

Predictor-Corrector Policy Optimization (PicCoLO)

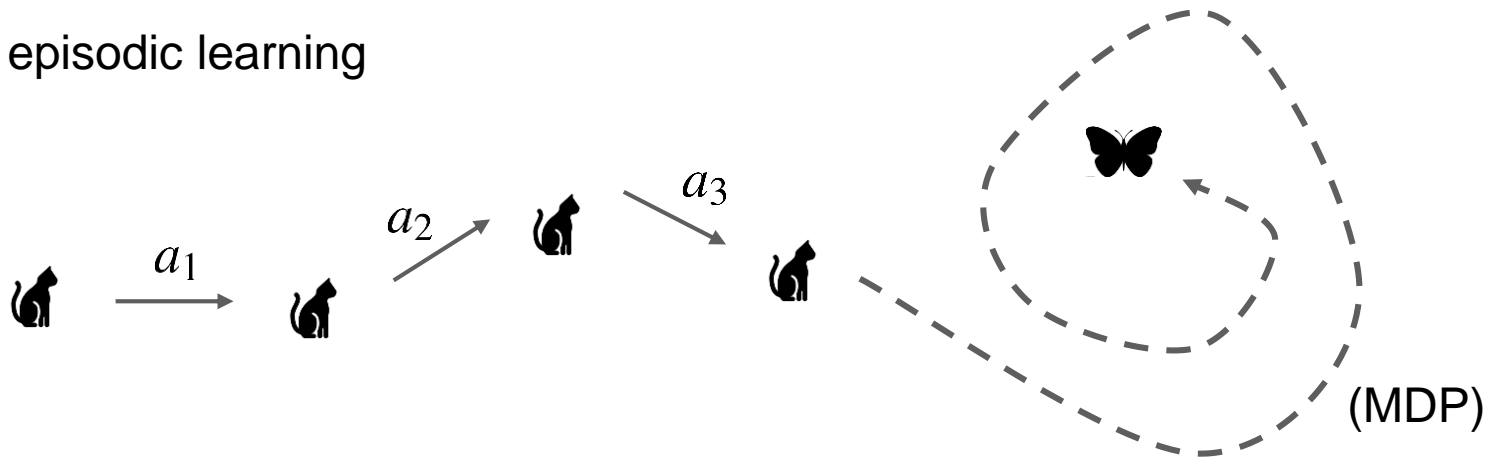


Ching-An Cheng^{#\$}, Xinyan Yan[#], Nathan Ratliff^{\$}, Byron Boots^{#\$}

Jun 11 2019 @ ICML

Policy optimization

We consider episodic learning



Optimize a policy $a_t \sim \pi(a_t | s_t)$ for sequential decision making

$$\min_{\pi \in \Pi} J(\pi), \quad J(\pi) = (1 - \gamma) \mathbb{E}_{s_0, a_0, s_1, \dots | \pi} \left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \right]$$

Learning efficiency

- Cost of Interactions $>$ cost of computation
 - learning efficiency = sample efficiency
 - we should maybe spend time on planning before real interactions

to do so we need models, *but should we?*



Why we should use models

- A way to summarize prior knowledge & past experiences
- Can optimize the policy indirectly without costly real-world interactions
- Can be provably more sample-efficient (Sun et al., 2019)



airsim



flex



Miniatur Wunderland

Why we should NOT use models

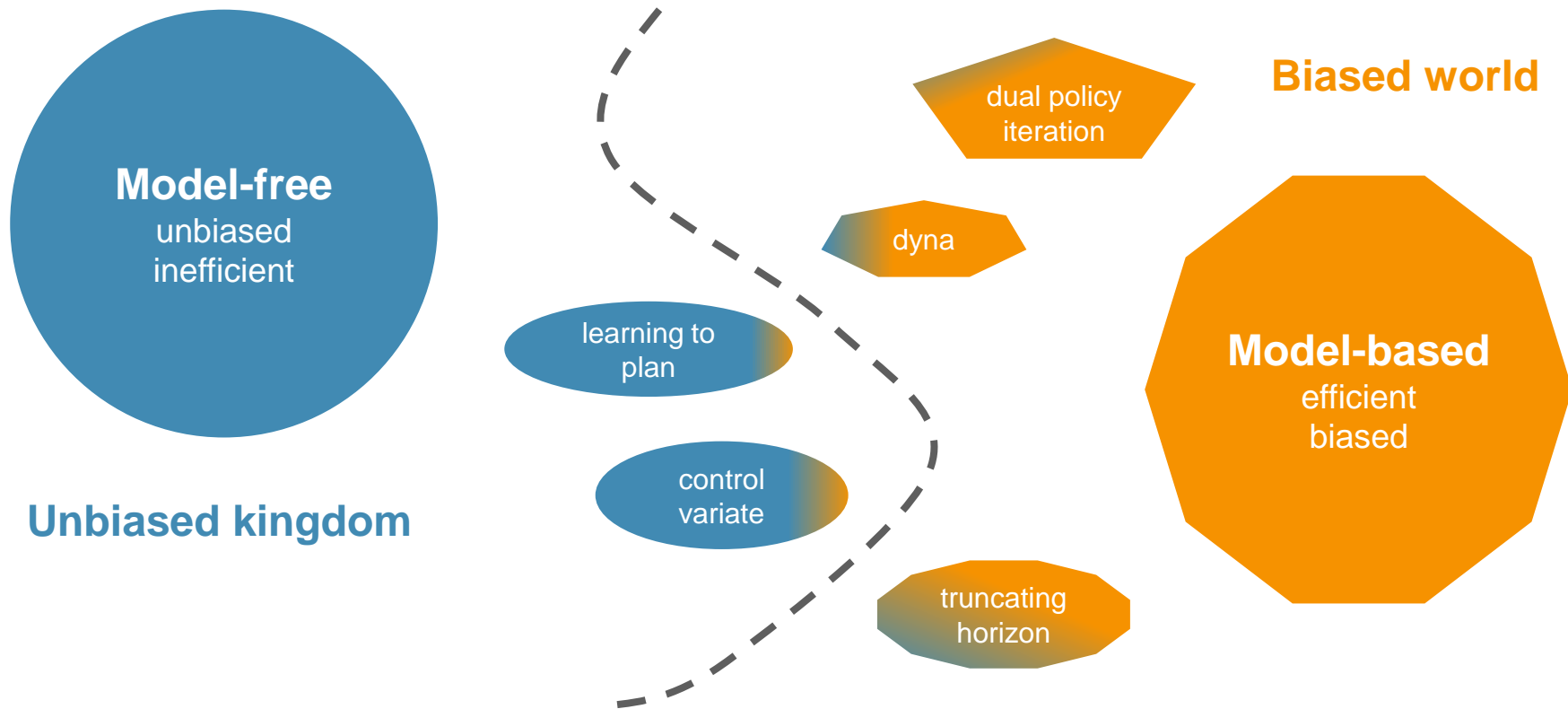
- Models are, by definition, *inexact*
- Weakness of the model can be exploited in policy optimization
- Result in biased performance of the trained policy

"The reality gap"

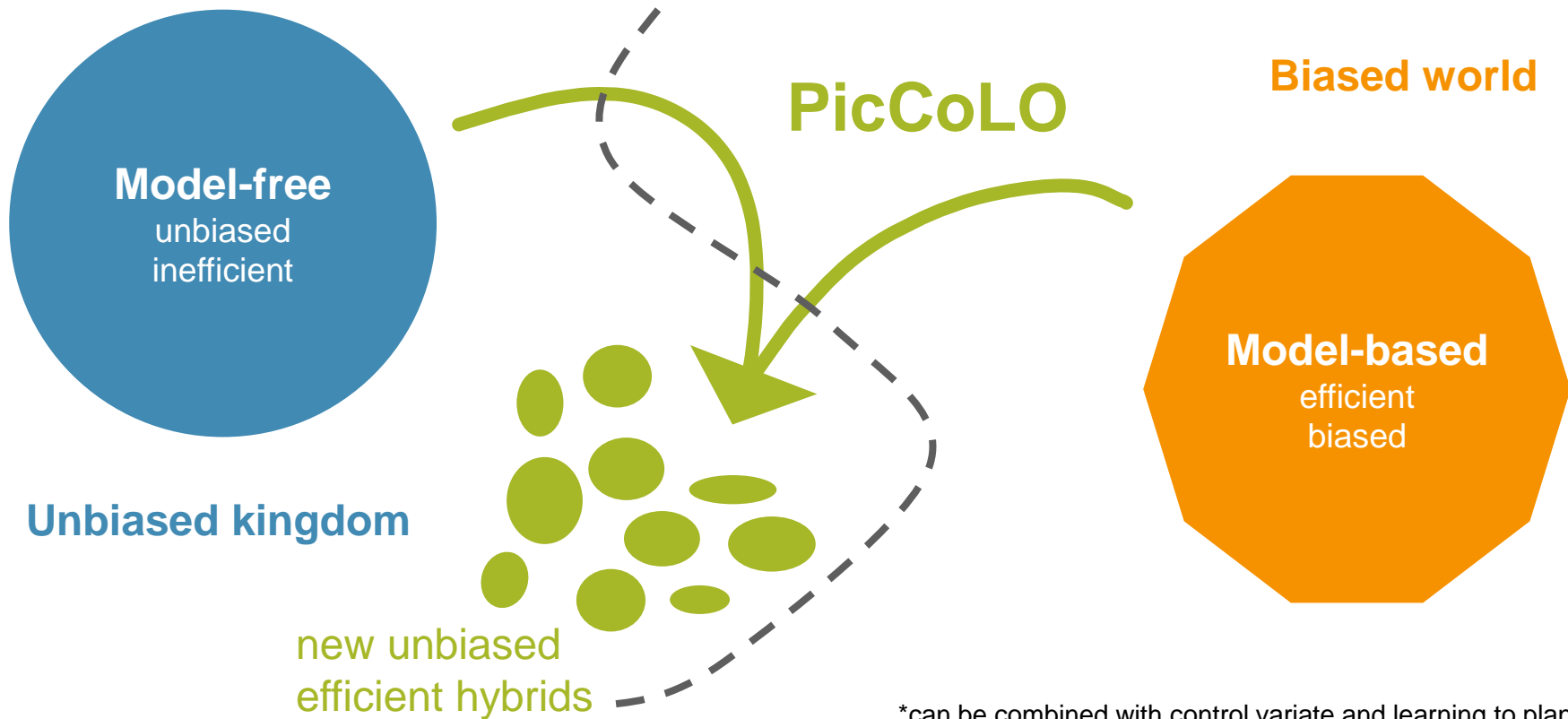


ItsJerryAndHarry (Youtube)

Toward reconciling both sides



A new direction



*can be combined with control variate and learning to plan

A new direction

- Main idea

We should not fully trust a model (e.g. the methods in the biased world) but leverage only the correct part

- How?

1. Frame policy optimization as predictable online learning (POL)
2. Design a reduction-based algorithm for POL to reuse known algorithms
3. When translated back, this gives a meta-algorithm for policy optimization

Online learning



LEARNER

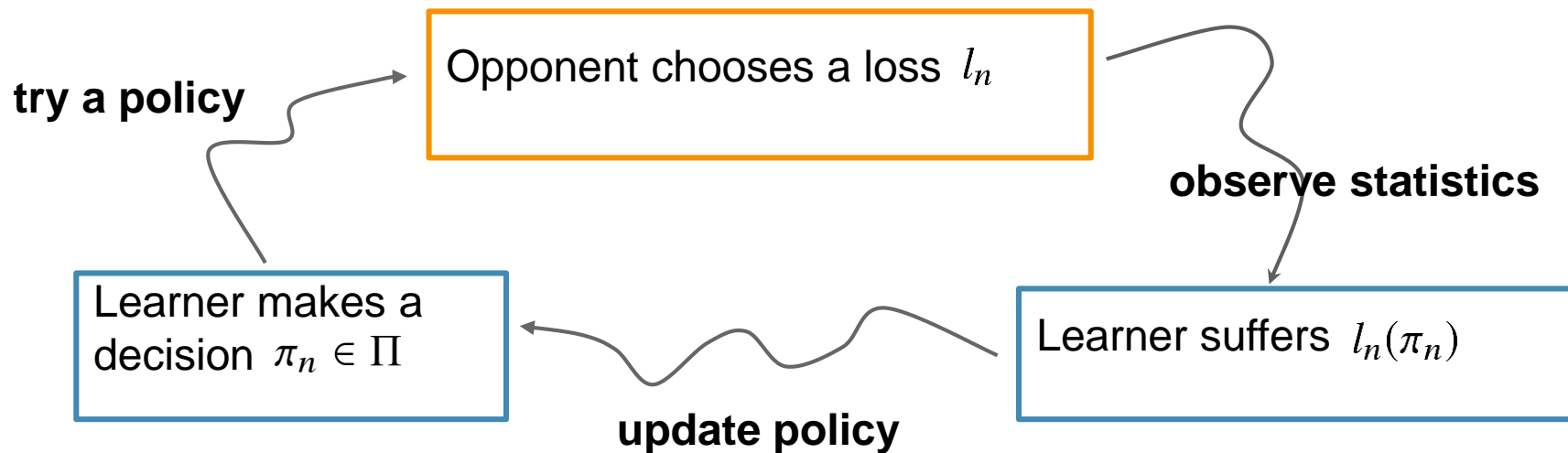


OPPONENT

policy optimization algorithm



Online learning



Online learning

- Loss sequence can be adversarially chosen by the opponent
- Common performance measure

$$\text{Regret}_N = \sum_{n=1}^N l_n(\pi_n) - \min_{\pi \in \Pi} \sum_{n=1}^N l_n(\pi)$$

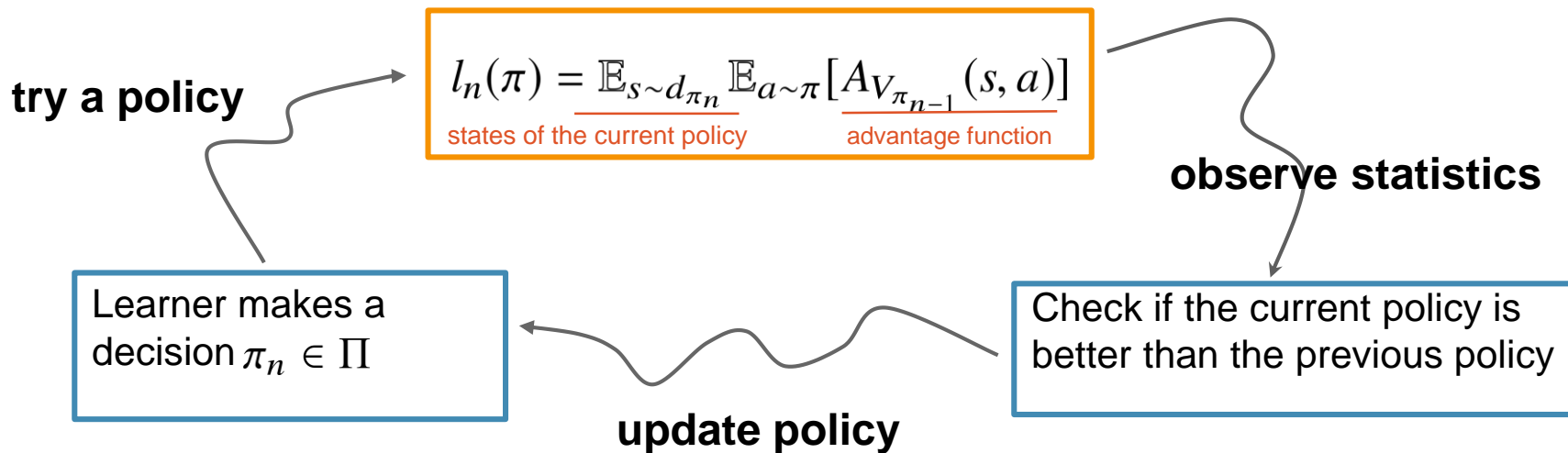
- For convex losses, algorithms with sublinear regret are well-known
e.g. mirror descent, follow-the-regularized-leader, etc.



Policy optimization as online learning

- Define online losses such that sublinear regret implies policy learning
- This idea started in the context of imitation learning (Ross et al., 2011)
- We show that episodic policy optimization can be viewed similarly

Policy optimization as online learning



The gradient of this loss is the actor-critic gradient (implemented in practice)



Possible algorithms

- We can try typical no-regret algorithms
 - e.g. mirror descent in online learning -> actor-critic update
- But it turns out they are not optimal. We can actually learn faster!
- Insight

loss functions here are not adversarial but can be inferred from the past

but these typical algorithms were designed for adversarial setups

e.g. similar policies visit similar states

$$l_n(\pi) = \mathbb{E}_{s \sim d_{\pi_n}} \mathbb{E}_{a \sim \pi} [A_{V_{\pi_{n-1}}}(s, a)]$$

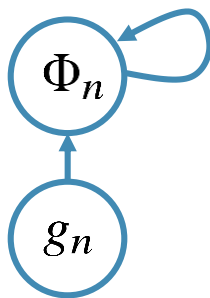
Predictability and predictive models

- We can view predictability, e.g., as the ability to predict future gradients

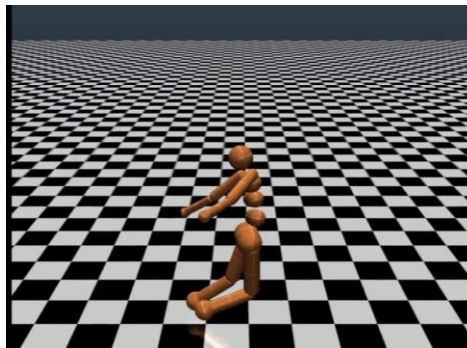
- **Predictive model:** a function that estimates **gradient of future loss**

$$\Phi_n(\pi) \approx \nabla l_n(\pi)$$

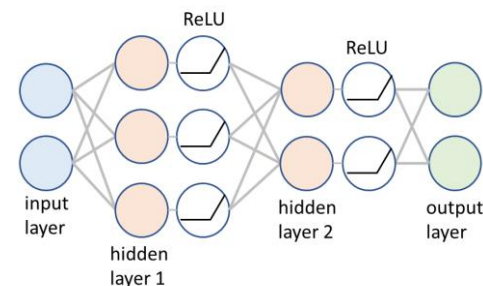
- Examples



(averaged) past gradients
replay buffer



inexact simulator



function approximator

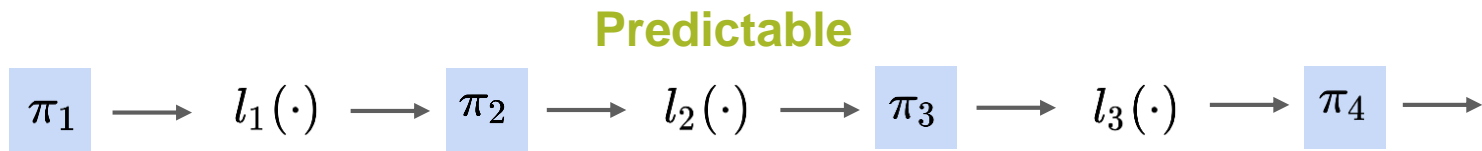
Policy optimization is **predictable** online learning

- We need algorithms that consider predictability
- There are two-step algorithms for predictable setups, but
- We have more sophisticated algorithms for adversarial problems, but ...
- That is, we need *a reduction from predictable to adversarial problems*

This is PicCoLO



The idea behind PicCoLO

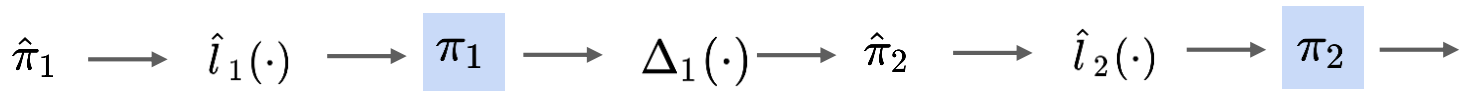


It suffices to consider

$$\hat{l}_n(\pi) = \langle \hat{g}_n, \pi \rangle$$

if we wisely select \hat{g}_n
via predictive model $\Phi_n(\cdot)$

$$\begin{aligned} l_n(\cdot) &= \hat{l}_n(\cdot) + (l_n - \hat{l}_n)(\cdot) \\ &= \underbrace{\hat{l}_n(\cdot)}_{\text{prediction}} + \underbrace{\Delta_n(\cdot)}_{\text{error}} \end{aligned}$$



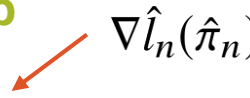
Adversarial

PicCoLO is a meta algorithm

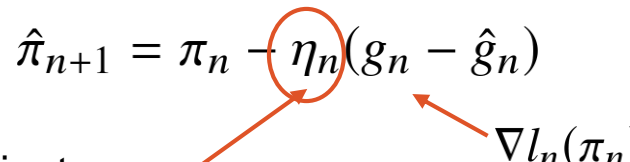
$$\hat{\pi}_1 \longrightarrow \hat{l}_1(\cdot) \longrightarrow \pi_1 \longrightarrow \Delta_1(\cdot) \longrightarrow \hat{\pi}_2 \longrightarrow \hat{l}_2(\cdot) \longrightarrow \pi_2 \longrightarrow$$

Apply a standard method (e.g. gradient descent) to this new sequence

Prediction Step

$$\pi_n = \hat{\pi}_n - \eta_{n-1} \hat{g}_n$$


Correction Step

$$\hat{\pi}_{n+1} = \pi_n - \eta_n (g_n - \hat{g}_n)$$


trick: adapt steps size based on the size of gradient error
take larger steps when the prediction is accurate and vice versa

(PicCoLO can use control variate to reduce further the variance of g_n, \hat{g}_n)

PicCoLO

Prediction Step

$$\pi_n = \hat{\pi}_n - \eta_{n-1} \hat{g}_n$$

Correction Step

$$\hat{\pi}_{n+1} = \pi_n - \eta_n (g_n - \hat{g}_n)$$

Same idea applies to *any* algorithm in the family of (adaptive) mirror descent and Follow-the-Regularized-Leader

PicCoLO recovers existing algorithms, e.g. extra-gradient, optimistic mirror descent, and provides their adaptive generalization

PicCoLO

Prediction Step

$$\pi_n = \hat{\pi}_n - \eta_{n-1} \hat{g}_n$$

Correction Step

$$\hat{\pi}_{n+1} = \pi_n - \eta_n (g_n - \hat{g}_n)$$

Theoretically we can show

$$\text{Regret}_N = O\left(\sqrt{\sum_{n=1}^N \|g_n - \hat{g}_n\|_{n,*}^2}\right)$$

- the performance is **unbiased**, even when the prediction (model) is incorrect
- learning **accelerates**, when the prediction is relatively accurate

How to compute the prediction \hat{g}_n

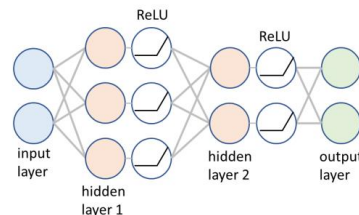
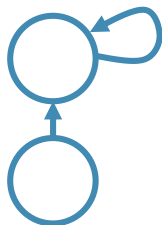
Prediction Step

$$\pi_n = \hat{\pi}_n - \eta_{n-1} \hat{g}_n$$

Correction Step

$$\hat{\pi}_{n+1} = \pi_n - \eta_n (g_n - \hat{g}_n)$$

- We want to set $\hat{g}_n \approx g_n$ because $\text{Regret}_N = O\left(\sqrt{\sum_{n=1}^N \|g_n - \hat{g}_n\|_{n,*}^2}\right)$
- We can use predictive model $\Phi_n(\pi) \approx \nabla l_n(\pi)$ to realize this!



How to compute the prediction \hat{g}_n

Prediction Step

$$\pi_n = \hat{\pi}_n - \eta_{n-1} \hat{g}_n$$

Correction Step

$$\hat{\pi}_{n+1} = \pi_n - \eta_n (g_n - \hat{g}_n)$$

- We want to set $\hat{g}_n \approx g_n$ because $\text{Regret}_N = O\left(\sqrt{\sum_{n=1}^N \|g_n - \hat{g}_n\|_{n,*}^2}\right)$
- Because $\Phi_n(\pi) \approx \nabla l_n(\pi)$ and $g_n = \nabla l_n(\pi_n)$, we can set $\hat{g}_n = \Phi_n(\pi_n)$
- We can select \hat{g}_n by solving a fixed-point problem (FP)

Prediction Step

$$\pi_n = \hat{\pi}_n - \eta_{n-1} \Phi(\pi_n)$$



How to compute the prediction \hat{g}_n

Prediction Step

$$\pi_n \approx \hat{\pi}_n - \eta_{n-1} \Phi_n(\pi_n)$$

Correction Step

$$\hat{\pi}_{n+1} = \pi_n - \eta_n (g_n - \hat{g}_n)$$

- When $\Phi_n(\pi) = \nabla \hat{J}_n(\pi)$, the FP becomes an optimization problem:

$$\pi_n = \arg \min_{\pi \in \Pi} \hat{J}_n(\pi) + \frac{1}{\eta_{n-1}} \|\pi - \hat{\pi}_n\|^2$$

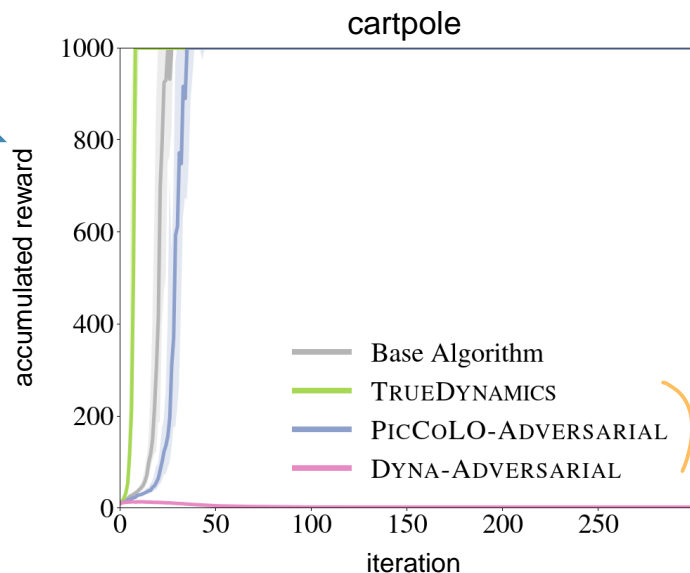
this is regularized optimal control

- Heuristic: set $\hat{g}_n = \Phi_n(\hat{\pi}_n)$ or just do a few iterations



Experiments

- For example, with ADAM as the base algorithm



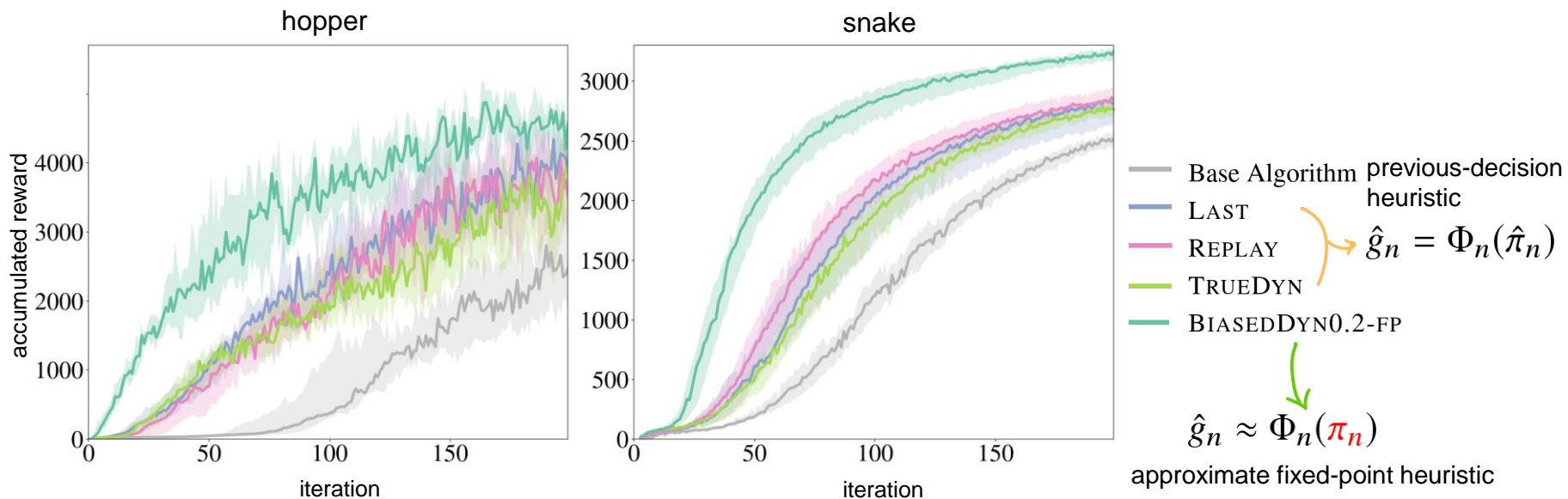
shows acceleration when predictions are accurate
robust against model error

$\hat{g}_n = \Phi_n(\hat{\pi}_n)$ previous-decision heuristic

similar properties observed for other base algorithms (e.g. natural gradient descent, TRPO)

Experiments

- For example, with ADAM as the base algorithm



The fixed-point formulation converges even faster

Summary

- “PicCoLOed” model-free algorithms can learn faster without bias
- The predictive model can be viewed as a unified interface for injecting prior

$$\Phi_n(\pi) \approx \nabla l_n(\pi)$$

learning and parameterizing predictive models are of practical importance

- As PicCoLO is designed for general predictable online learning, we expect applications to other problems and domains



Thanks for attention

Please come to our poster # 106