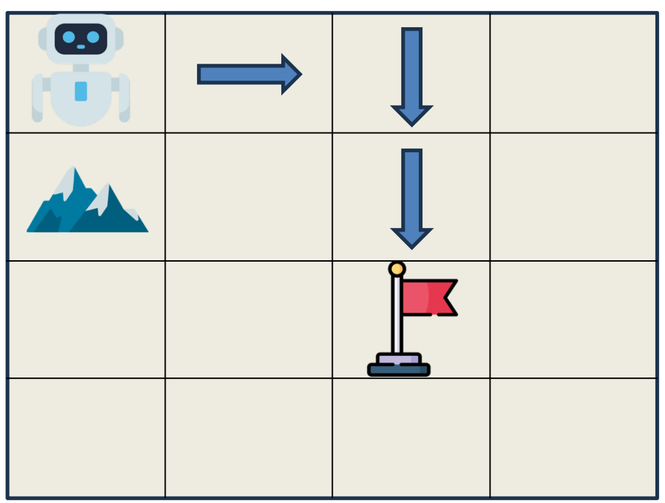
Provably Efficient Actor-Critic for Low-Rank MDPs

Ruiquan Huang
University of Kentucky

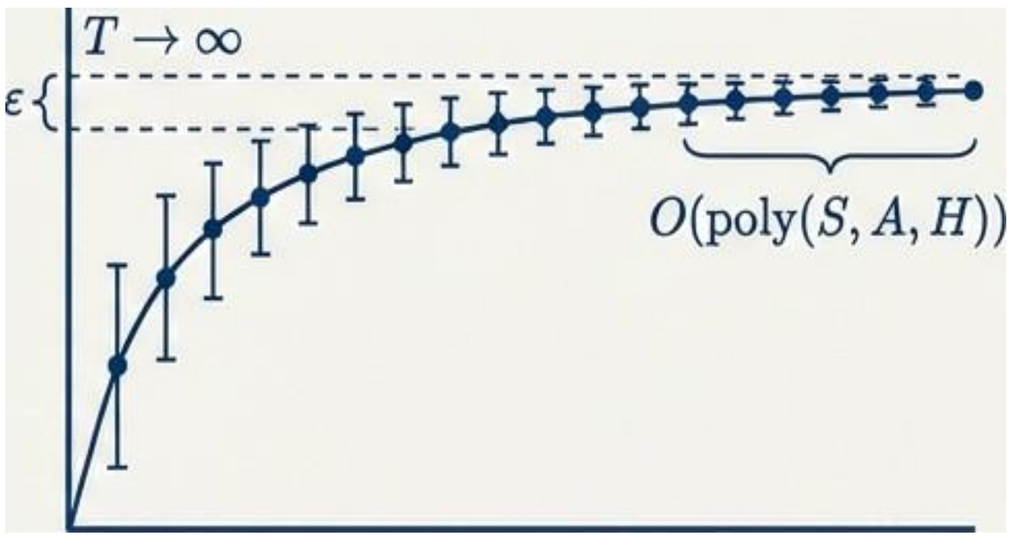


The Divide in Reinforcement Learning

Practice: Deep RL

- (Relative) High empirical performance (Go, Video Games, Self-driving)
- Implementable
- Sample inefficient (No guarantee), Black-box behavior

Theory: Provable RL

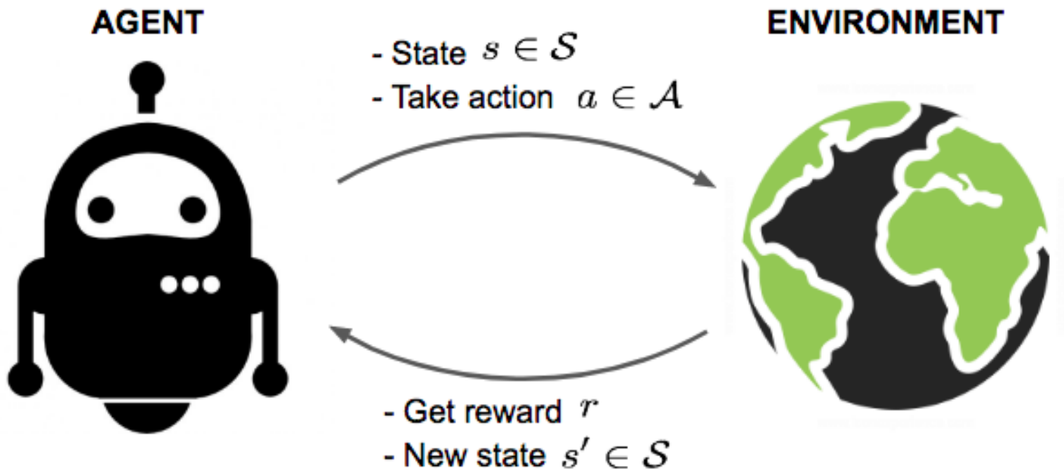


- Polynomial sample complexity bound
- Either simple environment,
- or intractable assumptions, not implementable

Can we design an algorithm that operates in complex environments, enjoys polynomial sample complexity bound, and is also easy to implement?

RL Preliminaries

- Policy: $\pi_h(a|s)$



- Transition: $P_{\theta,h}(s'|s,a)$
- Reward: $r_h(s,a)$
- Horizon: $1 \leq h \leq H$

Value function: $V^{\pi}_{r,h}(s) = \sum_a Q^{\pi}_{r,h}(s,a)\pi_h(a|s)$

Q-value function: $Q^{\pi}_{r,h}(s,a) = r_h(s,a) + \sum_{s'} P_h(s'|s,a)V^{\pi}_{r,h+1}(s')$

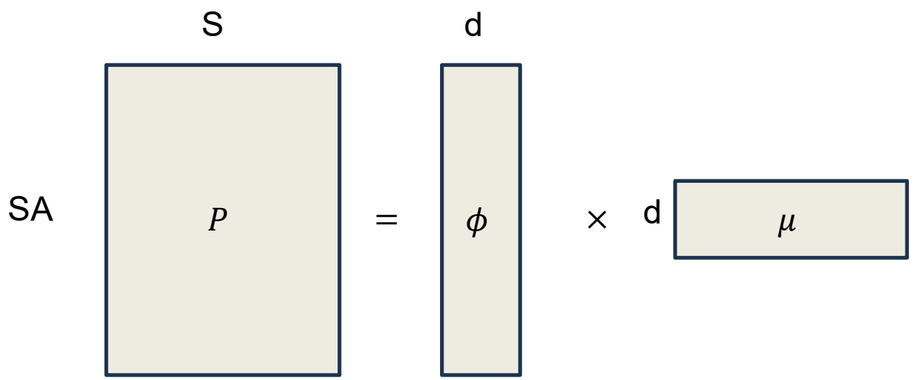
Goal: find $\hat{\pi}$ $s.t.$ $\max_{\pi} V^{\pi}_{1,r}(s_1) - V^{\hat{\pi}}_{1,r}(s_1) \leq \epsilon$

Low-rank MDPs

Tabular RL

S, A: finite

Low-rank RL



$$P_{\theta,h}(s'|s,a) = \phi_{\theta,h}(s,a)^{\top}\mu_{\theta,h}(s')$$

Unkown features parameterized by θ

Latent dim d <= S

General Function

Value functions belong to $\mathcal{F}$ whose “eluder dim” is small

Computation Oracles

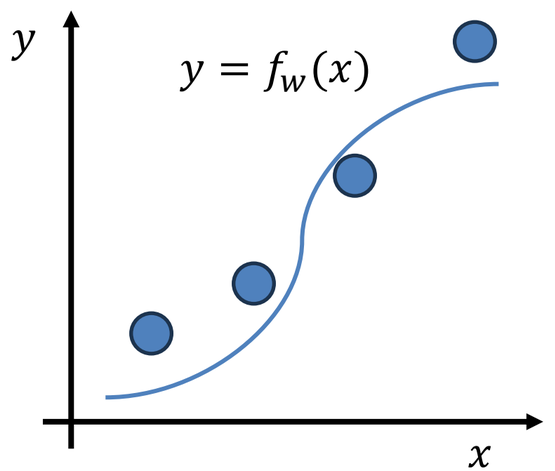
- # of arithmetic operations: inadequate for generation function approximation
- Supervised learning (SL): widely used in deep learning

$SL(\varepsilon)$: the complexity of finding an ε-optimal solution in SL

e.g. gradient descent iterations

Definition 3.2 (Supervised Learning Oracle)**.** A supervised learning oracle $\mathsf{SL}(\varepsilon)$ takes as input a parameter space $\mathcal{W}$, a data distribution (sampling oracle) ρ over input-output pairs (x, y), and a loss function $\ell : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}_+$. It returns a parameter $\hat{w} \in \mathcal{W}$ such that

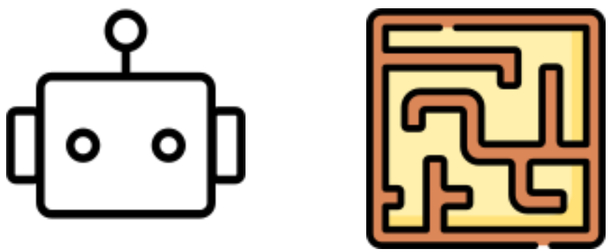
$$\mathbb{E}_{(x,y)\sim\rho}[\ell(f_{\hat{w}}(x), y)] \leq \inf_{w\in\mathcal{W}} \mathbb{E}_{(x,y)\sim\rho}[\ell(f_w(x), y)] + \varepsilon.$$



Supervised Learning

Computation Oracles

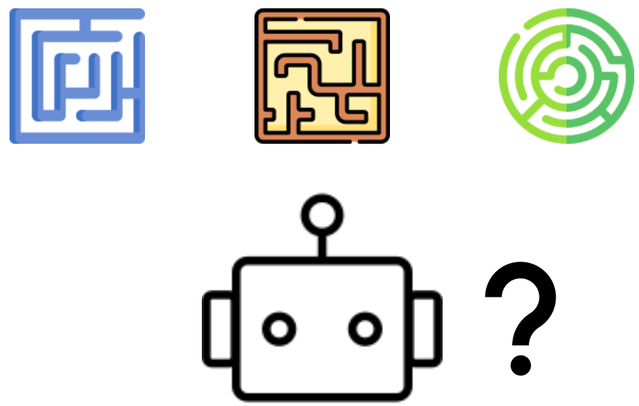
Policy Planning (PP): Known env., find the best policy



Definition 3.4 (Policy Planning Oracle)**.** A policy planning oracle $\mathsf{PP}(\varepsilon)$ takes as input a model θ and reward function r, and returns an action-value function $\{\hat{Q}_h\}_{h=1}^{H}$ satisfying, for a specified distribution ρ,

$$\mathbb{E}_{(s,a)\sim\rho}\left[\left|Q^{\star}_{\theta,r,h}(s,a)-\hat{Q}_h(s,a)\right|\right]\leq\varepsilon \quad \text{for all } h.$$

Constrained Planning (CP): Known env set., find the best policy of the best



Definition 3.5 (Constrained Planning Oracle)**.** A constrained planning oracle $\mathsf{CP}(\varepsilon)$ is given a reward function r and a constraint set over models $\Theta_c := \{\theta : L(\theta) \leq c\}$, where $L(\theta)$ is a supervised learning loss. It returns an action-value function $\hat{Q}$ such that, for a specified distribution ρ,

$$\mathbb{E}_{(s,a)\sim\rho}\left[\left|\hat{Q}_h(s,a)-\sup_{\theta\in\Theta_c} Q^{\star}_{\theta,h}(s,a)\right|\right]\leq\varepsilon \quad \text{for all } h.$$

Which one is (oracle) efficient?

PE < PP < CP

- $PE(\varepsilon) = SL(\varepsilon^2)$
- $PP(\varepsilon) = SL^H(\varepsilon^2 A^{-2H})$
- $CP(\varepsilon) = SL^{2^d}(\varepsilon^2 A^{-2H})$

Definition 3.3 (Policy Evaluation Oracle). A policy evaluation oracle $\mathrm{PE}(\varepsilon)$ takes as input a model θ, a policy π, and a reward function r, and returns an action-value function $\{\hat{Q}_h\}_{h=1}^H$ such that, for a specified distribution ρ,

$$\mathbb{E}_{(s,a)\sim\rho}\left[\left|Q^{\pi}_{\theta,r,h}(s,a) - \hat{Q}_h(s,a)\right|\right] \leq \varepsilon \quad \text{for all } h.$$

$SL^n(e)$: need to run supervised learning oracle n times with loss decreasing to the minimum plus e

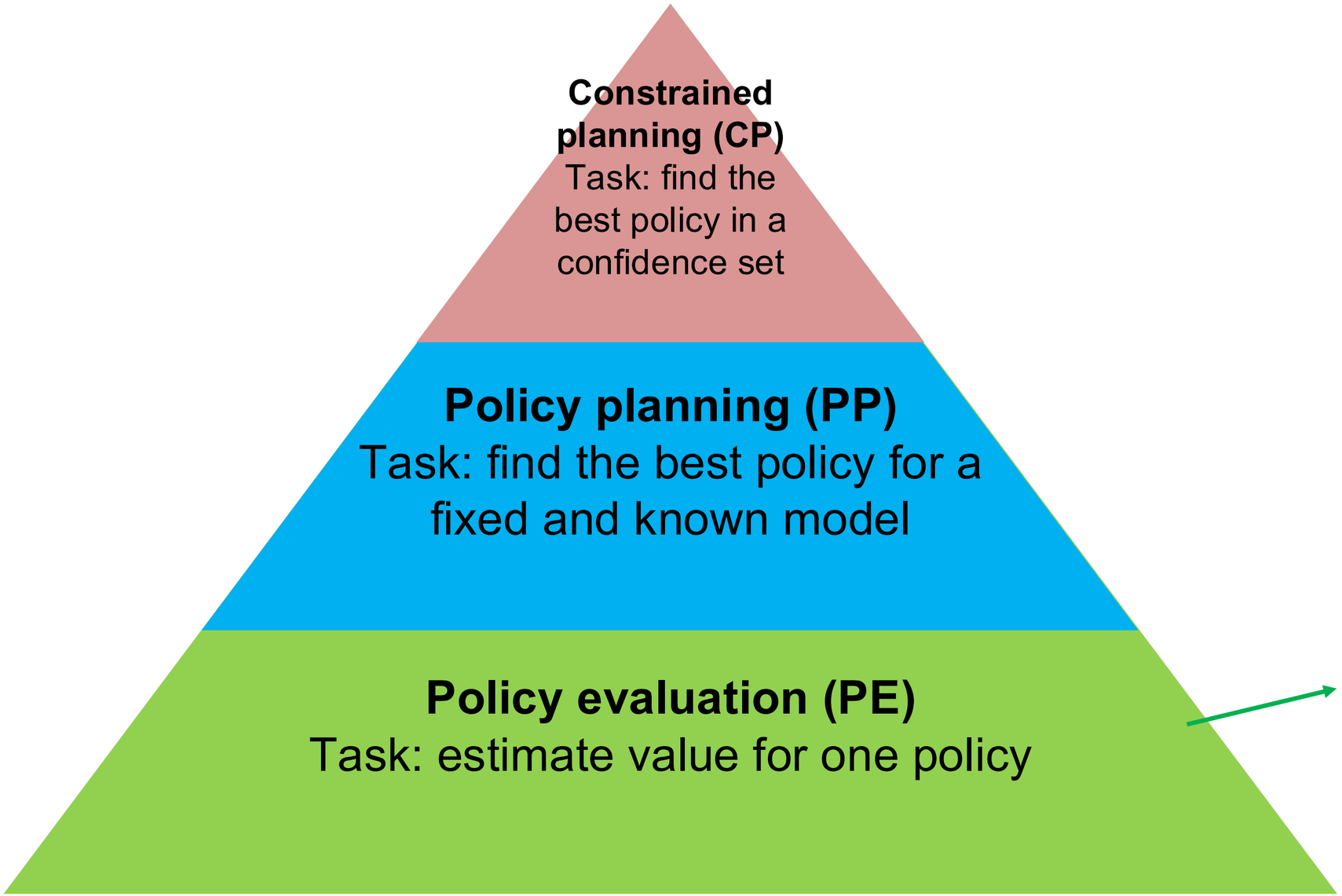
Q-value function:

$$Q^{\pi}_{r,h}(s,a) = r_h(s,a) + \mathbb{E}[Q^{\pi}_{r,h+1}(s',a')]$$

Optimal Q-value function:

$$Q^{*}_{r,h}(s,a) = r_h(s,a) + \mathbb{E}[\max_{a'} Q^{*}_{r,h+1}(s',a')]$$

Best-known SL-oracle Complexity



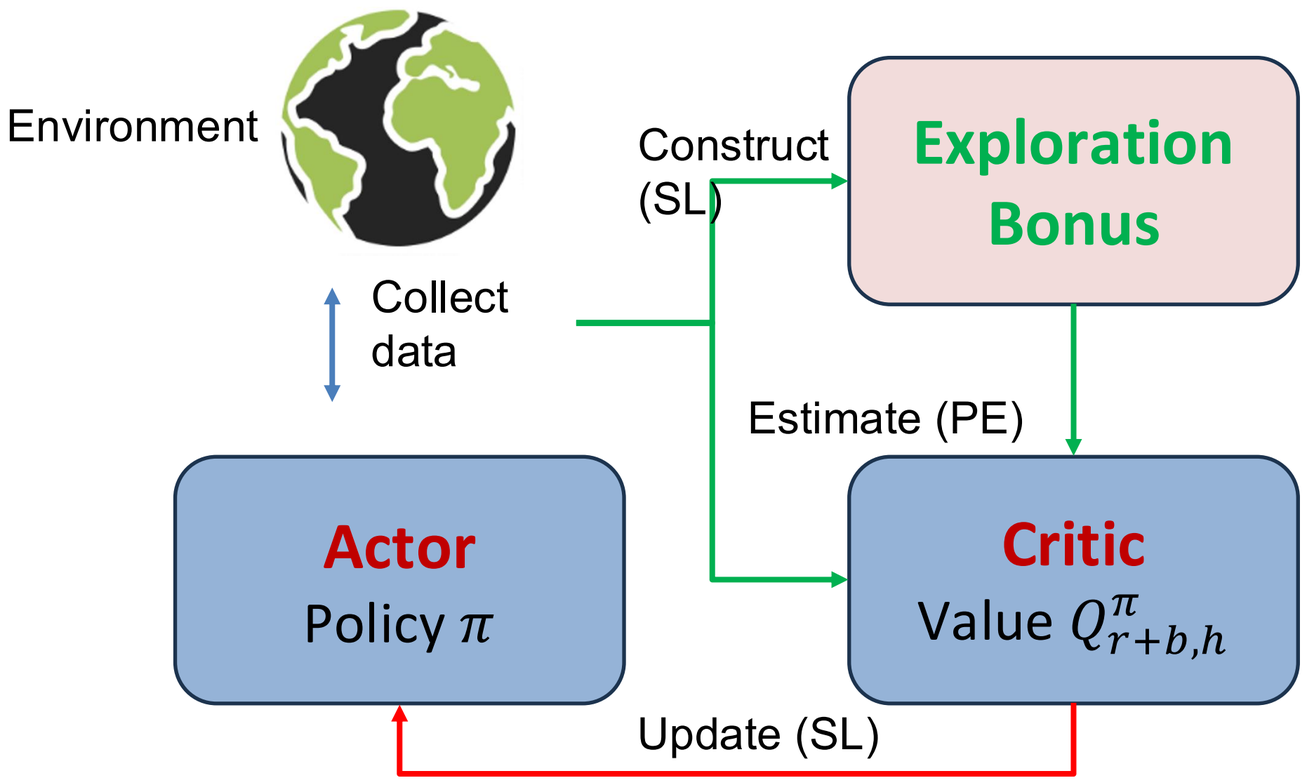
Reduce to supervised learning: solve using gradient descent

Our algorithm: Reducing RL to Supervised Learning

Supervised Learning Oracle (SL):

- Input: Dataset of (x, y) pairs.
- Task: Return function f minimizing loss $l(f(x), y)$
- Insight: This is exactly what modern Deep Learning's efficiently. If we reduce RL to this, we win.

Optimistic Actor-critic



For each iteration k:

1. Collect Data
2. Estimate Model (for representation)
3. Construct Exploration Bonus
4. Update Critic (PE)
5. Update Policy

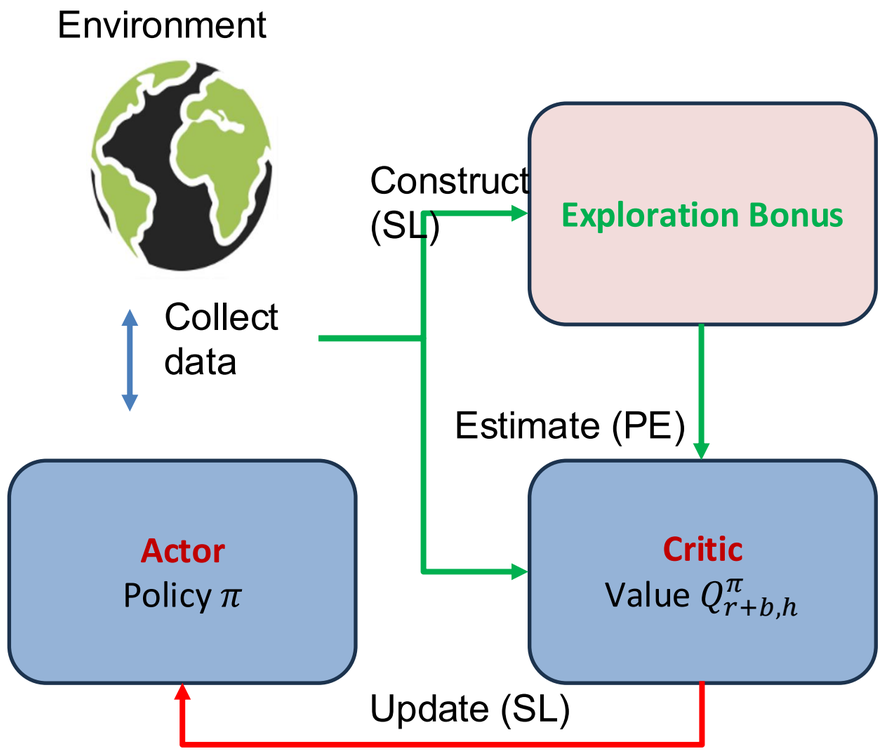
Collect Data and Obtain Representation

1. Collect data: Run current policy to step h then run uniform selection over actions.

2. Use existing data to learn the transition (SL)

$$\hat{\theta} = argmin_{\theta}\, L(\theta) = -\sum_{(s,a,s',h)\in D} \log P_{\theta,h}(s'|s,a)$$

$$P_{\theta,h}(s'|s,a) = \phi_{\theta,h}(s,a)^{\top}\mu_{\theta,h}(s')$$



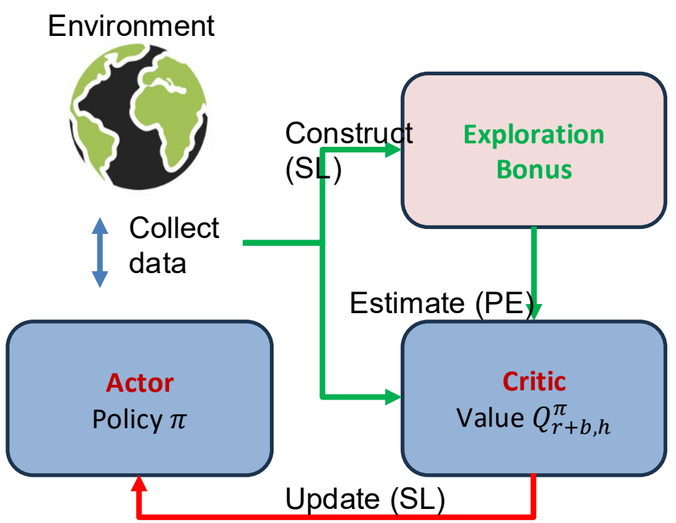
Exploration Bonus and Estimate Critic

1. Exploration bonus: $\tilde{b}_h^{(k)}(s,a) = \min\left\{\alpha\|\phi_{\hat{\theta}_k,h}(s,a)\|_{\hat{\Lambda}_{k,h}^{-1}},\ 1\right\}.$

$$\hat{\Lambda}_{k,h} = \lambda I + \sum_{n<k} \hat{\phi}_h^k(s_h^{n,h+1}, a_h^{n,h+1})\hat{\phi}_h^k(s_h^{n,h+1}, a_h^{n,h+1})^\top,$$

2. Estimate Critic $\hat{Q}_{k,h}(s,a)$: $\mathbb{E}_{(s,a)\sim\rho}\left[\left|\hat{Q}_{k,h}(s,a) - Q^{\pi^{(k)}}_{\hat{\theta}_k, r+\hat{b}^{(k)},h}(s,a)\right|\right] \le \frac{1}{\sqrt{K}}.$

- Sampling oracle: ρ
- Maximum iteration number: K
- Current iteration: k
- Current estimated model parameter: $\hat{\theta}_k$

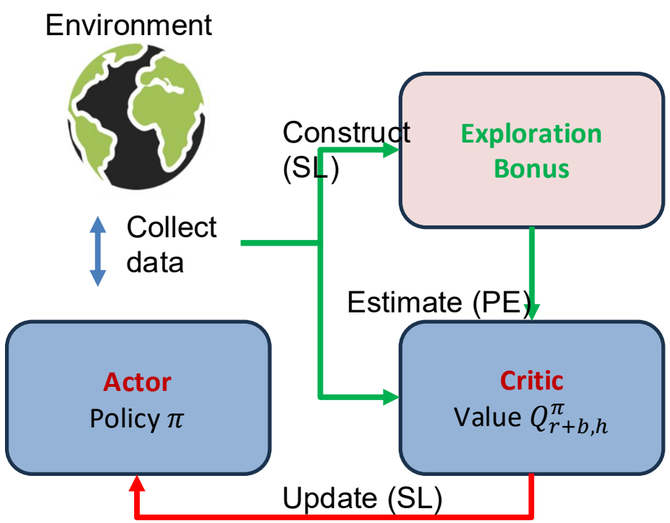


Update Policy

1. Mirror Descent:

$$\pi_h^{(k+1)} = \underset{\pi}{argmax} \ \langle \pi_h(\cdot\,|s), \hat{Q}_{k,h}(s,\cdot)\rangle - \frac{1}{\eta} KL\left(\pi_h(\cdot\,|s)||\pi_h^{(k)}(\cdot\,|s)\right)$$

$$\pi_h^{(k+1)}(a \mid s) \propto \pi_h^{(k)}(a \mid s)\exp\left(\eta\,\hat{Q}_{k,h}(s,a)\right).$$



Algorithm 1 Optimistic Actor–Critic (Opt-AC)

1: Initialize policy $\pi^{(0)}$ to be uniform over $\mathcal{A}$. Set hyper-parameters: maximum iteration number K, error rate ϵ, coefficients β, α, λ.
2: **for** $k = 0, 1, \ldots, K$ **do**
3: Collect H exploratory trajectories using $\pi^{(k)}$.
4: Learn model $\hat{\theta}_k$ via approximate MLE. (SL)
5: Construct exploration bonus $\hat{b}^{(k)}$ via (2).
6: Estimate optimistic critic $Q^{\pi^{(k)}}_{\hat{\theta}_k, r+\hat{b}^{(k)},h}$ by (3). (PE)
7: Update policy $\pi^{(k+1)}$ by (5).
8: **end for**
9: Return policy $\bar{\pi} = \text{Uniform}\{\pi_0, \ldots, \pi_K\}$

Theoretical Results

Number of Trajectories need to find an ϵ-optimal policy

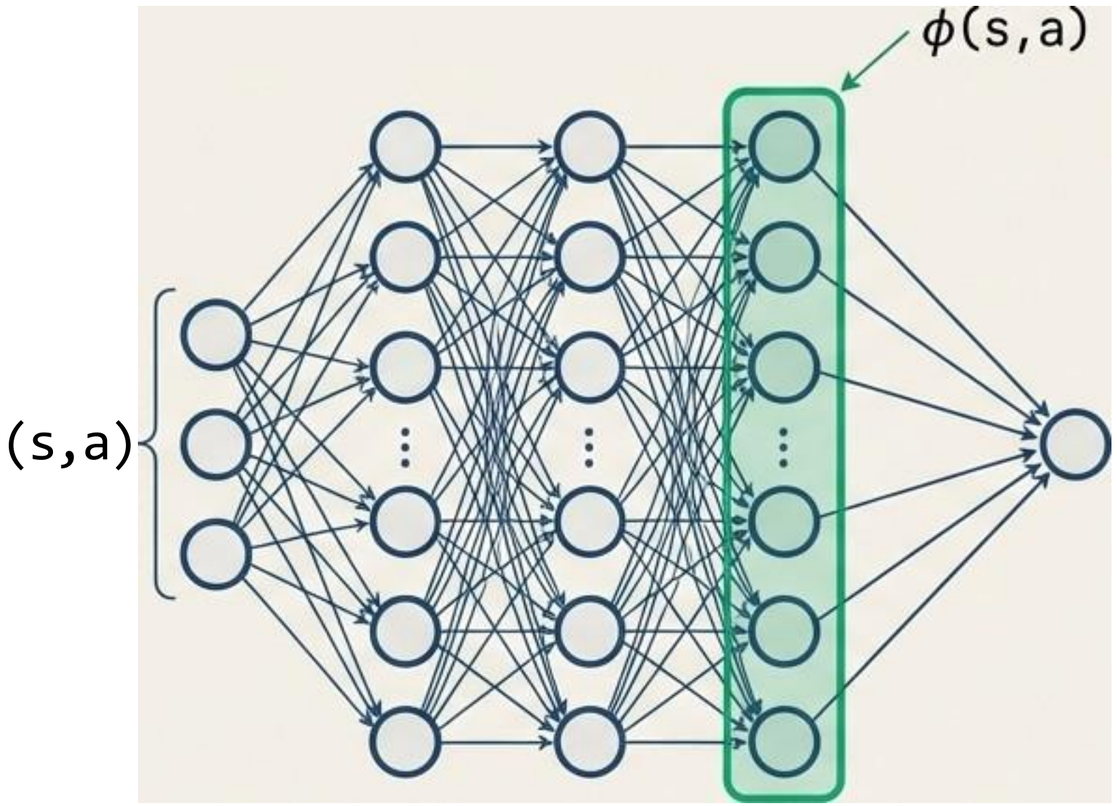
$$\tilde{O}\left(\frac{H^5 d^3 A^2 \log|\Theta|}{\epsilon^2}\right)$$

Theorem 5.2 (Sample Complexity). *Assume $\mathcal{M}$ is a low-rank MDP with dimension d, and Assumption 3.1 holds. Given any $\epsilon, \delta \in (0,1)$, let $\bar{\pi}$ be the output of Opt-AC and $\pi^\star$ be the optimal policy under the true model $P^\star$. Set $\beta = O(\log(K|\Theta|/\delta))$, $\alpha = \tilde{O}(\sqrt{|\mathcal{A}|}), \eta = O(\frac{1}{H\sqrt{K}})$ and $\lambda = \tilde{O}(1/d)$. Then, with probability at least $1-\delta$, we have $V^{\pi^\star}_{P^\star,r} - V^{\bar{\pi}}_{P^\star,r} \leq \epsilon$, and the total number of trajectories collected by Opt-AC is upper bounded by $\tilde{O}(\frac{H^5 d^3 |\mathcal{A}|^2 \log|\Theta|}{\epsilon^2})$.*

Name	Oracle Types	Convexity	Smoothness	Computation	Sample Complexity
FLAMBE (Agarwal et al., 2020)	PP(ε)	✗	✗	SL$^H(\varepsilon^2\|\mathcal{A}\|^{-2H})$	$\tilde{O}\left(\frac{H^{22} d^7 \|\mathcal{A}\|^9 \log\|\Theta\|}{\epsilon^{10}}\right)$
Rep-UCB (Uehara et al., 2021)	PP(ε)	✗	✗	SL$^H(\varepsilon^2\|\mathcal{A}\|^{-2H})$	$\tilde{O}\left(\frac{H^7 d^4 \|\mathcal{A}\|^2 \log\|\Theta\|}{\epsilon^2}\right)$
RAFFLE (Cheng et al., 2023)	PP(ε)	✗	✗	SL$^H(\varepsilon^2\|\mathcal{A}\|^{-2H})$	$\tilde{O}\left(\frac{H^5 d^4 \|\mathcal{A}\|^2 \log\|\Theta\|}{\epsilon^2}\right)$
SpanRL (Mhammedi et al., 2023)	CP(ε)	✗	✗	SL$^{2^d}(\varepsilon^2\|\mathcal{A}\|^{-2H})$.	$\tilde{O}\left(\frac{H^6 d^9 \|\mathcal{A}\|^4 \log\|\Phi\|}{\epsilon^2}\right)$
MOFFLE (Modi et al., 2021)	CP(ε)	✗	✗	SL$^{2^d}(\varepsilon^2\|\mathcal{A}\|^{-2H})$.	$\tilde{O}\left(\frac{H^7 d^{11} \|\mathcal{A}\|^{14} \log\|\Phi\|}{\epsilon^2}\right)$
Opt-AC (Ours)	PE(ε)	✓	✓	SL(ε^2)	$\tilde{O}\left(\frac{H^5 d^3 \|\mathcal{A}\|^2 \log\|\Theta\|}{\epsilon^2}\right)$
Lower Bound (Cheng et al., 2023)	-	-	-	-	$\Omega\left(\frac{HdA}{\epsilon^2}\right)$

From Theory to Practice: Feature Construction

Handling Unknown Features with Neural Networks



- Challenge: In Deep RL, we don't know the low-rank features Q(s, a).
- Solution: Conditional Random Fourier Features (cRFF).
- Implementation: We treat the penultimate layer of the Neural Network as the learned feature map Q.
- Theorem: Many real-world environments (smooth physical systems) can be approximated as Low-Rank MDPs using this technique.

Conditional Random Fourier Features

Motivation (RFF): $\mathrm{P}(y|x) \propto \exp\left(-\frac{||y-f(x)||^2}{2\sigma^2}\right) \approx \quad \langle \phi(x), \mu(y) \rangle$

$$[\phi(x)]_i = \frac{\cos\left(w_i^\top f(x) + b_i\right)}{\sqrt{D}}, \qquad [\mu(y)]_i = \frac{\cos\left(w_i^\top y + b_i\right)}{\sqrt{D}}$$

1. Sample $w_i \sim_{iid} N(0, \sigma^{-2} I)$
2. Sample $b_i \sim_{iid} Uniform[0, \pi]$

Theoretical Guarantee:

Under certain regularization conditions of the transitions, we have

Then, for any $\zeta > 0$, there exists a randomized algorithm cRFF (Algorithm 2) that constructs $2d(\zeta)$-dimensional feature maps $\hat{\phi}_\theta(s,a), \mu(s') \in \mathbb{R}^{2d}$ using $N(\zeta)$ samples such that, with probability at least $1-\delta$, uniformly over all $(\theta, s, a, s', h) \in \Theta \times \mathcal{S}^2 \times \mathcal{A} \times [H]$,

$$\left|\mathcal{T}_{\theta,h}(s' \mid s,a) - \hat{\phi}_\theta(s,a)^\top \mu(s')\right| \le \zeta.$$

Here $d(\zeta) = \tilde{O}(\zeta^{-2})$, and $N(\zeta) = \tilde{O}(\zeta^{-2})$.

Algorithm 2 Conditional Random Fourier Features (cRFF)

Input: $B_W = \{w \in \mathbb{R}^D : \|w\| \le W\}$.
Draw d i.i.d. frequencies $w_1, \ldots, w_d$ uniformly from B_W. Denote the volume of B_W by $\mathrm{Vol}(B_W)$.
Define the feature map $\mu : \mathcal{S} \to \mathbb{R}^{2d}$, where $[\mu(y)]_{2k} = \cos(2\pi w_k^\top s), [\mu(y)]_{2k+1} = -\sin(2\pi w_k^\top s), \forall k \in [d]$
Draw N samples $y^{(1)}, \ldots, y^{(N)} \sim \mathcal{T}_{\theta,h}(\cdot \mid s,a)$. For each $k \in [d]$, compute the empirical Fourier coefficient $\hat{g}_k = \frac{1}{N}\sum_{n=1}^{N} e^{-2\pi i w_k^\top y^{(n)}}$, and let $\hat{g}_k^{\mathrm{r}}$, $\hat{g}_k^{\mathrm{i}}$ be the real and imaginary part of $\hat{g}_k$, respectively.
Return $\hat{\phi}_\theta(s,a) \in \mathbb{R}^{2d}$ as

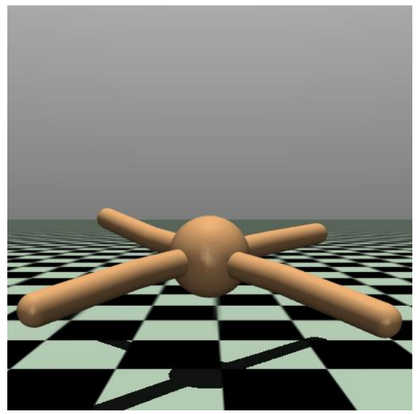
$$\hat{\phi}_\theta(s,a) = \left(\hat{g}_1^{\mathrm{r}}, \hat{g}_1^{\mathrm{i}}, \ldots, \hat{g}_d^{\mathrm{r}}, \hat{g}_d^{\mathrm{i}}\right) \times \mathrm{Vol}(B_W)/\sqrt{d}.$$

Experimental Setup

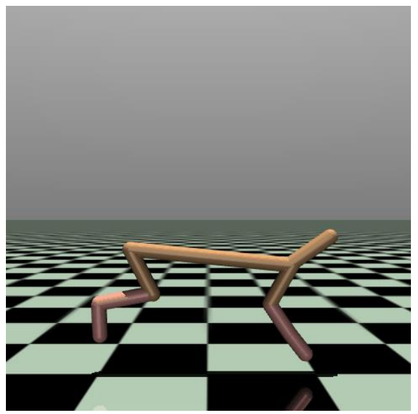
Algorithms:

- cRFFSAC: without the exploration bonus
- cRFFSAC + bonus: Full implementation
- SAC: soft actor-critic algorithm,
  - a strong baseline for continuous control.

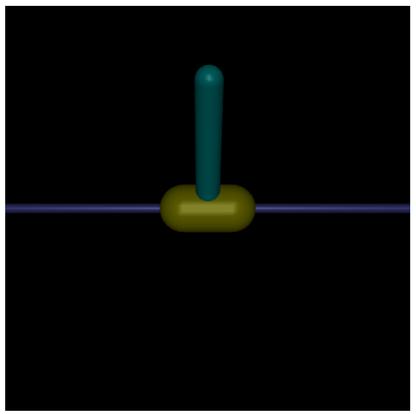
Ant

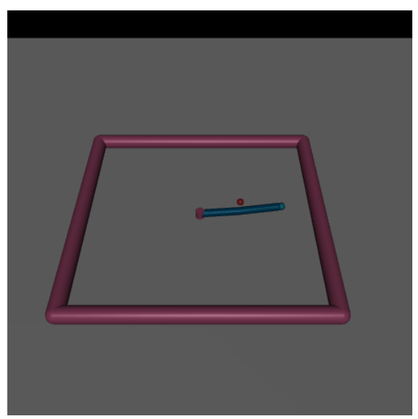


Half-cheetah

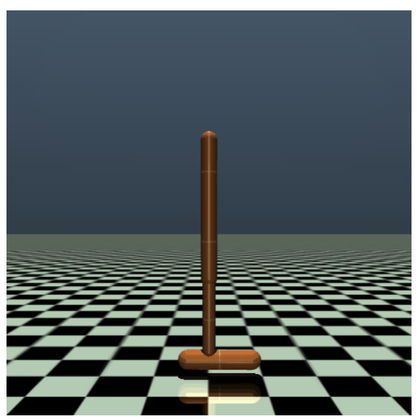


Inverted-pendulum





Reacher

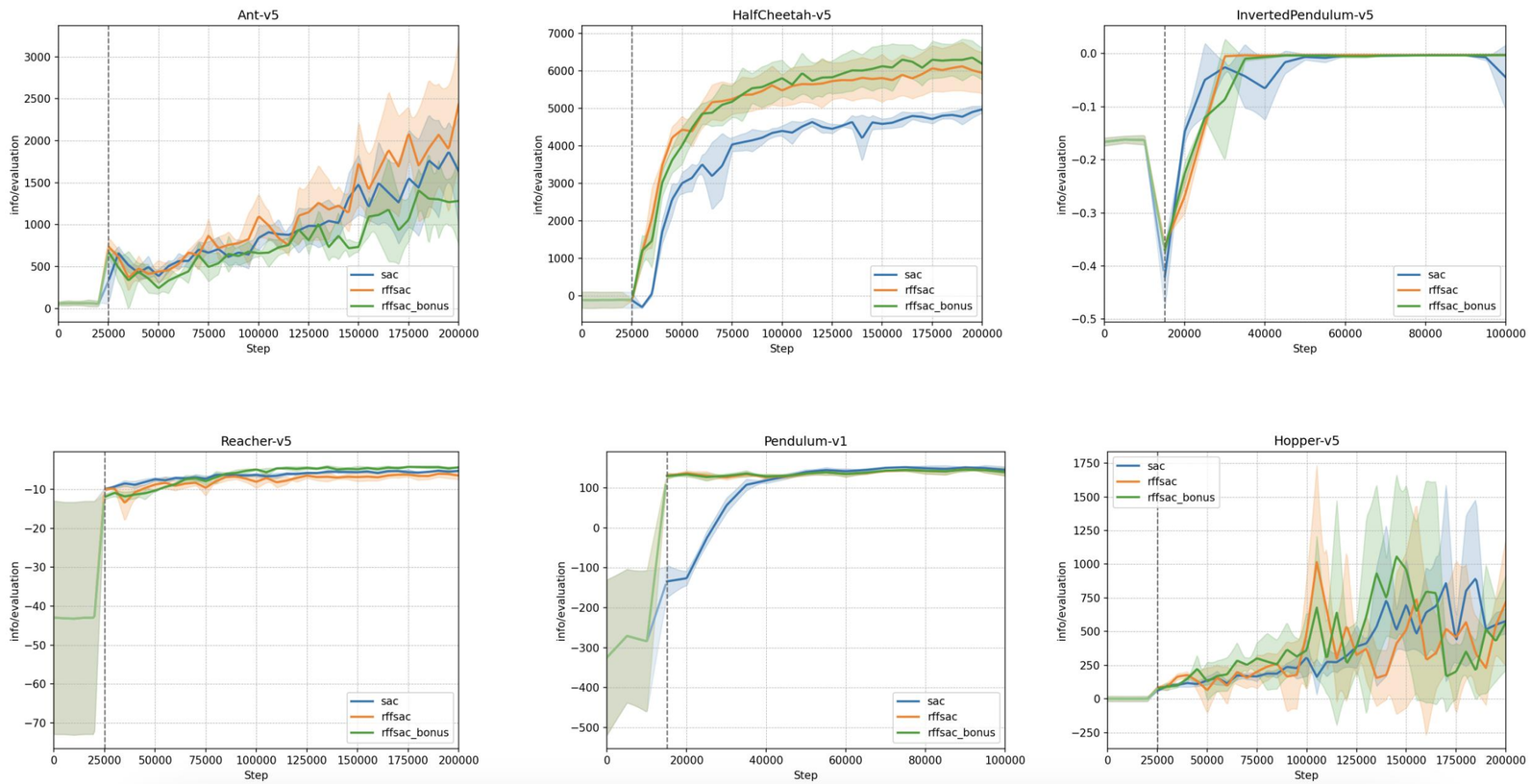


Hopper

Pendulum

Experimental Results

Algorithm	Ant	HalfCheetah	InvertedPendulum	Reacher	Pendulum	Hopper
SAC	2081±465	4994±99	**-0.0028±00.0**	-4.81±0.19	**156±3**	1316 ± 740
cRFFSAC	**2701±663**	6152±657	**-0.0028±00.0**	-5.75±0.85	153±4	**1449 ± 598**
cRFFSAC + Bonus	1910±422	**6454±479**	-0.0029±00.0	**-3.86±0.18**	153±3	1431 ± 538



Blue: SAC baseline

Orange: w/o bonus

Green: w. bonus

Summary

1. Computation primitive: Supervised learning
2. RL can be reduced to supervised learning with new algorithm
3. Favorable sample complexity (provably) can be achieved
4. From theoretical representation learning to practical algorithm