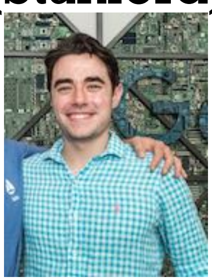
# Model-based Algorithms for Reinforcement Learning and Imitation Learning with Theoretical Analyses


Tengyu Ma
(Stanford)

Nick Landolfi
(Stanford)



Yuping Luo
(Princeton)

Garrett Thomas
(Stanford)

Huazhe Xu
(Berkeley)

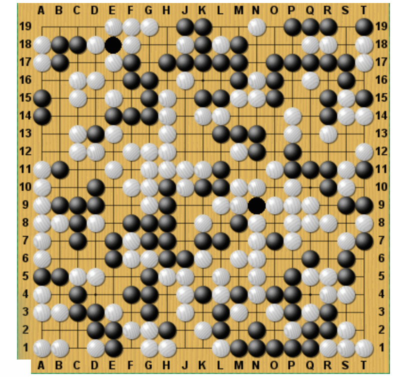
Trevor Darrell
(Berkeley)

Yuandong Tian
(Facebook)

# Sample-Efficiency Challenge in RL

Trials and errors:

- Try the current strategy and collet feedbacks
- Use the feedbacks to improve the strategy




millions of games

How to reduce the amount of trials (samples)?

- Model-based RL
- Imitation learning from expert demonstrations
- Multi-task, lifelong, continual RL
- Hierarchical RL
- Safe RL
- ...

# Backgrounds and Terminologies (on Continuous State-Space, Deterministic Dynamics)

- Time $t = 0, \ldots, T(\approx \infty)$
- State $s_t \in \mathbb{R}^d$; action $a_t \in \mathbb{R}^k$
- Unknown dynamics/environment $M^\star$: $s_{t+1} = M^\star(s_t, a_t)$
- Trajectory:

$$s_0 \sim D_{s_0} \xrightarrow{\pi(s_0)} s_1 \xrightarrow{\pi(s_1)} s_2 \xrightarrow{\pi(s_2)} s_3 \xrightarrow{\pi(s_3)} s_4 \cdots\cdots$$

- Policy $\pi$: states → actions
- Reward $r(s_t, a_t) \in \mathbb{R}$
- Expected payoff of a policy:

$$V^\pi := \mathbb{E}_{s_0 \sim D_{s_0}}[R(s_0, a_0) + R(s_1, a_1) + R(s_2, a_2) + \cdots]$$

- e.g., state = location of arm; action = desired movement

# Model-Based Reinforcement Learning

- Learn the dynamics $M^\star$ somewhat explicitly
- Standard model-based RL algorithm:

Repeat:

1. Sample trajectories from real dynamics $M^\star$ using current policy

$$s_0 \sim D_{s_0} \quad \longrightarrow \quad s_1 \longrightarrow \quad s_2 \longrightarrow \quad s_3 \longrightarrow \quad s_4 \cdots\cdots$$

2. Learn a dynamical model using existing trajectories

$$\min_M \sum ||M(s_t, a_t) - s_{t+1}||_2^2$$

3. Find a good policy for the learned dynamics $M$
   - Does not cost real samples; any RL algo. may be used as a blackbox

3. Planning the vacation at home

1. Go, enjoy, and explore

2. Keep notes on the good restaurants

Repeat:

1. Sample trajectories from real dynamics $M^\star$ using current policy

$$s_0 \sim D_{s_0} \longrightarrow s_1 \longrightarrow s_2 \longrightarrow s_3 \longrightarrow s_4 \cdots\cdots$$

2. Learn a dynamical model using existing trajectories

$$\min_M \sum ||M(s_t, a_t) - s_{t+1}||_2^2$$

3. Find a good policy for the learned dynamics $M$

➢ Does not cost real samples; any RL algo. may be used as a blackbox

# Challenges in Analyzing Deep Model-Based RL

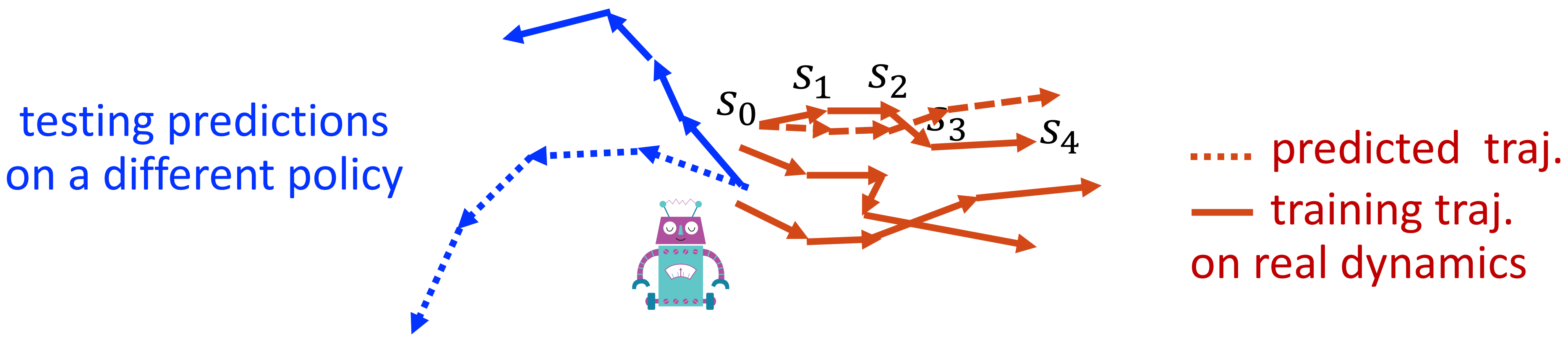
- High-dimensional state and action space
- Non-linear dynamics $M$, policy $\pi$ parameterized by neural networks
- Goal: an analyzable algorithm with # samples polynomial in dimension (assuming some computational oracles)

Prior work

- Finite state space: [Jaksch et al., 2010; Bartlett & Tewari, 2009; Fruit et al., 2018; Lakshmanan et al., 2015; Hinderer, 2005; Pirotta et al., 2015; 2013)
- Linear dynamics: [Abbasi-Yadkori & Szepesvári, 2011; Simchowitz et al., 2018; Dean et al., 2017; Sutton et al., 2012; Tamar et al., 2012]
- Sample complexity result: [Sun et al.'2017]

# Challenges in Analyzing Deep Model-Based RL (Cont'd)

- Issue: the learned dynamics are not accurate for those states unseen in training trajectories
- Exploitation: only go to places that the dynamics is certain
- Exploration: improve the certainty of the model by trying diverse policies



# A Classical Idea: Optimism in the Face of Uncertainty

Repeat:

1. Sample trajectories from real dynamics using current policy

2. policy, dynamics $\leftarrow \operatorname*{argmax}_{\pi, M} V^{\pi,M}$

   s.t., $M$ is consistent with existing trajectories

payoff of $\pi$ on learned dynamics

➢ Explore a policy if it is good for some reasonable dynamics

# A Classical Idea: Optimism in the Face of Uncertainty

Repeat:

1. Sample trajectories from real dynamics using current policy
2. $$\text{policy, dynamics} \leftarrow \underset{\pi,\, M}{\operatorname{argmax}}\, V^{\pi,M}$$
   s.t., $M$ is consistent with existing trajectories

payoff of $\pi$ on learned dynamics

Q1: how do we express the constraint for non-linear models?

- confidence intervals for finite state space or linear models; not feasible for neural nets

# A Classical Idea: Optimism in the Face of Uncertainty

Repeat:

1. Sample trajectories from real dynamics using current policy

2. policy, dynamics $\leftarrow \underset{\pi,\, M}{\operatorname{argmax}}\, V^{\pi,M}$

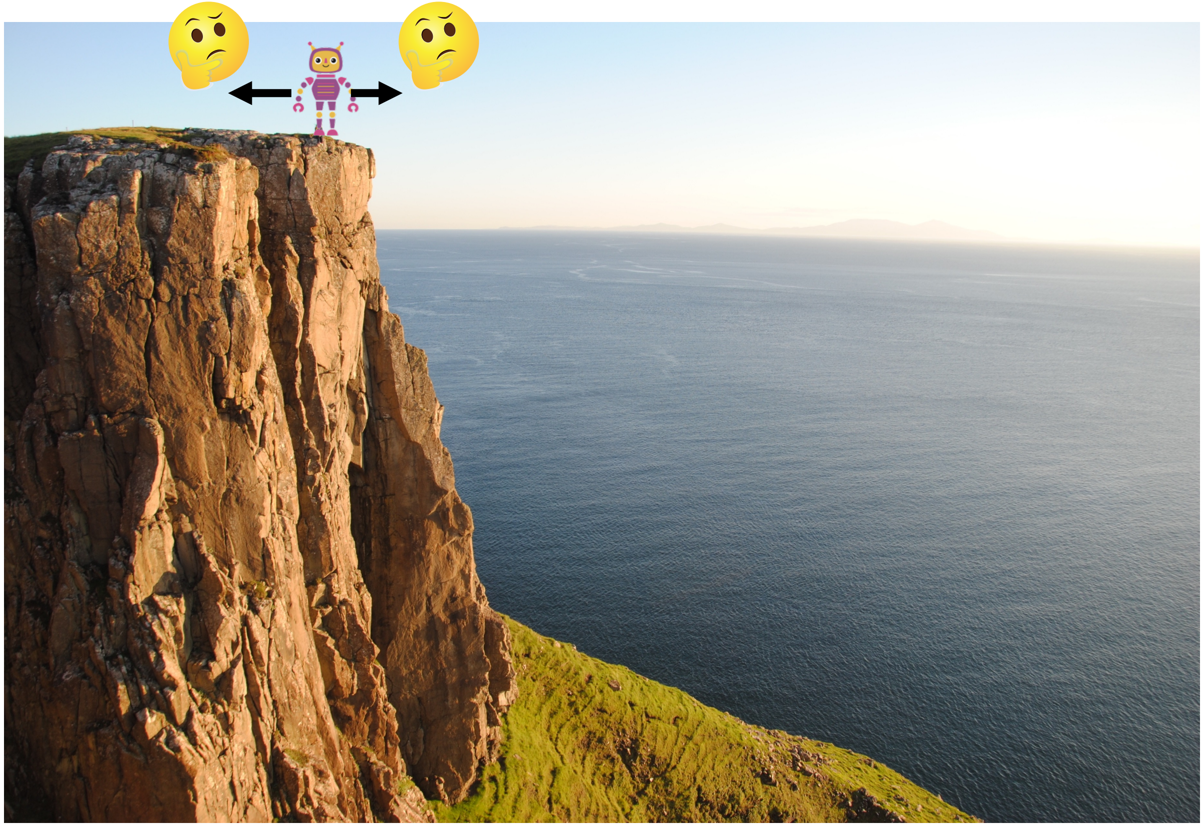
   s.t., $M$ is consistent with existing trajectories

payoff of $\pi$ on learned dynamics

Q2: how do we measure the "consistency"?

- how do we measure the errors of the learned dynamics?

# The Same Prediction Loss Could Mean Very Differently For Different States and Actions



# Our Idea

Ideal loss for $M \approx$ error of predicting future payoff using $M$

$$|V^{\pi,M} - V^{\pi,M^\star}|$$

total payoff on estimated dynamics $M$

total payoff on true dynamics $M^\star$

- Design an upper bound of the ideal loss and use it as a surrogate loss or a consistency measure

$$|V^{\pi,M} - V^{\pi,M^\star}| \leq \mathcal{D}(M,\pi)$$

- A dynamics $M$ has low loss (is consistent with existing data) if $M$ can predict the real reward with small error upper bound $\mathcal{D}(M,\pi)$

[Algorithmic Framework for Model-based Reinforcement Learning with Theoretical Guarantees Luo-Xu-Tian-Darrell-M.'19]

Repeat:

1. Sample trajectories from real dynamics using current policy

2. $$\text{policy, dynamics} \leftarrow \underset{\pi,\, M}{\operatorname{argmax}}\, V^{\pi,M}$$

   s.t., ~~$M$ is consistent with existing trajectories~~

$$\text{s.t., } \mathcal{D}(\pi, M) \leq \epsilon$$

- $\{M : \mathcal{D}(\pi, M) \leq \epsilon\}$ is a confidence region depending on $\pi$ and the reward function
- Next: absorb the constraint in the objective

# Meta-Algorithm for Model-Based RL with Convergence Guarantees

From $k = 1$ to $T$:

1. Sample trajectories using $\pi_k$, build upper bound $\mathcal{D}_{\pi_k}(\pi, M)$
2. $M_{k+1}, \pi_{k+1} = \mathrm{argmax}_{\pi, M} \underbrace{V^{\pi, M} - \mathcal{D}_{\pi_k}(\pi, M)}_{\text{lower bound of real reward } := L(M, \pi)}$

**Theorem:** Assume the model family contains $M^\star$, and the inner optimization is solvable, then,

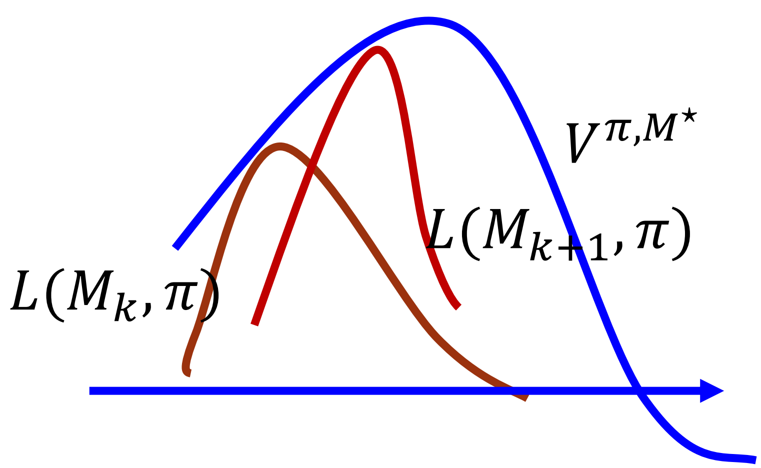
$$V^{\pi_1, M^\star} \le V^{\pi_2, M^\star} \le \cdots \le V^{\pi_T, M^\star} \le \ \ldots$$

and $V^{\pi_k, M^\star}$ converges to a local maximum of $V^{\pi, M^\star}$.

# Meta-Algorithm for Model-Based RL with Convergence Guarantees

From $k = 1$ to $T$:

1. Sample trajectories using $\pi_k$, build upper bound $\mathcal{D}_{\pi_k}(\pi, M)$
2. $$M_{k+1}, \pi_{k+1} = \text{argmax}_{\pi, M} \underbrace{V^{\pi, M} - \mathcal{D}_{\pi_k}(\pi, M)}_{\text{lower bound of real reward } := L(M, \pi)}$$



# Optimizable Upper Bounds of Ideal Loss

- Design an upper bound of the ideal loss

$$|V^{\pi,M} - V^{\pi,M^\star}| \le \mathcal{D}_{\pi_{\mathrm{ref}}}(M,\pi)$$

Lemma:

$$|V^{\pi,M} - V^{\pi,M^\star}| \le \mathop{\mathbb{E}}_{(s,a,s')\sim\pi,M^\star}\left[|V^{\pi,M}(M(s,a)) - V^{\pi,M}(s')|\right]$$

- $(s,a,s')$ shorthand for $(s_t,a_t,s_{t+1})$
- Issue: requires samples from the environment to estimate the loss

- $V^{\pi,M}(s) := $ the total payoff of the policy on dynamics $M$ starting from state $s$
- $V^{\pi,M} = \mathbb{E}_{s_0}[V^{\pi,M}(s_0)]$

# Optimizable Upper Bounds of Ideal Loss (Cont'd)

Improved Lemma:

$\forall \pi$ that is close to $\pi_{\text{ref}}$ :

$$|V^{\pi,M} - V^{\pi,M^\star}| \le \underbrace{\mathop{\mathbb{E}}_{(s,a,s')\sim\pi_{\text{ref}},M^\star}\left[|V^{\pi,M}(M(s,a)) - V^{\pi,M}(s')|\right]}_{:=\mathcal{D}_{\pi_{\text{ref}}}(M,\pi)}$$

- Upper bound can be re-used if $\pi$ doesn't change much
- Recovers the norm-based loss, if $V^{\pi,M}$ is Lipschitz w.r.t $||\cdot||$

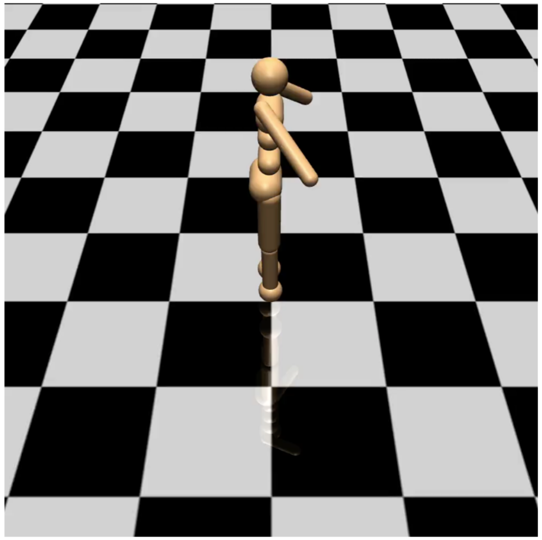
$$\mathcal{D}_{\pi_{\text{ref}}}(M,\pi) \lesssim \mathop{\mathbb{E}}_{(s,a)\sim\pi_k,M^\star}\|M(s,a) - s'\|$$

No square compared to MSE

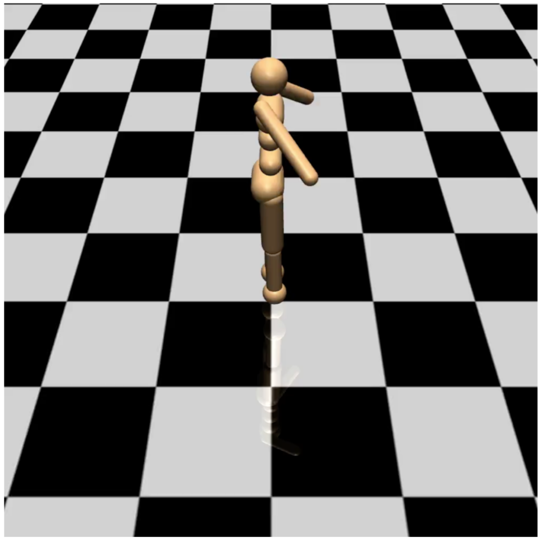
- Inspires a practical algorithm (SLBO) that uses $\ell_2$ loss (not MSE) and optimizes the objective with SGD
  - no optimism is practically needed though

# Demo: learning to walk to the right as fast as possible

- What the learned dynamics predicts



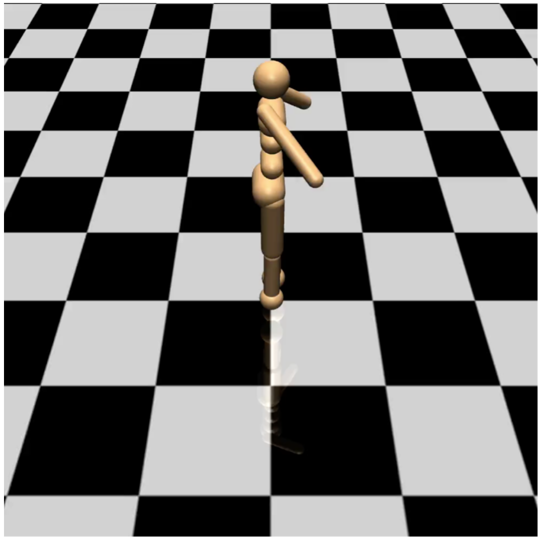
- What the humanoid does in reality

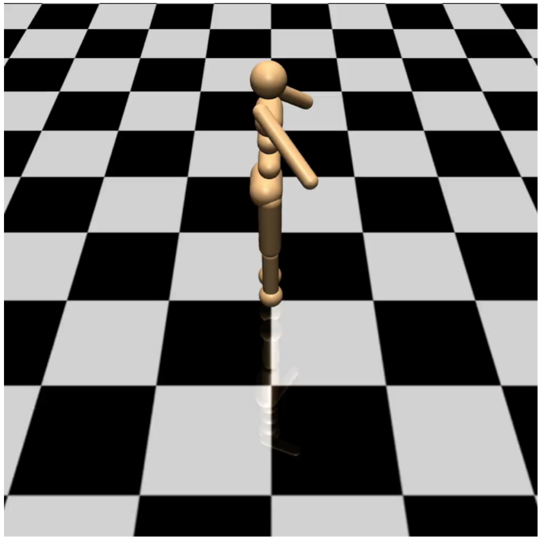


Iteration 10

# Demo: learning to walk to the right as fast as possible

- What the learned dynamics predicts
- What the humanoid does in reality

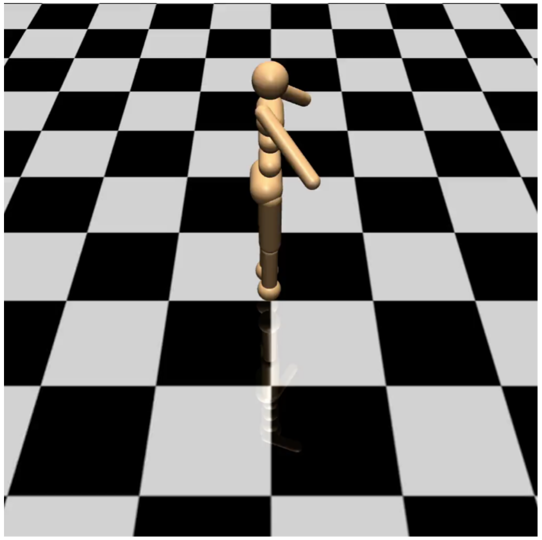




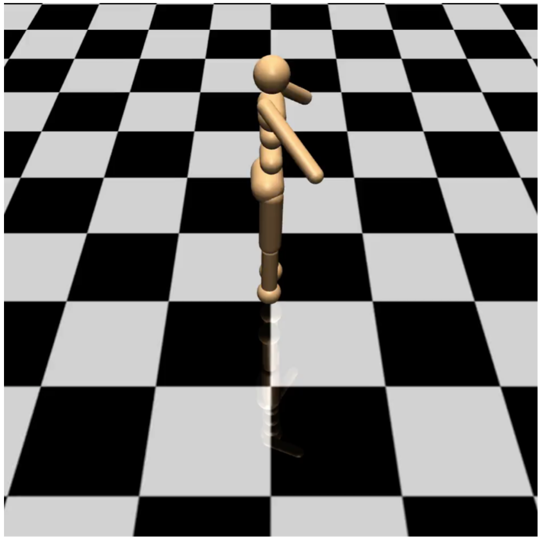
Iteration 20

# Demo: learning to walk to the right as fast as possible

- What the learned dynamics predicts

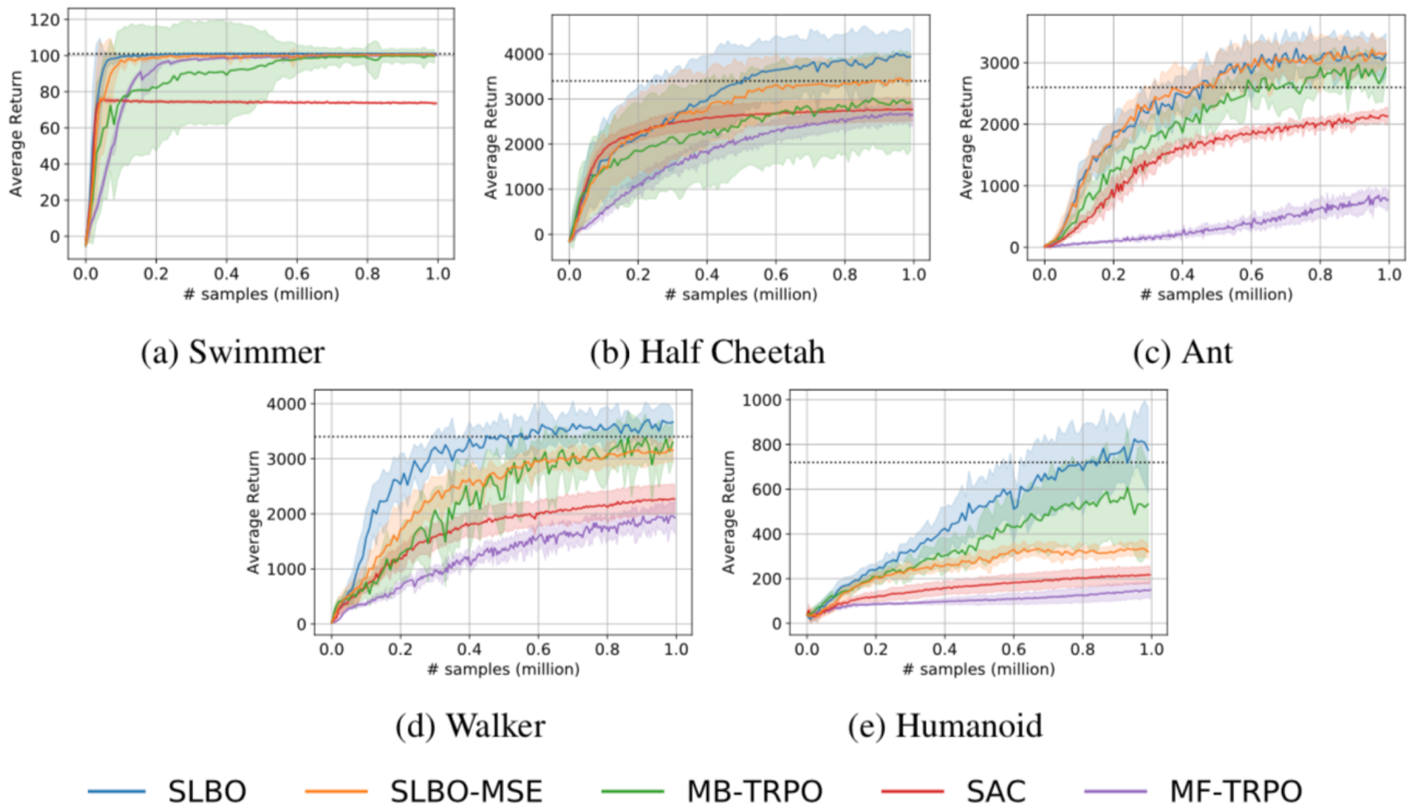


- What the humanoid does in reality



Iteration 210

# Evaluations on MuJoCo Benchmark Tasks




(a) Swimmer (b) Half Cheetah (c) Ant

(d) Walker (e) Humanoid

- ➢ Outperforms prior works when 1M (or fewer) samples are permitted

[Algorithmic Framework for Model-based Reinforcement Learning with Theoretical Guarantees Luo-Xu-Tian-Darrell-M.'19]

# Follow-up: Model-based Multi-task RL

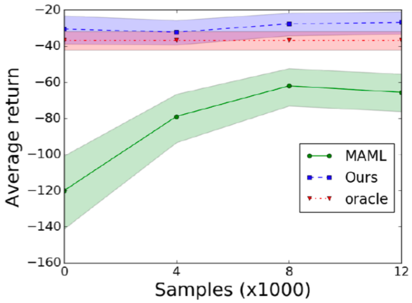
Setting: A single robot, but multiple tasks

- e.g., humanoid runs with different speeds and directions

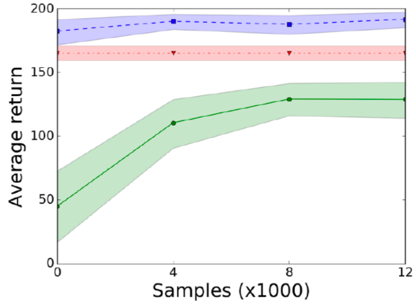
- Our algo.: learns a dynamics shared across tasks sequentially
  - Amortized sample costs over tasks

- Prior work: MAML (model-agnostic meta-learning)
  - Learns a shared policy that can be adapted to tasks

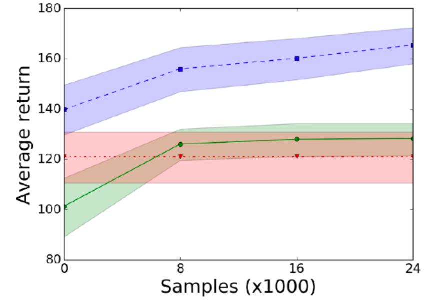
[A Model-based Approach for Sample-efficient Multi-task Reinforcement Learning
Landolfi-Thomas-M.'19]



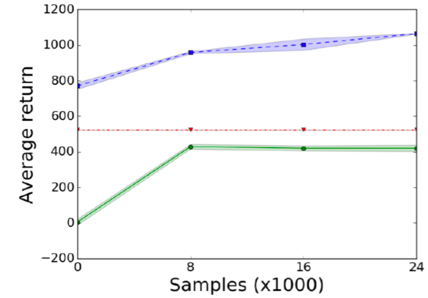

(a) Cheetah $[0, 2]$



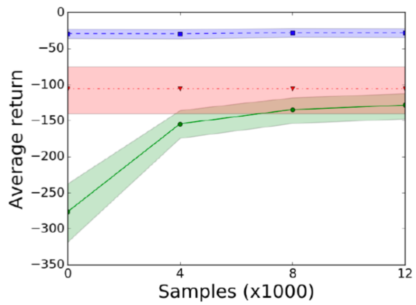

(b) Ant $[0, 3]$



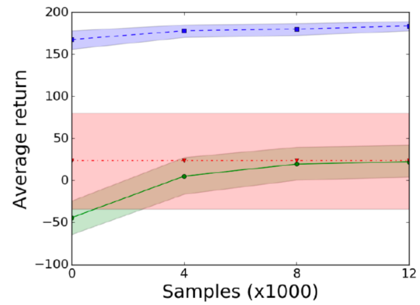

(c) Humanoid $[0, 1.5]$



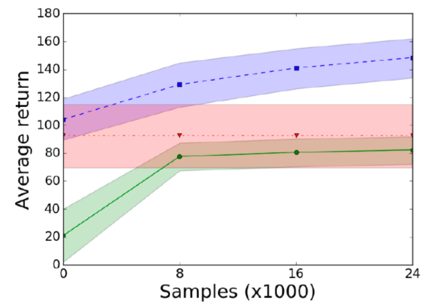

(d) Ant forward/back



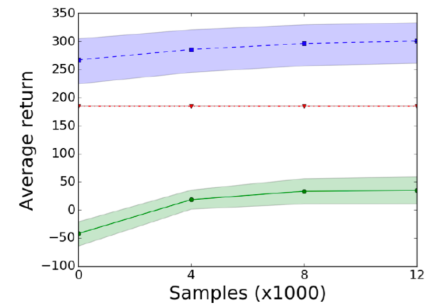

(e) Cheetah $[-2, 2]$




(f) Ant $[-3, 3]$




(g) Humanoid $[-1.5, 1.5]$




(h) Ant $[-3, 3]^2$

Tasks: running with random velocities in the interval/region

[A Model-based Approach for Sample-efficient Multi-task Reinforcement Learning
Landolfi-Thomas-M.'19]

# Imitation Learning

- RL from scratch alone may still be not sample-efficient enough
- Imitation learning: learning from (human) experts demonstration

Formulation:

- experts run a policy $\pi_e$ to collect trajectories

$$\mathcal{R} = \left\{ \left( s_0^{(i)}, a_0^{(i)}, \dots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, s_T \right) \right\}_{i=1}^{n}$$

- we learn a policy from $\mathcal{R}$ with or without additional samples

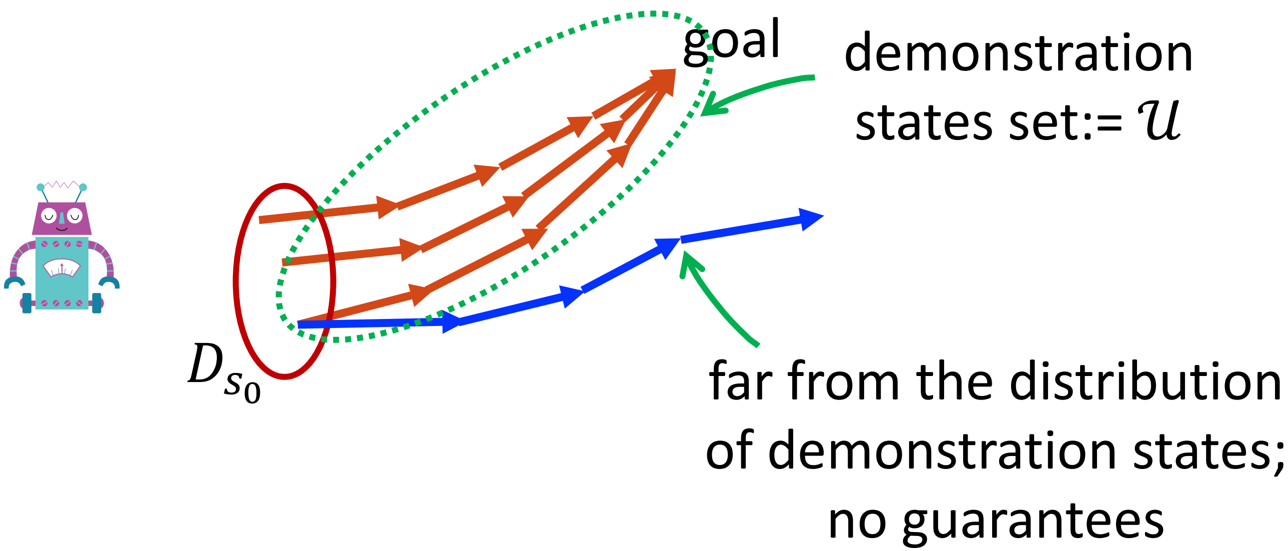
# A Classic Algorithm: Behavioral Cloning

- Supervised learning on demonstrations

  $$\mathcal{R} = \left\{\left(s_0^{(i)}, a_0^{(i)}, \ldots, s_{T-1}^{(i)}, a_{T-1}^{(i)}, s_T\right)\right\}_{i=1}^{n}$$

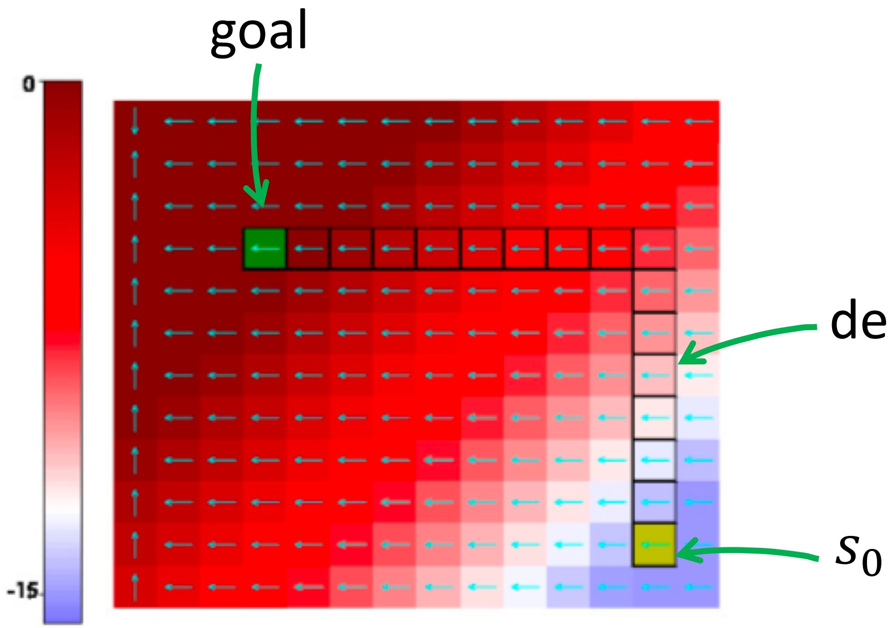
  - Fit a policy $\pi_{\mathrm{BC}}$ such that $\pi_{\mathrm{BC}}\left(s_t^{(i)}\right) \approx a_t^{(i)}$

- Well-known issue: distribution drift and cascading errors



# Another Attempt: Learning Value Functions from Demonstrations

- Recall $V^{\pi_e}(s) :=$ total payoff of expert policy starting from $s$
- If $s \in$ demonstration states $\mathcal{U}$, we know $V^{\pi_e}(s)$
- Attempt: learn $V^{\pi_e}$ by supervised learning on $\mathcal{U}$
- Same issue: $V^{\pi_e}$ extrapolates falsely outside $\mathcal{U}$

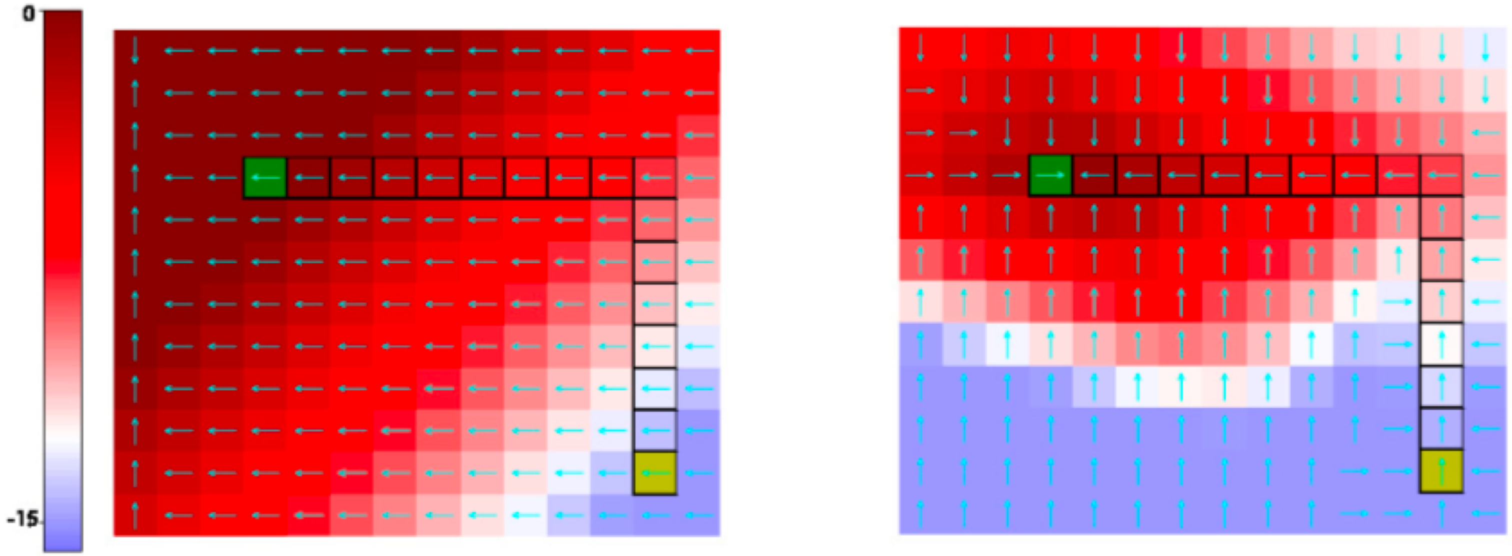



- Correct values on $\mathcal{U}$
- Wrong values outside $\mathcal{U}$

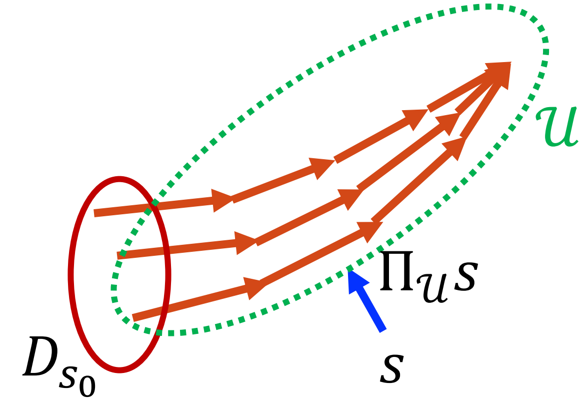
# Our Idea: Learning a Better Value Function (and Use it Correct Mistakes of Behavioral Cloning)

- Key: the value $V^{\pi_e}(s)$ should be relatively smaller for $s \notin \mathcal{U}$
- $\Rightarrow$ following the value function leads us back to $\mathcal{U}$

falsely-extrapolated conservatively-extrapolated



# Theoretical Results




- Conservatively-extrapolated value function $V$:

$$V(s) = V^{\pi_e}(s) \pm \delta_V, \qquad \text{if } s \in \mathcal{U}$$
$$V(s) = V^{\pi_e}(\Pi_{\mathcal{U}}(s)) - \lambda \|s - \Pi_{\mathcal{U}}(s)\| \pm \delta_V \qquad \text{if } s \notin \mathcal{U}$$

**Theorem** (informal): Assume the access to an approximate model $M$. Then, the policy induced from a conservatively-extrapolated value $V$ (below) stays close to $\mathcal{U}$ and has good performance:

$$\pi(s) = \operatorname*{argmax}_a V(M(s,a))$$

[Learning Self-Correctable Policies and Value Functions from Demonstrations with Negative Sampling Luo-Xu-M.'19]

# Theoretical Results (Cont'd)

- Note: dynamics may be hard to learn from demonstrations (can only expect it to work around expert actions)

model $M$ approximately correct near the demonstration

**Theorem** (informal): Assume the access to ~~an approximate model $M$~~.

Then, the policy induced from a conservatively-extrapolated value $V$ (below) stays close to $\mathcal{U}$ and has good performance:

$$\pi(s) \cancel{= \operatorname*{argmax}_a V(M(s,a))}$$

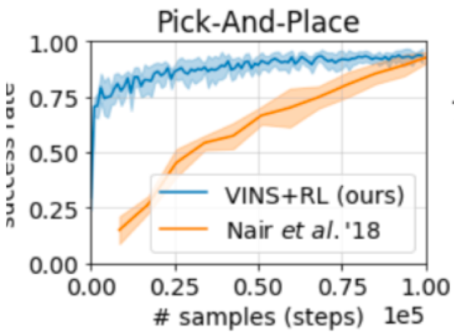
$$\pi(s) = \operatorname*{argmax}_{a:||a-\pi_{BC}(s)||\leq\epsilon} V(M(s,a))$$

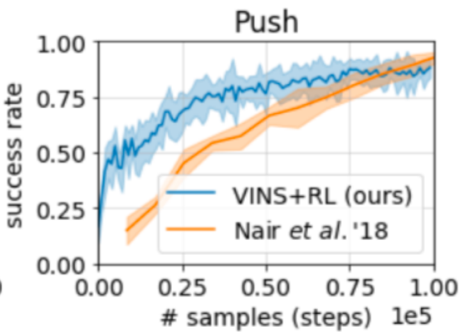
# Experiments

- Learn conservatively-extrapolated value functions with a heuristic borrowed from NLP, negative sampling

- Initialize an RL algorithm (that takes additional samples) with value function, policy, and dynamics learned from demonstrations

| | VINS (ours) | BC |
|---|---|---|
| Reach 100 | **$100 \pm 0\%$** | **$100 \pm 0\%$** |
| Pick 100 | **$75.7 \pm 1.0\%$** | $66.8 \pm 1.1\%$ |
| Pick 200 | **$84.0 \pm 0.5\%$** | $82.0 \pm 0.8\%$ |
| Push 100 | **$44.0 \pm 1.5\%$** | $37.3 \pm 1.1\%$ |
| Push 200 | **$55.2 \pm 0.7\%$** | $51.3 \pm 0.6\%$ |




[Learning Self-Correctable Policies and Value Functions from Demonstrations with Negative Sampling Luo-Xu-M.'19]

# Summary

- Convergence guarantees for a meta model-based RL algorithm
  - Reward-aware loss for learning dynamics
  - SLBO: a much simplified instantiation of the meta-algorithm that works well empirically
- Model-based multi-task RL
- Learning self-correctable policy via learning conservatively-extrapolated value functions
- Open questions:
  - How to empirically leverage optimism in model-based RL?
  - How to customize algorithms for particular environments?
  - How to apply dynamical models to other settings (e.g., hierarchical RL)?

**Thank you!**