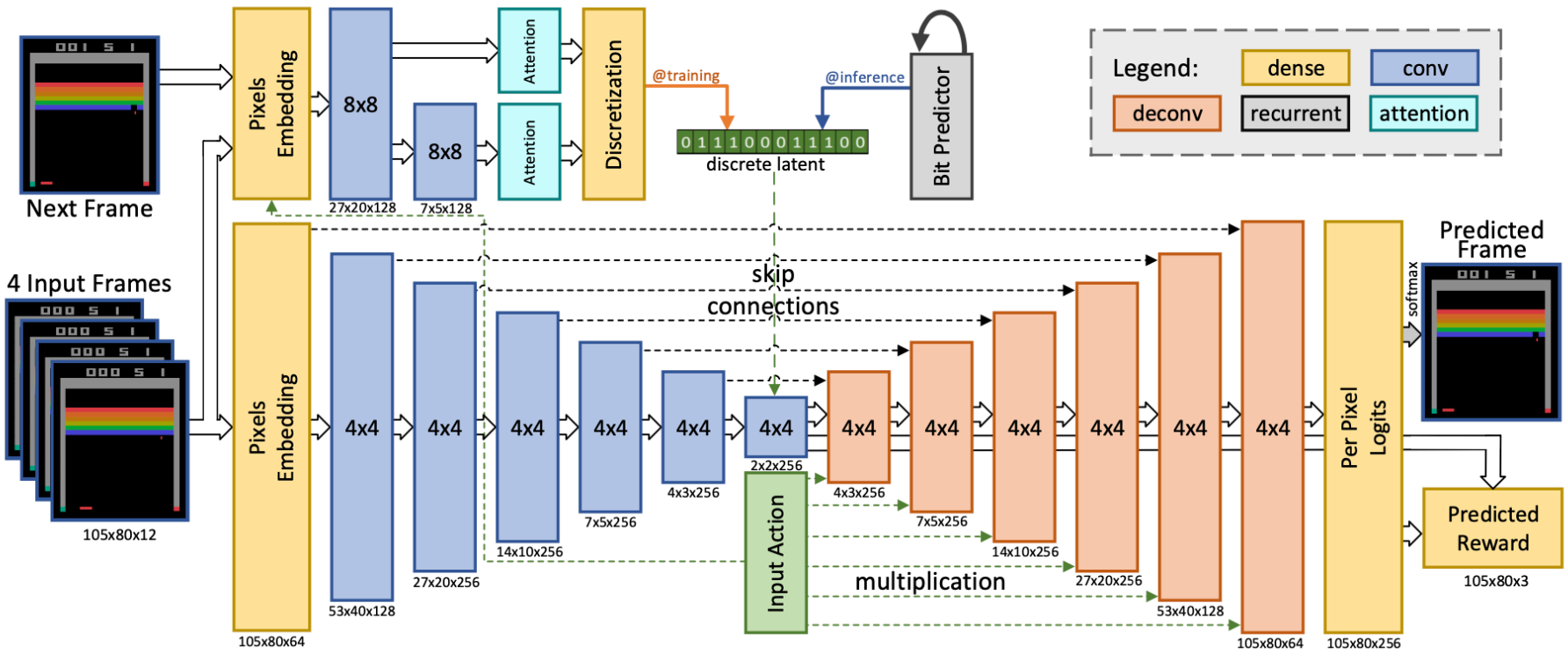
Stochastic model

To learn a parametric distribution

$$p_\theta^i(s_{t+1}, r | s_t, a_t) = \mathcal{N}(\mu_\theta^i(s_t, a_t), \Sigma_\theta^i(s_t, a_t)))$$

What's more, **VAE and GAN**

Using VAE to learn a model for Atari games



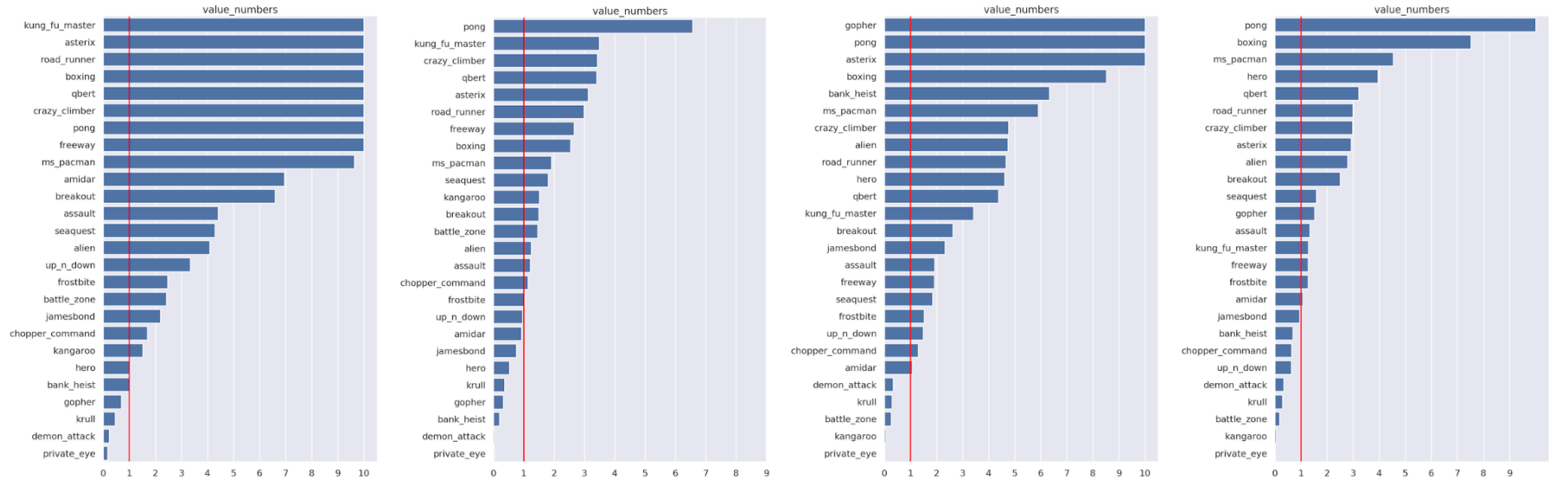


Figure 4: Fractions of Rainbow and PPO scores at different numbers of interactions calculated with the formula $(SimPLe\_score@100K - random\_score)/(baseline\_score - random\_score)$*; if denominator is smaller than 0, both nominator and denominator are increased by 1. From left to right, the baselines are: Rainbow at 100K, Rainbow at 200K, PPO at 100K, PPO at 200K. SimPLe outperforms Rainbow and PPO even when those are given twice as many interactions.*

Convergence of MBRL algorithm

Some definitions

$$J(\boldsymbol{\pi}, \boldsymbol{W}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t)\right] \qquad \ell(\boldsymbol{M}, \mu) = \mathbb{E}_{(s,a)\sim\mu}\left[D_{KL}\left(P_{\boldsymbol{W}}(\cdot|s,a), P_{\boldsymbol{M}}(\cdot|s,a)\right)\right].$$

Lemma 1. *(Simulation Lemma) Suppose* $\boldsymbol{M}$ *is such that* $D_{TV}\left(P_{\boldsymbol{W}}(\cdot|s,a), P_{\boldsymbol{M}}(\cdot|s,a)\right) \leq \epsilon_{\boldsymbol{M}}\ \forall (s,a)$. *Then, for any policy* $\boldsymbol{\pi}$, *we have*

$$J(\boldsymbol{\pi}, \boldsymbol{W}) \geq J(\boldsymbol{\pi}, \boldsymbol{M}) - O\left(\frac{\epsilon_{\boldsymbol{M}}}{(1-\gamma)^2}\right) \quad \forall \boldsymbol{\pi}. \tag{2}$$

where D_{TV} to denote total variation distance.

$$D_{TV}(P, Q) = \sup_{A\in\mathcal{F}} |P(A) - Q(A)|.$$

Theorem 1. *(Global performance of approximate equilibrium pair; informal) Suppose we have a pair of policy and model,* $(\boldsymbol{\pi}, \boldsymbol{M})$*, such that simultaneously*

$$\ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}}) \leq \epsilon_{\boldsymbol{M}} \quad \textit{and} \quad J(\boldsymbol{\pi}, \boldsymbol{M}) \geq \sup_{\boldsymbol{\pi}'} J(\boldsymbol{\pi}', \boldsymbol{M}) - \epsilon_{\boldsymbol{\pi}}.$$

Let $\boldsymbol{\pi}^*$ *be an optimal policy and denote corresponding performance as* $J_{\boldsymbol{W}}^* = \sup_{\boldsymbol{\pi}'} J(\boldsymbol{\pi}', \boldsymbol{W})$*. Then, the performance gap is bounded by*

$$J_{\boldsymbol{W}}^* - J(\boldsymbol{\pi}, \boldsymbol{W}) \leq O\left(\epsilon_{\boldsymbol{\pi}} + \frac{\sqrt{\epsilon_{\boldsymbol{M}}}}{(1-\gamma)^2} + \frac{1}{1-\gamma} D_{TV}\left(\mu_{\boldsymbol{W}}^{\boldsymbol{\pi}^*}, \mu_{\boldsymbol{M}}^{\boldsymbol{\pi}^*}\right)\right) \tag{5}$$

Remark:

1, The first two terms are related to sub-optimality in policy optimization (planning) and model learning

2, There may be multiple Nash equilibria for the MBRL game, and the third domain adaptation or transfer learning term in the bound captures the quality of an equilibrium

Game theoretical perspective (Alternating optimization)

$$\overbrace{\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \boldsymbol{M})}^{\text{policy-player}} \quad , \quad \overbrace{\min_{\boldsymbol{M}} \ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}})}^{\text{model-player}}$$

Independent simultaneous learners

Gradient Descent Ascent (GDA) In GDA, each player performs an improvement step holding the parameters of the other player fixed. The resulting updates are given below.

$$\boldsymbol{\pi}_{k+1} = \boldsymbol{\pi}_k + \alpha_k \nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}_k, \boldsymbol{M}_k) \qquad \text{(conservative policy step)} \qquad (6)$$

$$\boldsymbol{M}_{k+1} = \boldsymbol{M}_k - \beta_k \nabla_{\boldsymbol{M}} \ell(\boldsymbol{M}_k, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}_k}) \qquad \text{(conservative model step)} \qquad (7)$$

Best Response (BR) In BR, each player fixes the parameters of the other player and computes the *best response* – the parameters that optimize their objective. To approximate the best response, we can take a large number of gradient steps.

$$\boldsymbol{\pi}_{k+1} = \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \boldsymbol{M}_k) \qquad \text{(aggressive policy step)} \qquad (8)$$

$$\boldsymbol{M}_{k+1} = \arg\min_{\boldsymbol{M}} \ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}_k}) \qquad \text{(aggressive model step)} \qquad (9)$$

Stackelberg formulation (Subproblem optimization)

Policy As Leader (PAL): Choosing the policy player as leader results in the following optimization:

$$\max_{\boldsymbol{\pi}} \left\{ J(\boldsymbol{\pi}, \boldsymbol{M}^{\boldsymbol{\pi}}) \;\; s.t. \;\; \boldsymbol{M}^{\boldsymbol{\pi}} \in \arg\min_{\boldsymbol{M}} \; \ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}}) \right\} \tag{14}$$

We solve this nested optimization using the first order gradient approximation, resulting in updates:

$$\boldsymbol{M}_{k+1} \approx \arg\min_{\boldsymbol{M}} \ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}_k}) \qquad \text{(aggressive model step)} \tag{15}$$

$$\boldsymbol{\pi}_{k+1} = \boldsymbol{\pi}_k + \alpha_k \nabla_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \boldsymbol{M}_{k+1}) \qquad \text{(conservative policy step)} \tag{16}$$

Model as Leader (MAL): Conversely, choosing model as the leader results in the optimization

$$\max_{\boldsymbol{\pi}} \left\{ J(\boldsymbol{\pi}, \boldsymbol{M}^{\boldsymbol{\pi}}) \;\; s.t. \;\; \boldsymbol{M}^{\boldsymbol{\pi}} \in \arg\min_{\boldsymbol{M}} \ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}}) \right\}. \tag{17}$$

Similar to the PAL formulation, using first order approximation to the bi-level gradient results in:

$$\boldsymbol{\pi}_{k+1} \approx \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}, \boldsymbol{M}_k) \qquad \text{(aggressive policy step)} \tag{18}$$

$$\boldsymbol{M}_{k+1} = \boldsymbol{M}_k - \beta_k \nabla_{\boldsymbol{M}} \ell(\boldsymbol{M}, \mu_{\boldsymbol{W}}^{\boldsymbol{\pi}_{k+1}}) \qquad \text{(conservative model step)} \tag{19}$$

Experiments on a suite of continuous control tasks

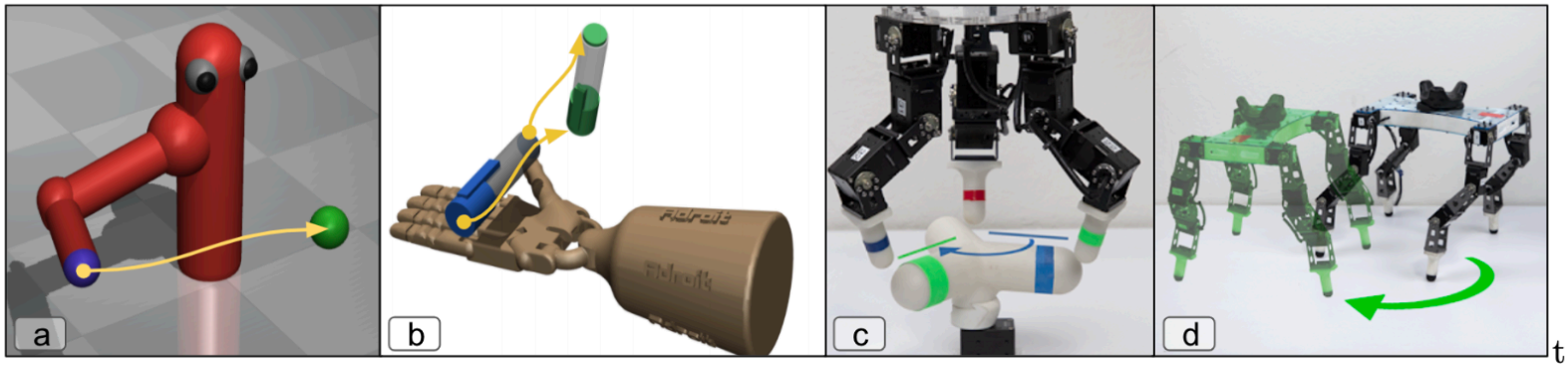
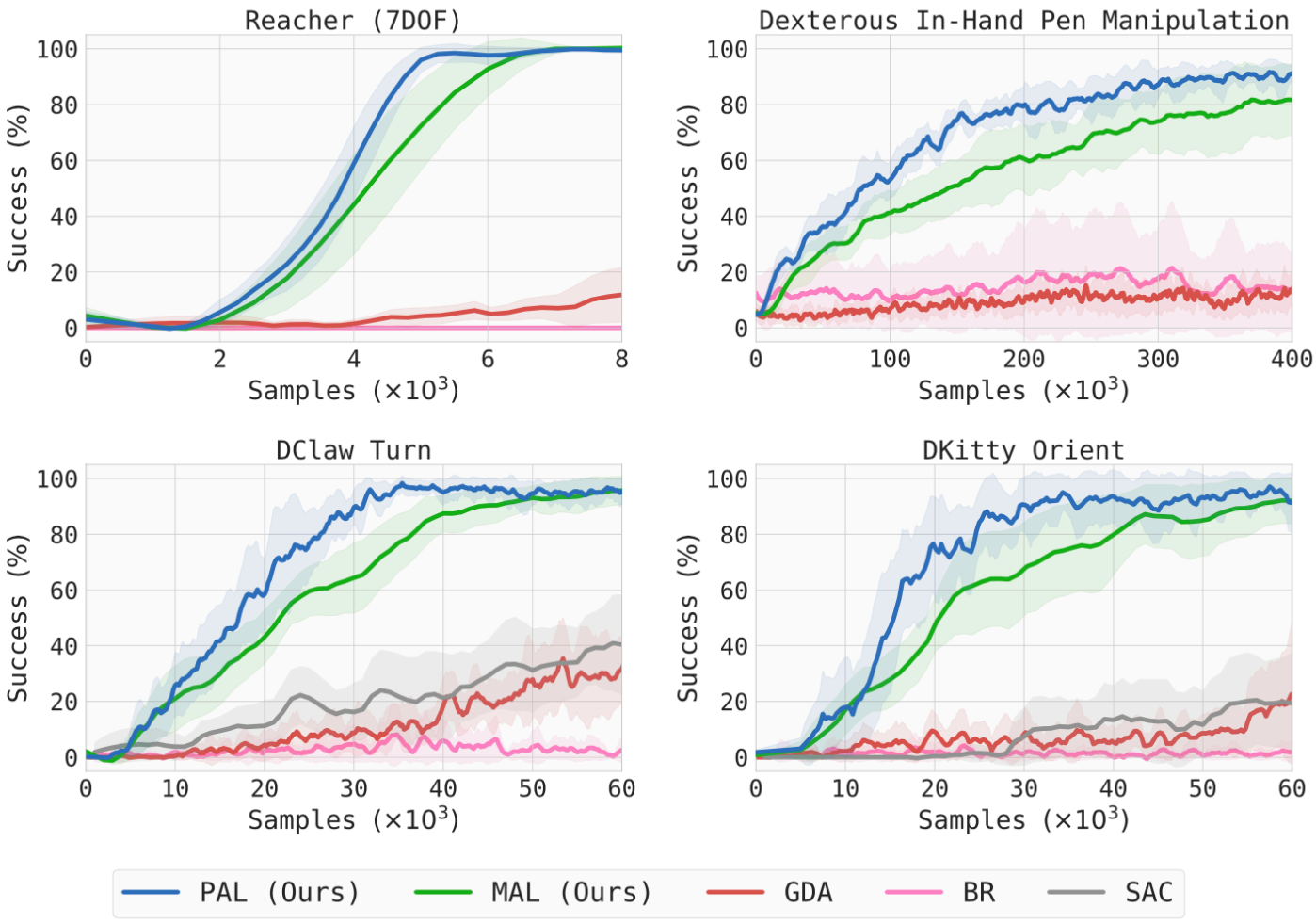


Figure 1: (a) Reacher task with a 7DOF arm. (b) In-hand manipulation task with a 24DOF dexterous hand. (c) DClaw-Turn task with a 3 fingered "claw". (d) DKitty-Orient task with a quadrupedal robot. In all the tasks, the desired goal locations and/or orientations are randomized for every episode. This forces the learning of generalizable policies that can be successful for many goal specifications, and we measure the success rate in our experiments.


Reacher (7DOF)
Dexterous In-Hand Pen Manipulation
DClaw Turn
DKitty Orient
Success (%)
Samples (×10^3)
PAL (Ours)
MAL (Ours)
GDA
BR
SAC

Model ensemble & Model rollout (playout)

Definition of **model rollout (**k-branched**)**

We begin a rollout from a state under the previous policy's state distribution $\mathcal{D}_{env}$ and run **k** steps according to policy **π** under the learned model p_θ

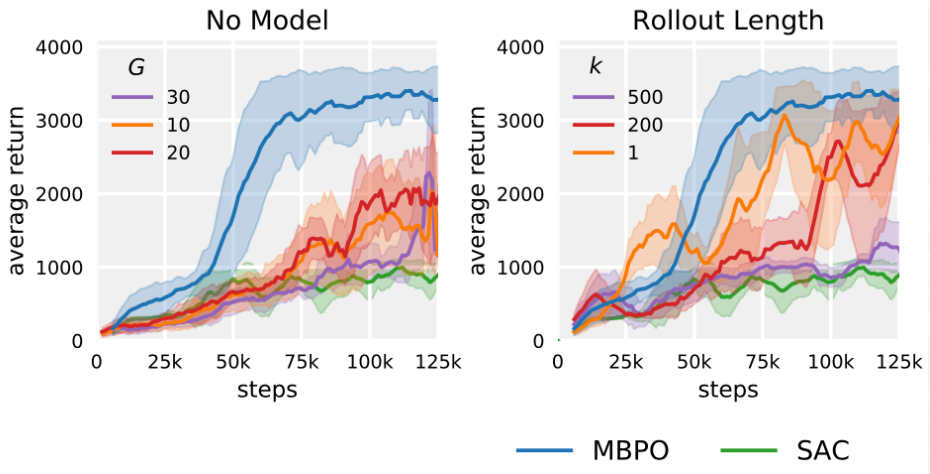
$$\eta[\pi] \geq \eta^{\text{branch}}[\pi] - 2r_{\max}\left[\frac{\gamma^{k+1}\epsilon_\pi}{(1-\gamma)^2} + \frac{\gamma^k + 2}{(1-\gamma)}\epsilon_\pi + \frac{k}{1-\gamma}(\epsilon_m + 2\epsilon_\pi)\right]$$

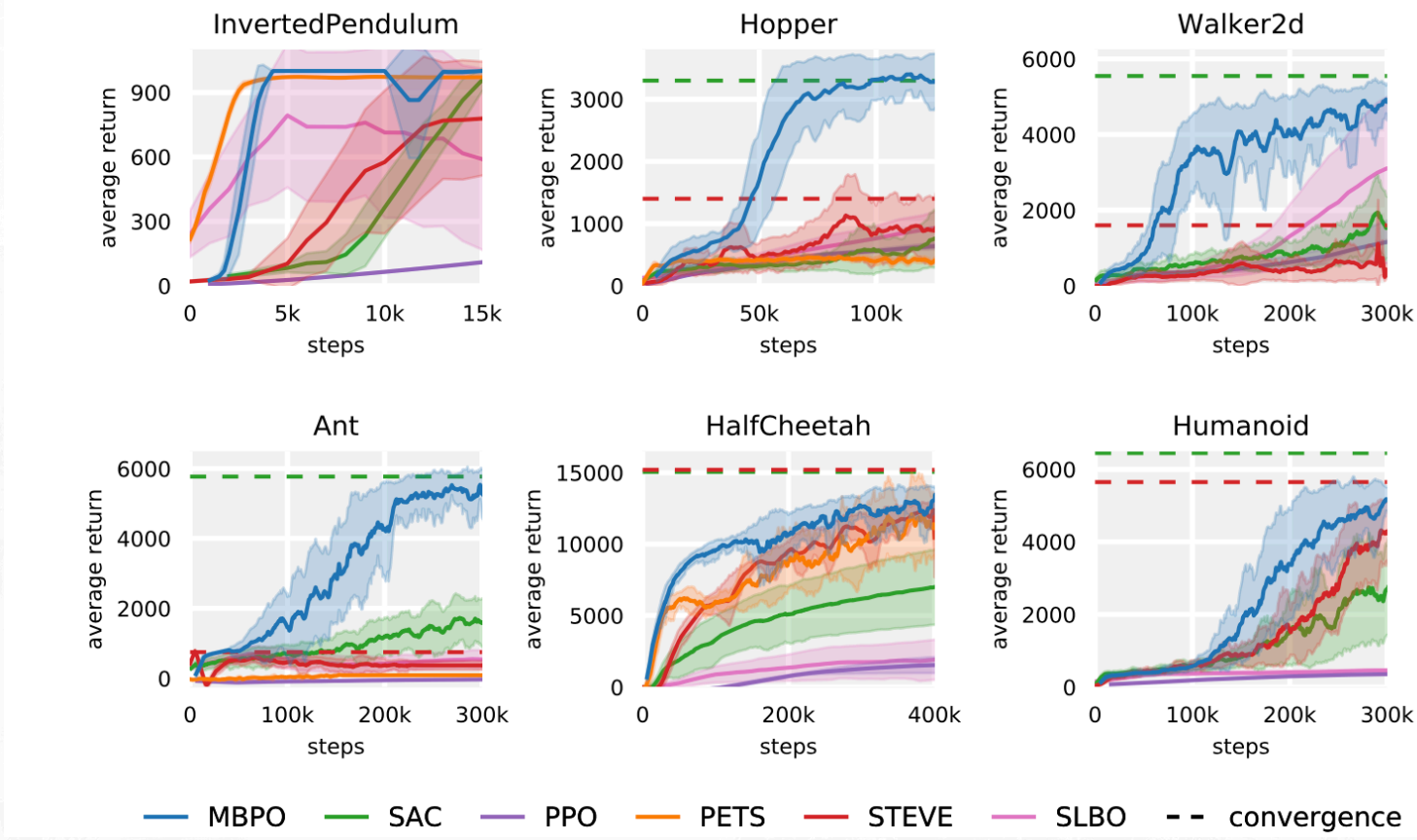
where $\eta[\pi] = E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$ is the returns of the policy in the true MDP

$\eta^{\text{branch}}[\pi]$ is the returns of policy from k-branched rollouts from the model

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: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$




InvertedPendulum
Hopper
Walker2d
Ant
HalfCheetah
Humanoid
average return
steps
MBPO
SAC
PPO
PETS
STEVE
SLBO
convergence

Dyna framework

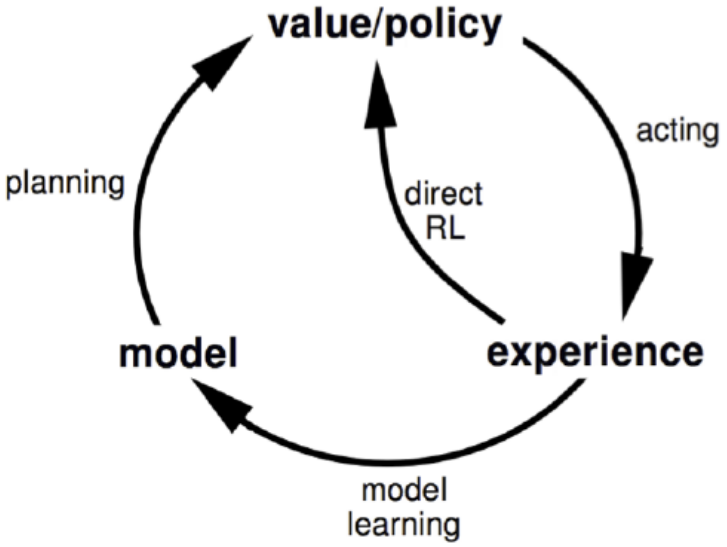
For each step:

agent take action with policy/value

interact with environment and add to data $\mathcal{D}_{env}$

learn a model with sample from $\mathcal{D}_{env}$

update value/policy with sample from $\mathcal{D}_{model}$ and $\mathcal{D}_{env}$

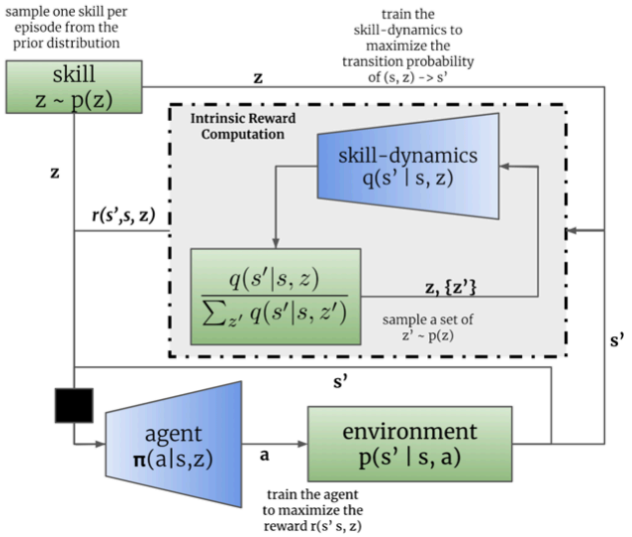


Dynamics-Aware Discovery of Skills (DADS)

Motivation： learning an accurate model for complex dynamical systems is difficult, and even then, the model might not generalize well out- side the distribution of states on which it was trained .

True dynamics of env → MBRL estimation → skill-conditioned transition function

$p(s'|s,a)$ → $\hat{p}\,(s'|s,a)$ → $\mathrm{q}(s'|s,z)$



Algorithm 1: Dynamics-Aware Discovery of Skills (DADS)

Initialize π, q_ϕ;
while *not converged* **do**
  Sample a skill $z \sim p(z)$ every episode;
  Collect new M on-policy samples;
  Update q_ϕ using K_1 steps of gradient descent on M transitions;
  Compute $r_z(s, a, s')$ for M transitions;
  Update π using any RL algorithm;
end

In particular, we propose to maximize the mutual information between the next state **s'** and current skill **z** conditioned on the current state **s**.

$$\begin{aligned}\mathcal{I}(s';z \mid s) &= \mathcal{H}(z \mid s) - \mathcal{H}(z \mid s', s) \\ &= \mathcal{H}(s' \mid s) - \mathcal{H}(s' \mid s, z)\end{aligned}$$

Using the definition of conditional mutual information, we can rewrite

$$\mathcal{I}(s';z \mid s) = \int p(z, s, s') \log \frac{p(s' \mid s, z)}{p(s' \mid s)} ds' ds dz$$

where $p(z, s, s') = p(z)p(s \mid z)p(s' \mid s, z)$, $p(z)$ is user specified prior over $\mathcal{Z}$, $p(s|z)$ denotes the stationary state-distribution induced by $\pi(a|s, z)$ for a skill z and $p(s'|s, z)$ denotes the transition distribution under skill z.

$$\begin{aligned}\mathcal{I}(s';z \mid s) &= \mathbb{E}_{z,s,s' \sim p}\left[\log \frac{p(s' \mid s, z)}{p(s' \mid s)}\right] \\ &= \mathbb{E}_{z,s,s' \sim p}\left[\log \frac{q_\phi(s' \mid s, z)}{p(s' \mid s)}\right] + \mathbb{E}_{s,z \sim p}\Big[\mathcal{D}_{KL}(p(s' \mid s, z) \mid\mid q_\phi(s' \mid s, z))\Big] \\ &\geq \mathbb{E}_{z,s,s' \sim p}\left[\log \frac{q_\phi(s' \mid s, z)}{p(s' \mid s)}\right]\end{aligned}$$

Tighten variational lower bound

$$\nabla_\phi \mathbb{E}_{s,z}[\mathcal{D}_{KL}(p(s' \mid s,z) \,||\, q_\phi(s' \mid s,z))] = \nabla_\phi \mathbb{E}_{z,s,s'}\left[\log \frac{p(s' \mid s,z)}{q_\phi(s' \mid s,z)}\right]$$
$$= -\mathbb{E}_{z,s,s'}\left[\nabla_\phi \log q_\phi(s' \mid s,z)\right]$$

which corresponds to maximizing the likelihood of the samples from p under q_ϕ.

we approximate the reward function for π:

$$r_z(s,a,s') = \log \frac{q_\phi(s' \mid s,z)}{\sum_{i=1}^{L} q_\phi(s' \mid s,z_i)} + \log L, \quad z_i \sim p(z).$$

The approximation is motivated as follows:

$$p(s' \mid s) = \int p(s' \mid s,z)p(z|s)dz \approx \int q_\phi(s' \mid s,z)p(z)dz \approx \frac{1}{L}\sum_{i=1}^{L} q_\phi(s' \mid s,z_i) \text{ for } z_i \sim p(z)$$

where L denotes the number of samples from the prior $p(z)$.

In order to perform planning, we employ the model-predictive-control (MPC) paradigm:

a set of action plans $P_k = (a_{k,1}, \ldots a_{k,H}) \sim P$ for a planning horizon H

trajectory $\hat{\tau}_k = (s_{k,1}, a_{k,1} \ldots s_{k,H+1})$

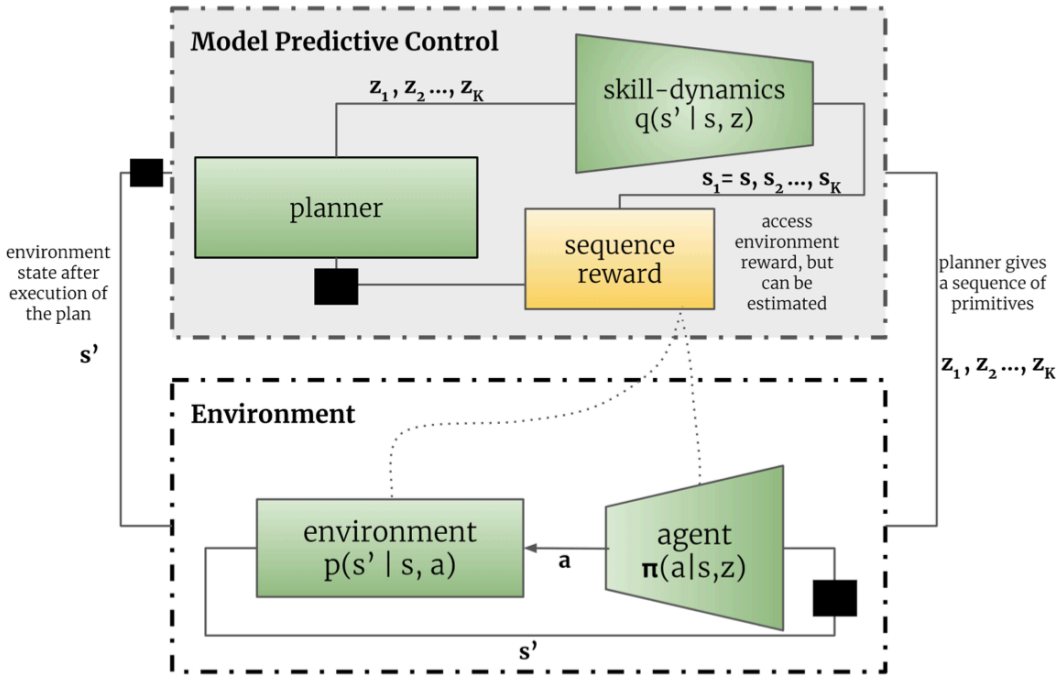
reward $r(\hat{\tau}_k)$ for its trajectory

For DADS, we define $P_k = (z_{k,1}, \ldots z_{k,H_P})$

$$\hat{\tau}_k = (s_{k,1}, z_{k,1}, a_{k,1}, s_{k,2}, z_{k,2}, a_{k,2}, \ldots s_{k,H+1})$$

Compute the reward $r(\hat{\tau}_k)$ for refining the prior

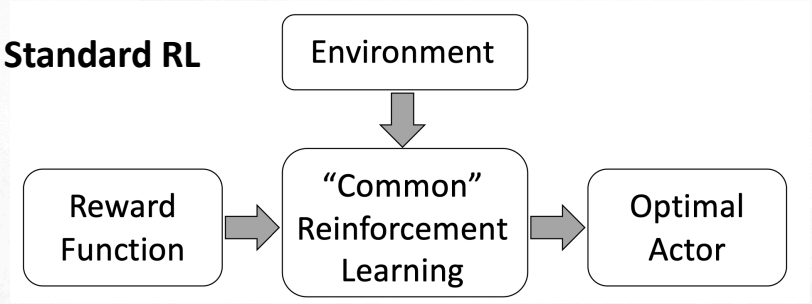
$$\mu_i = \sum_{k=1}^{K} \frac{\exp(\gamma r_k)}{\sum_{p=1}^{K} \exp(\gamma r_p)} z_{k,i} \quad \forall i = 1, \ldots H_P$$

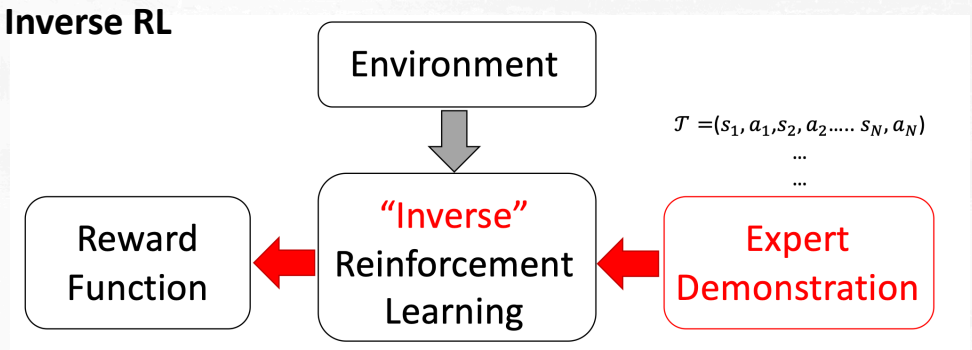


Algorithm 2: Latent Space Planner

$s \leftarrow s_0$;
Initialize parameters $\mu_1, \ldots \mu_{H_P}$;
for $i \leftarrow 1$ **to** H_E / H_Z **do**
  for $j \leftarrow 1$ **to** R **do**
    $\{z_i, \ldots z_{i+H_P-1}\}_{k=1}^{K} \sim \mathcal{N}_i, \ldots \mathcal{N}_{i+H_P-1}$;
    Compute r_{env} for $\{z_i, \ldots z_{i+H_P-1}\}_{k=1}^{K}$;
    Update $\mu_i, \ldots, \mu_{i+H_P-1}$;
  end
  Sample z_i from $\mathcal{N}_i$;
  Execute $\pi(a|s, z_i)$ for H_Z steps;
  Initialize μ_{i+H_P};
end

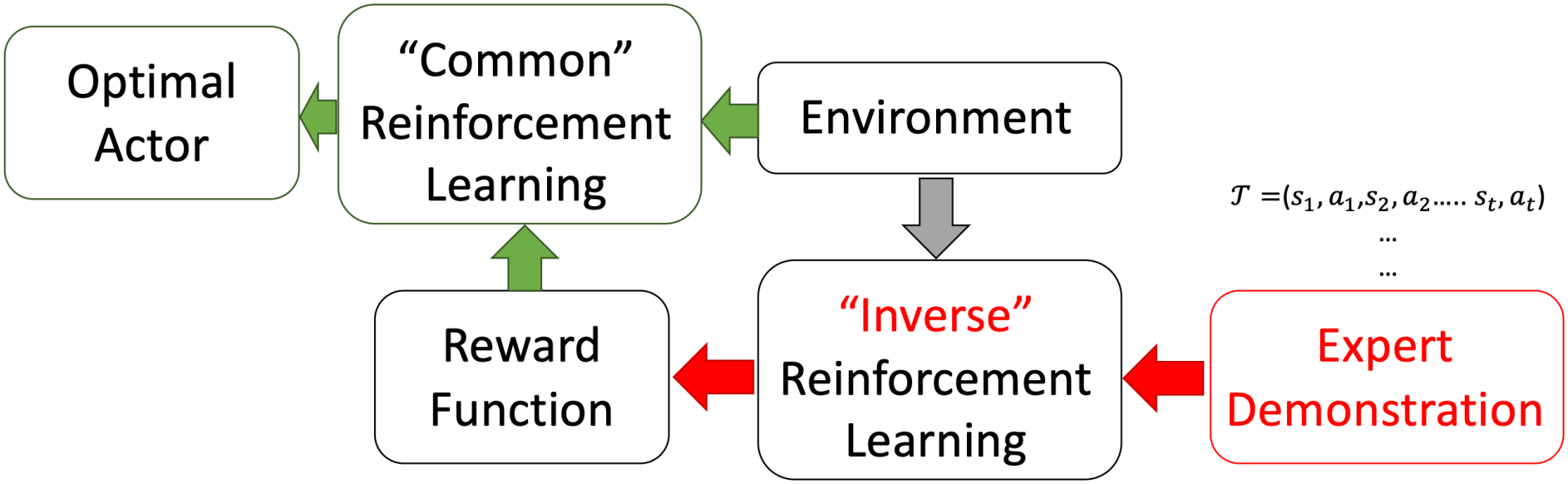
Inverse RL & SeqGAN*





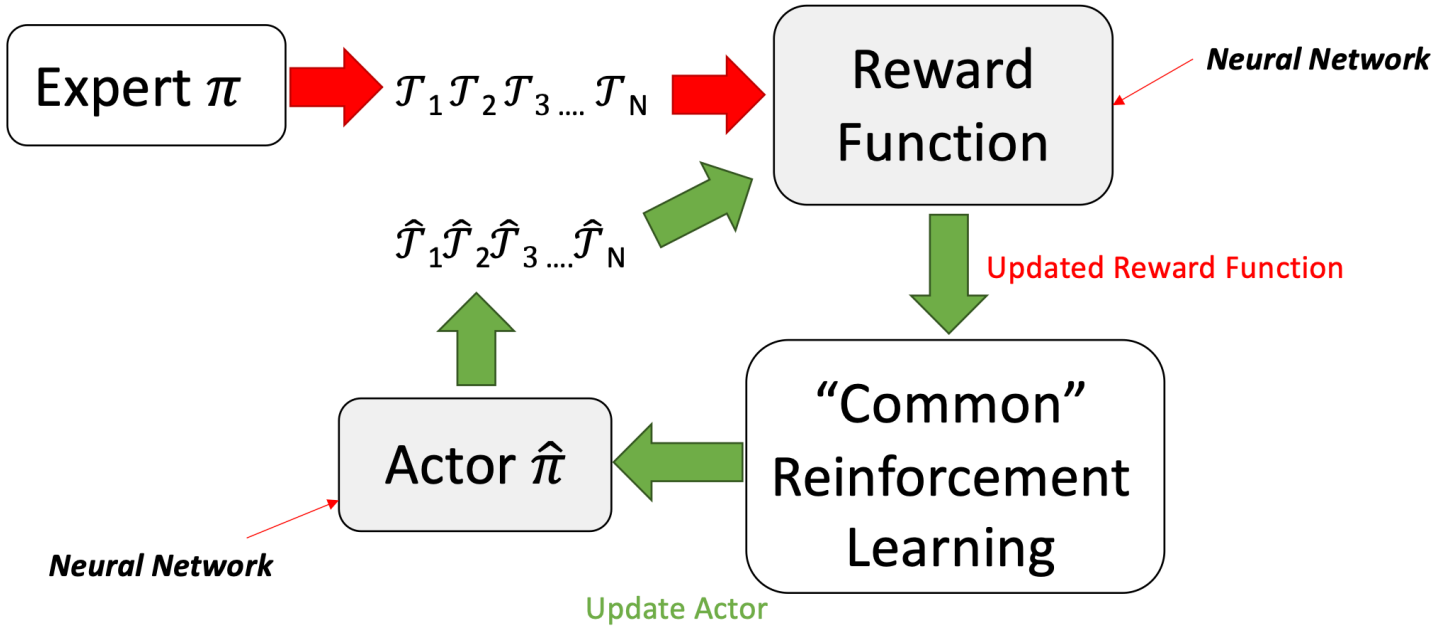
*This part of slide credits to H.Dong in *Deep Generative Model*

- **Inverse Reinforcement Learning**



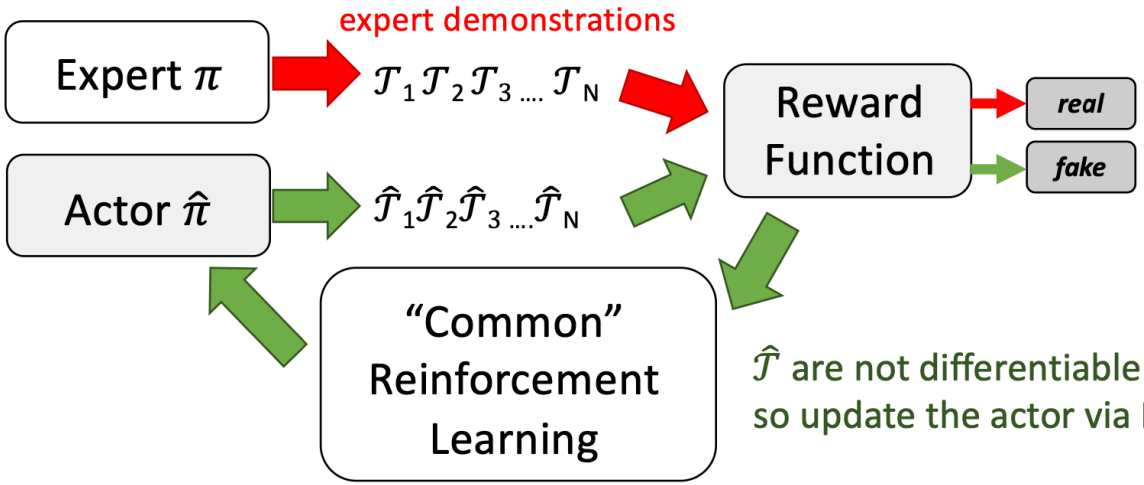
$$\sum_{n=1}^{N} R(\mathcal{T}_n) > \sum_{n=1}^{N} R(\hat{\mathcal{T}}_n)$$

Goal: The expert is always the best

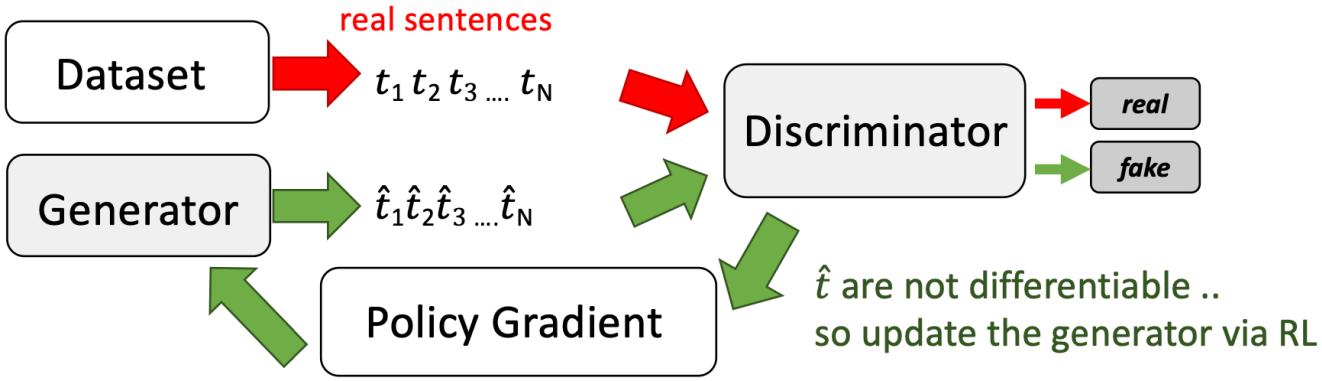


SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient

- **Inverse RL**



- **SeqGAN**



Discussion & Conclusion

	Advantages	Disadvantages
Model-free RL	- Straight and simple - Needs no accurate representation of the environment.	- Actual experiences need to be gathered for training, which makes exploration more expensive. - Cannot carry an explicit plan of how environmental dynamics affects the system
Model-based RL	- Low cost of exploration - can train on simulator.	- Introducing model errors to RL tasks. - Computationally more complex than model-free methods.

Thank you

Q&A