HyperAgent: A Simple, Efficient, Scalable and Provable RL framework for Complex Environment

Yingru Li

Academic webpage: `https://richardli.xyz/`
The Chinese University of Hong Kong, Shenzhen, China

Informs Optimization Society Conference, March 23, 2024

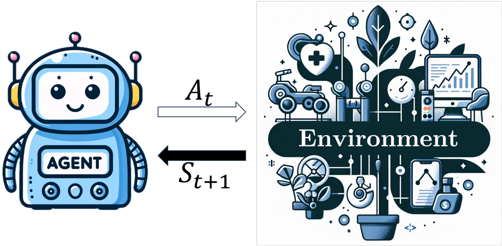
Jiawei Xu

Zhi-Quan Luo

arXiv:2402.10228

Outline

Reinforcement Learning Problem



Agent-Environment Interface. Experience: $A_0, S_1, A_1, S_2, \dots, A_t, S_{t+1}, \dots$

Environment $M = (\mathcal{S}, \mathcal{A}, P)$

- State $S_{t+1} \sim P(\cdot \mid S_t, A_t)$ for $t = 0, 1, \dots$.

Agent$(\mathcal{S}, \mathcal{A}, r, \mathcal{D}_t) \to \pi_t$ max long-term rewards

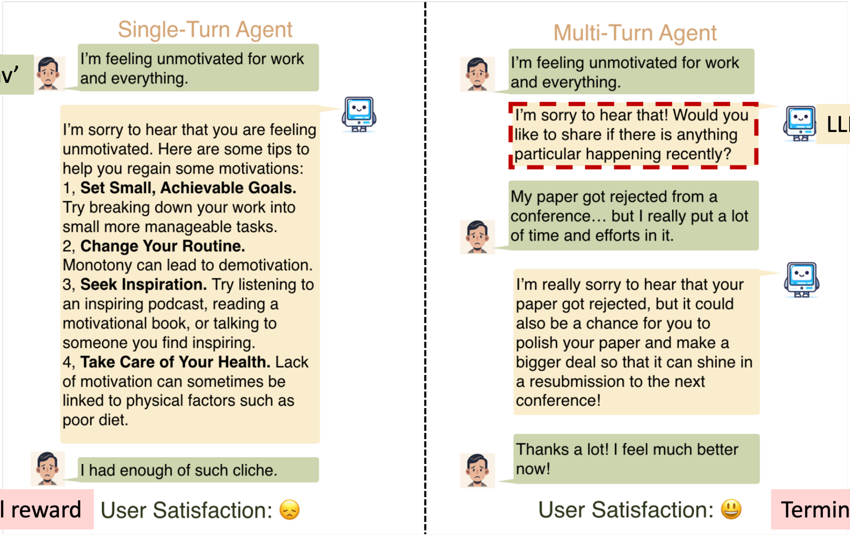
- **Reward** $R_{t+1} = r(S_t, A_t, S_{t+1})$ preference
- **Data** $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{A_{t-1}, S_t\}$ **accumulated**.
- **Policy** $\pi_t = \text{Agent}(\mathcal{S}, \mathcal{A}, r, \mathcal{D}_t)$.
- **Action** $A_t \sim \pi_t(\cdot \mid S_t)$;
- **Objective** $\pi_{\text{agent}} = (\pi_0, \pi_1, \dots)$ to maximize

$$\boxed{\mathbb{E}\Big[\sum_{t=0}^{T-1} R_{t+1} \mid \pi_{\text{agent}}, M\Big]}\,. \tag{1}$$

Motivating example: “multi-turn” LLM agent

User: Env’

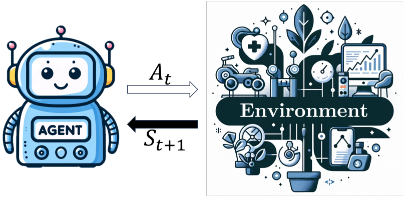
LLM: Agent



Terminal reward

Terminal reward

Challenges in Real-world RL Agent



Agent-Environment Interface. Experience: $A_0, S_1, A_1, S_2, \ldots, A_t, S_{t+1}, \ldots$

Complex Environment:

- **Large state space**: (images, videos, audio, text, high-dimensional feature vectors, etc.) $|\mathcal{S}| \approx 10^{100}$
- **Accumulated data** $\mathcal{D} \uparrow$ as interacting with the environment.

Resource Constraints for Agent:

- Computation & memory (bounded per-step complexity)
- Experimental budgets (limited data collection, human feedback, etc.)

Outline

RL in Complex Environment

- Scaling up! Then?

Introducing HyperAgent: Simple, Efficient, Scalable

- Insights and theoretical analysis

Development of RL Algorithms: A history of "Scale up!"

50s-60s	70s-90s	2010s --
R. Bellman, R. A. Howard, D. Blackwell **Dynamic Programming** **(Markov) Decision Processes**	R.S. Sutton & A. Barto C. J.C.H. Watkins D. Bertsekas & J. Tsitsiklis **Temporal Difference, Q-Learning** **Approximate (Neuro) DPs** **Parallel and Distributed Computation**	DeepMind OpenAI Etc. **Deep RL (Advances in Algorithms and Computations)** **AlphaGo & GPTs (Foundation Models)**

Scale up!

Computation First

- Problem **Scale up**$\uparrow$: (S1) Larger$\uparrow$ state space $\mathcal{S}$; (S2) Data $\mathcal{D}$ accumulated$\uparrow$.
- **Modern RL Paradigm**: (S1) **Function Approximation** (Deep Neural Networks); (S2) **Incremental update** with SGD, Experience Replay and/or Target Network.

(K1) **Bounded Per-step Complexity**: **'NOT Scale'** polynomially with $|\mathcal{S}|$ and $|\mathcal{D}|$.

Scalablity $\neq$ Efficiency: RLHF for LLMs

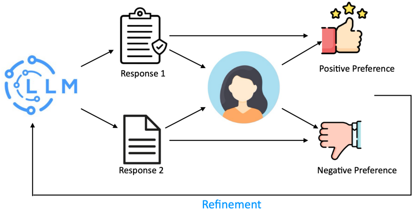


Table 4: E2E time breakdown for training a 13 billion parameter ChatGPT model via DeepSpeed-Chat on a single DGX node with 8 NVIDIA A100-40G GPUs.

Model Sizes	Step 1	Step 2	Step 3	Total
Actor: OPT-13B, Reward: OPT-350M	2.5hr	0.25hr	10.8hr	13.6hr

Table 5: E2E time breakdown for training a 66 billion parameter ChatGPT model via DeepSpeed-Chat on 8 DGX nodes with 8 NVIDIA A100-80G GPUs/node.

Model Sizes	Step 1	Step 2	Step 3	Total
Actor: OPT-66B, Reward: OPT-350M	82 mins	5 mins	7.5hr	9hr

Bounded per-step complexity as Scale up $\uparrow$

- $\mathcal{S}\uparrow$: more complex or longer conversations
- $\mathcal{D}\uparrow$: adapt to extensive human feedbacks

Inefficiency $\downarrow$

- **Data Hungary**: 1.5M (Offline) and 1.7M (Online) in LLaMA2 [TMS+23] **Human feedback**
  - **scarce & expensive**
- **Computation Costs**: RLHF occupies most of the training time. [YAR+23]

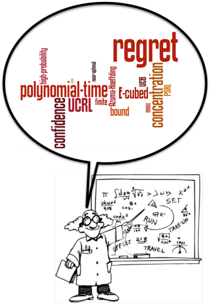
Practical advancements for "efficient" Deep RL

Algorithm		Components
DDQN	(16)	Incremental SGD with experience replay (finite buffer) and target network
Rainbow	(18)	(**DDQN**) + Prioritized replay, Dueling networks, Distributional RL, Noisy Nets.
BBF	(23)	(**DDQN**) + Prioritized replay, Dueling networks, Distributional RL, Self-Prediction, Harder resets, Larger network, Annealing hyper-parameters.

Table: The extra components used in various algorithms, e.g. DDQN [VHGS16], Rainbow [HMVH+18], BBF [SCC+23].

- ✓ **Scalable**: e.g. DDQN use incremental SGD with experience replay and target network.
- ✗ **Deployment inefficient**: Complicated component and many heuristic tricks. **Hard to tune**.
- ✗ **Provably inefficient**: e.g. BBF use ϵ-greedy **exploration strategy** which need exponential many sample in some environment, provably [Kak03, Str07, OVRRW19, DMM+22].

Principled approaches for data efficieny



R.A. Fisher, William R. Thompson (30s)
T. L. Lai and H. Robbins (80s)
Sequential design and allocations (MAB)

Before 90s

E3/PSRL/UCRL ...
Tabular RL

2000 --

Eluder dimension/Information Ratio
Bellman Rank/Bilinear Structure
Decoupling coefficient/DEC/GEC ...
Structural assumption and Algorithms for RL with function approximation

2014 --

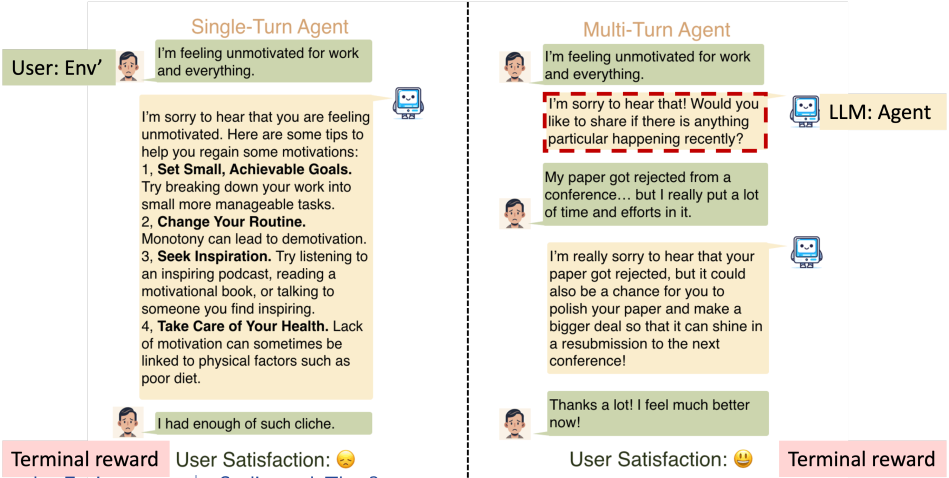
Data First

Sequential decision making under **uncertainty** with **sublinear regret**.

Deep exploration in "multi-turn" LLM agent

Deep exploration

The decision may position the agent to more effectively **acquire information over subsequent** time steps.

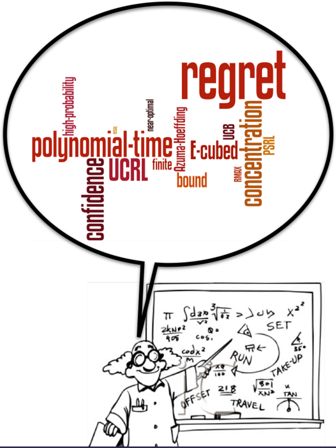


Data Efficient Exploration under Function Approximation (FA)

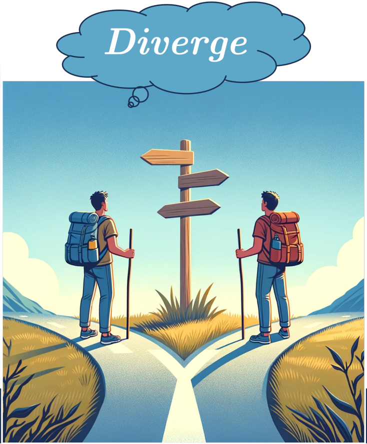
- **Posterior sampling**: data-efficient exploration strategy
  Require **conjugacy** for posterior update
  Feasible **only** in tabular MDP with dirichlet prior.

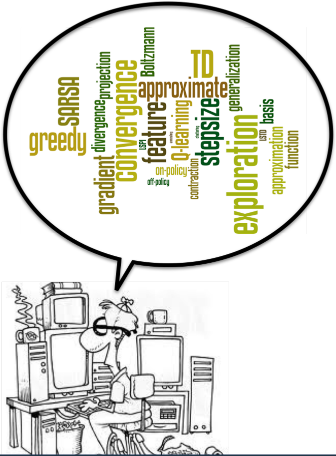
- Extending **posterior sampling** to general FA:
  ✗ **Intractable computation**: sample from intricate distribution [Zha22, DMZZ21, ZXZ$^+$22].
  ✗ **Unbounded memory and computation**:
  (1) Store entire history and retrain for each episode, e.g. RLSVI [OVRRW19], LSVI-PHE [ICN$^+$21].
  (2) Computation cost scale poly w. # episodes, say LMC-LSVI [ILX$^+$24]

- Same challenges for **OFU**-based algorithms under general FA.



Theory! Data First





Scale up! Computation First

Research question?

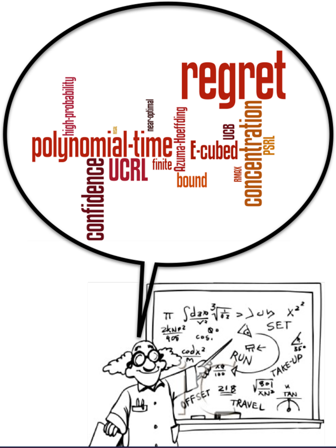
Towards fulfilling the promise of RL in real-world complex environment, can we design

(A1) **Simple** Algorithm — easy to use and deploy (E3)

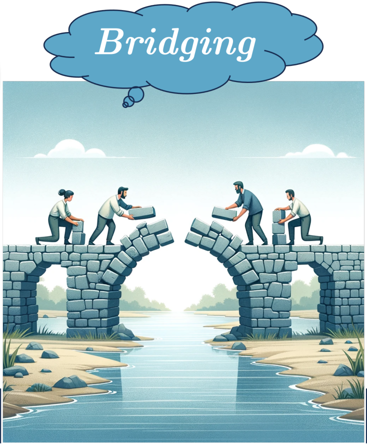
(A2) **Efficient** Algorithm — low data (E1) and computation cost (E2)

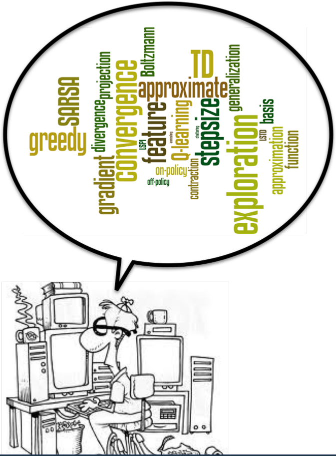
(A3) **Scalable** Algorithm — large $\mathcal{S} \uparrow$ (S1) and accumulated $\mathcal{D} \uparrow$ (S2)

"To complicate is easy. To simplify is difficult." – Bruno Munari



Theory! Data First





Scale up! Computation First

Outline

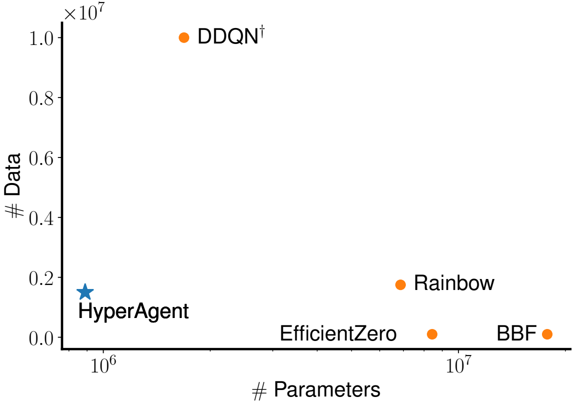
HyperAgent: Simple and Scalable Algorithmic Component

Algorithm	Components
DDQN (16)	Incremental SGD with experience replay (finite buffer) and target network
Rainbow (18)	(**DDQN**) + Prioritized replay, Dueling networks, Distributional RL, Noisy Nets.
BBF (23)	(**DDQN**) + Prioritized replay, Dueling networks, Distributional RL, Self-Prediction, Harder resets, Larger network, Annealing hyper-parameters.
HyperAgent	**Hypermodel**

Table: The extra techniques used in different algorithms, e.g. DDQN [VHGS16], Rainbow [HMVH+18], BBF [SCC+23] and **our HyperAgent**.

- ✓ **Simple**: **Only one** additional component, **hypermodel**, compatiable with all feedforward DNN.
- ✓ **Scalable**: **Incremental** SGD under DNN function approximation, same as DDQN.
- ✓ **Efficient**: **Incremental approximation of posteriors** over general value function **without conjugacy**
- $\Rightarrow$ **data-efficient** exploration via approximate posterior sampling w. **bounded per-step computation**.

HyperAgent in Atari suite: Human-level performance (1 IQM)



- ✓ **Data efficient**: 15% data consumption of DQN[VHGS16] by **Deepmind**. (1.5**M interactions**)
- ✓ **Computation efficient**: 5% model parameters of BBF[SCC+23] by **Deepmind**.
- **Ensemble+** [OAC18, OVRRW19] achieves a **mere** 0.22 **IQM score under** 1.5**M** interactions but necessitates **double the parameters of** `HyperAgent`.

HyperAgent: Introducing Hypermodel

paramatric function   reference distribution

- **Hypermodel**: in general $(\boxed{f_\theta}, \boxed{P_\xi})$ s.t.

$$f_\theta(x, \boxed{\xi}) \text{ is an approximate posterior predictive sample on data } x.$$

$\xi \sim P_\xi$

Special case: predictive sampling from Linear-Gaussian model

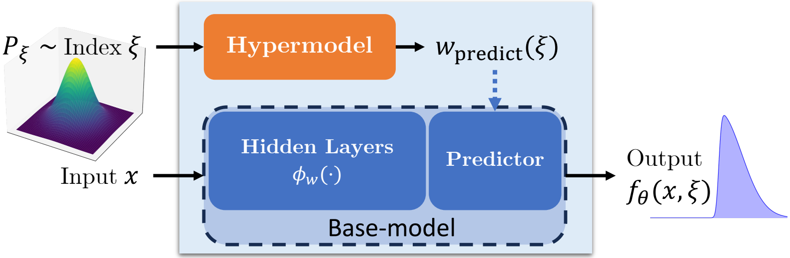
Suppose $\theta^* \sim N(\mu, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma}$ represent the model uncertainty.

Box-Muller transform: $P_\xi = N(0, I_M)$, $\theta = (\boldsymbol{A} \in \mathbb{R}^{d\times M}, \mu \in \mathbb{R}^d)$ s.t. $\boldsymbol{A}\boldsymbol{A}^\top = \boldsymbol{\Sigma}$.

$$f_\theta(x, \xi) := \langle x, \mu + \boldsymbol{A}\xi \rangle \sim N(x^\top \mu, x^\top \boldsymbol{A}\boldsymbol{A}^\top x)$$

HyperAgent: Hypermodel for Feedforward Deep Networks

- **Base model**: DNN $\langle \phi_w(\cdot), w_{\text{predict}} \rangle$



- **Hypermodel**: Choose $f_\theta(x, \xi) = \langle \phi_w(x), w_{\text{predict}}(\xi) \rangle$ with $w_{\text{predict}}(\xi) = A\xi + b$ where $\xi \sim P_\xi$.

$$f_\theta(x, \xi) = \underbrace{\langle \phi_w(x), b \rangle}_{\textbf{'mean'}\ \mu_\theta(x)} + \underbrace{\langle \phi_w(x), A\xi \rangle}_{\textbf{'variance'}\ \sigma_\theta(x, \xi)}$$

The degree of uncertainty

HyperAgent: Hypermodel for Deep RL

- **Base model** for **DQN-type value function**

$$f_\theta(s,a) = \langle \boxed{\phi_w(s)}, \theta^{(a)} \rangle$$

with parameters $\theta = \{w, \boxed{(\theta^{(a)} \in \mathbb{R}^d) : a \in \mathcal{A}}\}$

Action-specific parameters for discrete action set $\mathcal{A}$

- **Hypermodel** for **randomized value function** depends on (s,a) and a **random** index $\xi \sim P_\xi$:

$$f_\theta(s,a,\boxed{\xi}) = \langle \boxed{\phi_w(s)}, \underbrace{A^a\xi + b^a}_{\theta^a(\xi)} \rangle$$

Random index $\xi \sim P_\xi$

with parameters $\theta = \{w, \boxed{(A^{(a)} \in \mathbb{R}^{d\times M}, b^{(a)}) : a \in \mathcal{A}}\}$.

Action-specific parameters

- **Tabular** representation: $\boxed{\phi_w(s)}$ is fixed one-hot vector in $\mathbb{R}^{|\mathcal{S}|}$ where $d = |S|$. **(Unification!)**

HyperAgent: Algorithmic Framework

Algorithm `HyperAgent` Framework

Input: Initial parameter θ_{init}, hypermodel f_θ with reference dist. P_ξ and perturbation dist. $P_{\mathbf{z}}$.
Init. $\theta = \theta^- = \theta_{\text{init}}$, train step $j = 0$ and buffer D
for each episode $k = 1, 2, \ldots$ **do**
  Sample index mapping $\xi_k \sim P_\xi$
  Set $t = 0$ and **Observe** $S_{k,0} \sim \rho$
  repeat
    Select $A_{k,t} = \arg\max_{a \in \mathcal{A}} f_\theta(S_{k,t}, a, \xi_k(S_{k,t}))$
    Observe $S_{k,t+1}$ **from environment** and $R_{k,t+1} = r(S_{k,t}, A_{k,t}, S_{k,t+1})$.
    Sample perturbation random vector $\mathbf{z}_{k,t+1} \sim P_{\mathbf{z}}$
    D.**add**$((S_{k,t}, A_{k,t}, R_{k,t+1}, S_{k,t+1}, \mathbf{z}_{k,t+1}))$
    Increment step counter $t \leftarrow t + 1$
    $\theta, \theta^-, j \leftarrow \texttt{update}(D, \theta, \theta^-, \xi^- = \xi_k, t, j)$
  until $S_{k,t} = s_{\text{terminal}}$
end for

HyperAgent: Objective for generic hypermodel (f_θ, P_ξ)

- For a transition tuple $d = (s, a, r, s', \overbrace{\mathbf{z}}^{\text{perturbation random vector}}) \in D$ and given index ξ, the temporal difference (TD) error:

$$\ell^{\gamma,\sigma}(\underbrace{\theta}_{\text{main parameters, optimization variables}}; \underbrace{\theta^-}_{\text{target parameters, fixed here and updated in an outer loop}}, \boldsymbol{\xi}^-, \xi, d) = \left(f_\theta(s, a, \xi) - (r + \underbrace{\sigma}_{\text{control the std of injected noise}} \xi^\top \mathbf{z} + \overbrace{\gamma}^{\text{discounted factor}} \max_{a' \in \mathcal{A}} f_{\theta^-}(s', a', \boldsymbol{\xi}^-(s'))) \right)^2 \quad (2)$$

- $\boldsymbol{\xi}^-$: the target index mapping s.t. $\boldsymbol{\xi}^-(s)$ one-to-one maps each state $s \in \mathcal{S}$ to a random vector from P_ξ, all of which are **independent** with ξ.

- Integrate ξ over Equation (2) yields objective $L^{\gamma,\sigma,\beta}$ where $\beta \geqslant 0$ is for the prior regularization

$$L^{\gamma,\sigma,\beta}(\theta;\theta^-,\boldsymbol{\xi}^-,D) = \mathbb{E}_{\xi\sim P_\xi}[\sum_{d\in D}\frac{1}{|D|}\ell^{\gamma,\sigma}(\theta;\theta^-,\boldsymbol{\xi}^-,\xi,d)] + \frac{\beta}{|D|}\|\theta\|^2 \tag{3}$$

- Optimize Equation (3) using **mini-batch SGD (in practice, default Adam)**:

$$\tilde{L}(\theta;\theta^-,\boldsymbol{\xi}^-,\boxed{\tilde{D}}) = \frac{1}{|\tilde{\Xi}|}\sum_{\xi\in\boxed{\tilde{\Xi}}}\left(\sum_{d\in\boxed{\tilde{D}}}\frac{1}{|\tilde{D}|}\ell^{\gamma,\sigma}(\theta;\theta^-,\boldsymbol{\xi}^-,\xi,d)\right) + \frac{\beta}{|D|}\|\theta\|^2 \tag{4}$$

a batch of data $\tilde{D}$ sampled from D

a batch of indices $\tilde{\Xi}$ sampled from P_ξ

- **Update** the main parameters θ in each step according to Equation (4), and updates the target parameters θ^- periodically with less frequency. $\Rightarrow$ **Bounded per-step computation**.

Simple illustration for Deep Exploration: DeepSea Environment

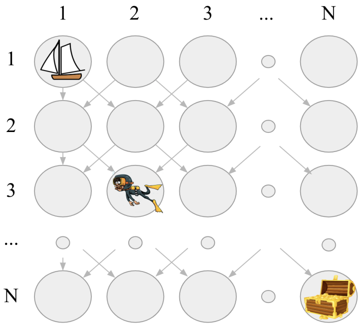
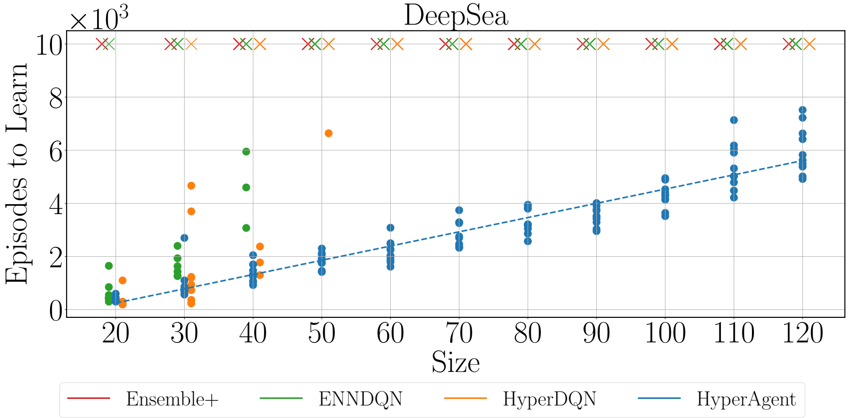


Figure: DeepSea: The agent receives a reward of 0 for $\swarrow$, and a penalty of $-(0.01/N)$ for $\searrow$, where N denotes the size of DeepSea. The agent will earn a reward of 1 upon reaching the lower-right corner but she do NOT know in advances whether there is a reward until reaching the goal.

exploration method	expected episodes to learn
optimal	$\Theta(N)$
pure exploitation	∞
dithering (ϵ-greedy)	$\Theta(2^N)$
optimistic	$\Theta(N)$
randomized	$\Theta(N)$

Expected number of episodes required to learn an optimal policy for DeepSea with size N.
Optimistic: optimism in the face of uncertainty (**OFU**);
Randomized: randomizing the belief of the environment, e.g. **Posterior sampling**

HyperAgent: Efficiency in benchmarks (DeepSea)



Comparison with Ensemble+ [OAC18, OVRRW19], HyperDQN [LLZ+22], ENN-DQN[OWA+23].

- ✓ Scalable as size $N \uparrow$. State representation: one-hot vector in high-dimension $\mathbb{R}^N$.
- ✓ Data efficient: `HyperAgent` the only and first achieving optimal episode complexity $\Theta(N)$.

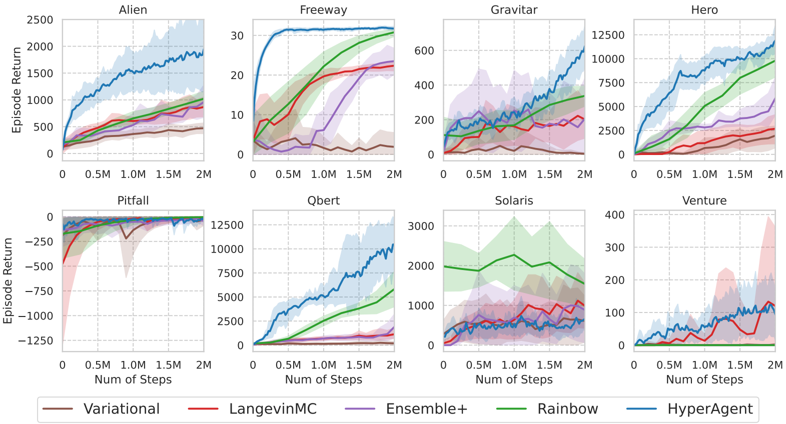


Figure: Comparison on approximate posterior sampling methods: variational approximation (SANE [AL21]), Langevin Monte-Carlo (AdamLMCDQN [ILX$^+$24]) and Ensemble+[OAC18, OVRRW19]

Outline

How does HyperAgent achieve efficient deep exploration?

- Tabular HyperAgent: $f_{\theta_k}(s,a,\xi) = \mu_{k,sa} + \tilde{m}_{k,sa}{}^\top \xi$
- **Incremental update** with computation complexity $O(M)$:

Perturbation random vector

$$\tilde{m}_{k,sa} = \frac{(N_{k-1,sa}+\beta)\,\tilde{m}_{k-1,sa} + \sum_{t\in E_{k-1,sa}} \sigma\, \mathbf{z}_{\ell,t+1}}{(N_{k,sa}+\beta)} \in \mathbb{R}^M \qquad (5)$$

Visitation counts of (s,a) up to episode k

Set of timesteps encoutering (s,a) in episode $k-1$

Lemma 1 (Sequential posterior approximation).

For $\tilde{m}_k$ recursively defined in Equation (5) with $\mathbf{z} \sim \mathcal{U}(\mathbb{S}^{M-1})$. For any $k \geqslant 1$, define the good event of ε-approximation

$$\mathcal{G}_{k,sa}(\varepsilon) := \left\{ \|\tilde{m}_{k,sa}\|^2 \in \left((1-\varepsilon)\frac{\sigma^2}{N_{k,sa}+\beta}, (1+\varepsilon)\frac{\sigma^2}{N_{k,sa}+\beta} \right) \right\}.$$

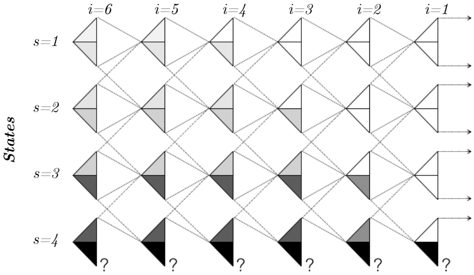
The joint event $\cap_{(s,a)\in\mathcal{S}\times\mathcal{A}} \cap_{k=1}^K \mathcal{G}_{k,sa}(\varepsilon)$ holds w.p. at least $1-\delta$ if $M \simeq \varepsilon^{-2}\log(|\mathcal{S}||\mathcal{A}|T/\delta)$.

How does HyperAgent achieve efficient deep exploration?

Stochastic Bellman backup for `HyperAgent`:

$$f_{\theta_k^{(i+1)},\xi_k} = F_k^{\gamma} f_{\theta_k^{(i)},\xi_k} \approx \Big(r_{sa} + \gamma \langle V_{f_{\theta_k^{(i)},\xi_k}}, \overbrace{\hat{P}_{k,sa}}^{\text{Empirical transition}} \rangle\Big) + \underbrace{\overbrace{\tilde{m}_{k,sa}^{\top} \xi_k(s)}^{\text{W. std} \propto \sqrt{\frac{1}{N_{k,sa}}}}}_{\text{"Randomized bonus"}}, \qquad (6)$$

where $f_{\theta,\xi^-}(s,a) = f_\theta(s,a,\xi^-(s))$ and $V_Q(s) := \max_a Q(s,a), \forall s$ is the greedy value w.r.t. Q.



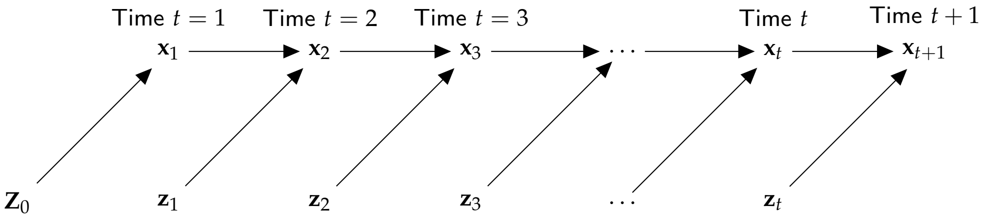
(1) $N_{k,(4,\searrow)} = 1$. Other (s,a) almost **infinite data**.
(2) **Propagation of uncertainty** from later time period to earlier time period.
Incentivize deep exploration.
(3) **Darker shade** indicates **higher** degree of uncertainty.

HyperAgent: Theoretical Guarantees in RL

	Practice in General FA			$(LB)\ \Omega(\sqrt{H^3SAK})$	Theory in Tabular
Alg↓ Metric→	**Tract'**	**Incre'**	**Effici'**	**Regret**	**Per-step Comp'**
PSRL	✗	✗	✗	$(B\Re)\ \tilde{O}(H^2\sqrt{SAK})$	$O(S^2A)$
Bayes-UCBVI	✗	✗	✗	$(F\Re)\ \tilde{O}(\sqrt{H^3SAK})$	$O(S^2A)$
RLSVI	✓	✗	✗	$(B\Re)\ \tilde{O}(H^2\sqrt{SAK})$	$O(S^2A)$
Ensemble+	✓	✓	●	N/A	N/A
LMC-LSVI	✓	✓	●	$(F\Re)\ \tilde{O}(H^2\sqrt{S^3A^3K})$	$\tilde{O}(KSA+S^2A)$
`HyperAgent`	✓	✓	✓	$(B\Re)\ \tilde{O}(H^2\sqrt{SAK})$	✓ $O(\log(K)SA+S^2A)$

- Finite-horizon tabular RL: # states: S, # actions: A, # horizons: $\tau = H$, # episodes: K
- **Per-step computation** $\mathrm{poly}(K)$ scaling is unacceptable under bounded computation. $K\uparrow \Leftrightarrow |\mathcal{D}|\uparrow$!
- **HyperAgent** is the **first** efficient and scalable RL agent, with practical efficiency as well as logarithmic per-step computation $\tilde{O}(\log K)$ & sublinear regret in tabular setting.

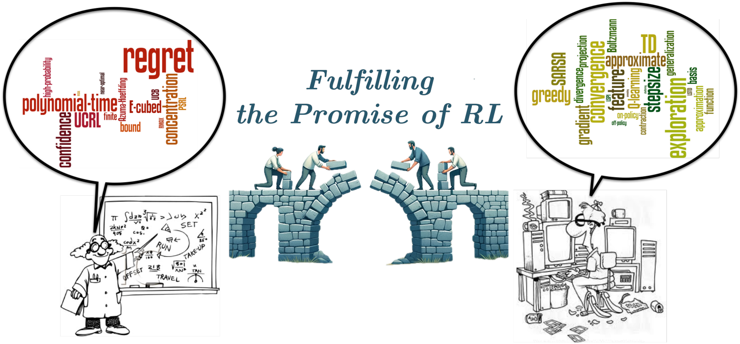
The novelty and difficulty in the mathemtical analysis: No Prior Art



Difficulty: Sequential dependence of high-dimensional R.V. due to the adaptive nature of Sequential Decision Making.

First probability tool for **sequential random projection**.

A **non-trivial** martingale extension of the Johnson–Lindenstrauss (JL) lemma and subspace embedding. [Li24a, Li24b, LXL24]



HyperAgent is the first principled RL agent that is

- **Simple, Efficient** and **Scalable**;
- **Empirically** and **Theoretically** justified.
- **No Prior Art**. Set up a **new benchmark**.

Promising future directions

Based on Li, Y., Xu, J., Han, L., & Luo, Z. Q. (2024). HyperAgent: A Simple, Scalable, Efficient and Provable Reinforcement Learning Framework for Complex Environments. arXiv preprint arXiv:2402.10228.

- Integrate **actor-critic** type of deep RL framework with **HyperAgent** for continuous control.
- Integrate **transformer-based model** with **HyperAgent** is also doable.
  - **LLM-based Agent!**
  - **Data-efficient** RLHF! (Efficient exploration for LLMs [DAHVR24])
- Extension to RL under **Linear Function Approximation** pose no much more difficulty.
- Extension to function approximation with **generalized linear model and neural tangent kernel with SGD update** is possible. Further bridging the gap!

References I

[AL21] Siddharth Aravindan and Wee Sun Lee. State-aware variational thompson sampling for deep q-networks. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 124–132, 2021.

[DAHVR24] Vikranth Dwaracherla, Seyed Mohammad Asghari, Botao Hao, and Benjamin Van Roy. Efficient exploration for llms. *arXiv preprint arXiv:2402.00396*, 2024.

[DMM$^+$22] Chris Dann, Yishay Mansour, Mehryar Mohri, Ayush Sekhari, and Karthik Sridharan. Guarantees for epsilon-greedy reinforcement learning with function approximation. In *International conference on machine learning*, pages 4666–4689. PMLR, 2022.

[DMZZ21] Christoph Dann, Mehryar Mohri, Tong Zhang, and Julian Zimmert. A provably efficient model-free posterior sampling method for episodic reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12040–12051, 2021.

References II

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[ICN+21] Haque Ishfaq, Qiwen Cui, Viet Nguyen, Alex Ayoub, Zhuoran Yang, Zhaoran Wang, Doina Precup, and Lin Yang. Randomized exploration in reinforcement learning with general value function approximation. In *International Conference on Machine Learning*, pages 4607–4616. PMLR, 2021.

[ILX+24] Haque Ishfaq, Qingfeng Lan, Pan Xu, A. Rupam Mahmood, Doina Precup, Anima Anandkumar, and Kamyar Azizzadenesheli. Provable and practical: Efficient exploration in reinforcement learning via langevin monte carlo. In *The Twelfth International Conference on Learning Representations*, 2024.

[Kak03] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. University of London, University College London (United Kingdom), 2003.

References III

[Li24a] Yingru Li. Probability Tools for Sequential Random Projection. 2024.

[Li24b] Yingru Li. Simple, unified analysis of Johnson-Lindenstrauss with applications. under review, 2024.

[LLZ+22] Ziniu Li, Yingru Li, Yushun Zhang, Tong Zhang, and Zhi-Quan Luo. HyperDQN: A randomized exploration method for deep reinforcement learning. In *International Conference on Learning Representations*, 2022.

[LXL24] Yingru Li, Jiawei Xu, and Zhi-Quan Luo. Approximate Thompson sampling via Hypermodel and Index sampling. To appear on arXiv, 2024.

[OAC18] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[OVRRW19] Ian Osband, Benjamin Van Roy, Daniel J. Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019.

References IV

[OWA+23] Ian Osband, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy. Approximate thompson sampling via epistemic neural networks. *arXiv preprint arXiv:2302.09205*, 2023.

[SCC+23] Max Schwarzer, Johan Samir Obando Ceron, Aaron Courville, Marc G Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pages 30365–30380. PMLR, 2023.

[Str07] Alexander L Strehl. *Probably approximately correct (PAC) exploration in reinforcement learning*. PhD thesis, Rutgers University-Graduate School-New Brunswick, 2007.

[TMS+23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

References V

[VHGS16] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[YAR+23] Zhewei Yao, Reza Yazdani Aminabadi, Olatunji Ruwase, Samyam Rajbhandari, Xiaoxia Wu, Ammar Ahmad Awan, Jeff Rasley, Minjia Zhang, Conglong Li, Connor Holmes, et al. Deepspeed-chat: Easy, fast and affordable rlhf training of chatgpt-like models at all scales. *arXiv preprint arXiv:2308.01320*, 2023.

[Zha22] Tong Zhang. Feel-good thompson sampling for contextual bandits and reinforcement learning. *SIAM Journal on Mathematics of Data Science*, 4(2):834–857, 2022.

[ZXZ+22] Han Zhong, Wei Xiong, Sirui Zheng, Liwei Wang, Zhaoran Wang, Zhuoran Yang, and Tong Zhang. Gec: A unified framework for interactive decision making in mdp, pomdp, and beyond. *arXiv preprint arXiv:2211.01962*, 2022.