Provable and Practical: Efficient Exploration in Reinforcement Learning via Langevin Monte Carlo

Haque Ishfaq * [1,2] ✉, Qingfeng Lan * [3], Pan Xu [4], Rupam Mahmood [3], Doina Precup [1,2], Anima Anandkumar [5,6], Kamyar Azizzadenesheli [6]

[1]Mila – Quebec AI Institute [2]McGill University [3]University of Alberta [4]Duke University [5]Caltech [6]Nvidia

Motivation

- Explorations techniques are crucial for an agent to be able to solve novel complex problems.
- Thompson sampling based on Laplace approximation is not a good estimation for the posterior distribution when the value function has more general forms than linearity.
- Sampling from a Gaussian distribution with general covariance matrix in high dimensional problem is computationally inefficient.

Highlights

- We propose a practical and efficient online RL algorithm **Langevin Monte Carlo Least-Squares Value Iteration (LMC-LSVI)**, which only needs to perform noisy gradient descent updates for exploration.
- We theoretically prove that **LMC-LSVI** achieves a $\widetilde{O}(d^{3/2}H^{3/2}\sqrt{T})$ regret under linear MDP settings, where d is the dimension of the feature mapping, H is the planning horizon, and T is the total number of steps.
- We further propose, **Adam Langevin Monte Carlo Deep Q-Network (Adam LMCDQN)**, a preconditioned variant of LMC-LSVI based on the Adam optimizer, which provides improved empirical performance.

Setting

- We consider online finite horizon MDPs $(\mathcal{S}, \mathcal{A}, H, \mathbb{P}, r)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, H is the horizon length, $\mathbb{P}$ is the state transition kernel and r is the reward function.
- **Value function** and **Action-value function** of policy π:

$$V_h^\pi(x) = \mathbb{E}_\pi\left[\sum_{h'=h}^{H} r_{h'}(x_{h'}, a_{h'}) \,\Big|\, x_h = x\right], \qquad Q_h^\pi(x,a) = \mathbb{E}_\pi\left[\sum_{h'=h}^{H} r_{h'}(x_{h'}, a_{h'}) \,\Big|\, x_h = x, a_h = a\right].$$

- Any algorithm can be measured by it's **regret**

$$\text{Regret}(K) = \sum_{k=1}^{K}\left[V_1^*(x_1^k) - V_1^{\pi^k}(x_1^k)\right].$$

Langevin Monte Carlo for Reinforcement Learning

- Define a general loss function

$$L_h^k(w_h) = \sum_{\tau=1}^{k-1}\left[r_h(x_h^\tau, a_h^\tau) + \max_{a\in\mathcal{A}} Q_{h+1}^k(x_{h+1}^\tau, a) - Q(w_h; \phi(x_h^\tau, a_h^\tau))\right]^2 + \lambda\|w_h\|^2$$

- Langevin Monte Carlo update:

$$w_{k+1} = w_k - \eta_k \nabla L(w_k) + \sqrt{2\eta_k\beta^{-1}}\epsilon_k,$$

  - It approximately samples from $\pi_k \propto \exp(-\beta L_k(w))$.
  - When Q is linear, $\pi_k = \mathcal{N}(\widehat{w}_k, \beta^{-1}\Lambda_k^{-1})$ where $\Lambda_k = \sum_{\tau=1}^{k-1}\phi(x_h^\tau, a_h^\tau)\phi(x_h^\tau, a_h^\tau)^\top + \lambda I$.
- LMC-LSVI approximately samples from the true posterior distribution.
- LMC-LSVI is computationally efficient due to
  - it only needs to sample from isotropic Gaussian $\mathcal{N}(0, I)$.
  - it only needs to perform noisy gradient descent updates.

Algorithm

Algorithm 1 Langevin Monte Carlo Least-Squares Value Iteration (LMC-LSVI)

Input: step sizes $\{\eta_k > 0\}_{k\ge1}$, inverse temperature $\{\beta_k\}_{k\ge1}$, loss function $L_k(w)$.
Initialize $w_h^{1,0} = \mathbf{0}$ for $h \in [H]$, $J_0 = 0$.
for episode $k = 1, 2, \ldots, K$ **do**
  Receive the initial state s_1^k.
  for step $h = H, H-1, \ldots, 1$ **do**
    $w_h^{k,0} = w_h^{k-1,J_{k-1}}$
    for $j = 1, \ldots, J_k$ **do**
      $\epsilon_h^{k,j} \sim \mathcal{N}(0, I)$
      $w_h^{k,j} = w_h^{k,j-1} - \eta_k \nabla L_h^k(w_h^{k,j-1}) + \sqrt{2\eta_k\beta_k^{-1}}\epsilon_h^{k,j}$
    end for
    $Q_h^k(\cdot,\cdot) \leftarrow \min\{Q(w_h^{k,J_k}; \phi(\cdot,\cdot)), H-h+1\}^+$
  end for
  for step $h = 1, 2, \ldots, H$ **do**
    Take action $a_h^k \leftarrow \operatorname{argmax}_{a\in\mathcal{A}} Q_h^k(s_h^k, a)$. Observe reward $r_h^k(s_h^k, a_h^k)$, get next state s_{h+1}^k.
  end for
end for

Theoretical Results

Theorem 1 (Regret bound for linear MDP). *For any $\delta \in (0,1)$ and appropriate β_k, η_k, under the assumption of linear MDP, the regret of Algorithm 1 satisfies*

$$\text{Regret}(K) = \widetilde{O}(d^{3/2}H^{3/2}\sqrt{T}),$$

with probability at least $1-\delta$.

Table 1. Regret upper bound for episodic, non-stationary, linear MDPs.

Algorithm	Regret	Exploration	Computational Efficiency	Scalability
LSVI-UCB [Jin et al., 2020]	$\widetilde{O}(d^{3/2}H^{3/2}\sqrt{T})$	UCB	Yes	No
OPT-RLSVI [Zanette et al., 2020]	$\widetilde{O}(d^2H^2\sqrt{T})$	TS	Yes	No
ELEANOR [Zanette et al., 2020]	$\widetilde{O}(dH^{3/2}\sqrt{T})$	Optimism	No	No
LSVI-PHE [Ishfaq et al., 2021]	$\widetilde{O}(d^{3/2}H^{3/2}\sqrt{T})$	TS	Yes	No
LMC-LSVI (this paper)	$\widetilde{O}(d^{3/2}H^{3/2}\sqrt{T})$	LMC	Yes	Yes

Deep Q-Network with LMC Exploration

Algorithm 2 Adam LMCDQN Update

for step $h = H, H-1, \ldots, 1$ **do**
  $w_h^{k,0} = w_h^{k-1,J_{k-1}}, m_h^{k,0} = m_h^{k-1,J_{k-1}}, v_h^{k,0} = v_h^{k-1,J_{k-1}}$
  for $j = 1, \ldots, J_k$ **do**
    $\epsilon_h^{k,j} \sim \mathcal{N}(0, I)$
    $w_h^{k,j} = w_h^{k,j-1} - \eta_k\left(\nabla\widetilde{L}_h^k(w_h^{k,j-1}) + a m_h^{k,j-1} \oslash \sqrt{v_h^{k,j-1} + \lambda_1\mathbf{1}}\right) + \sqrt{2\eta_k\beta_k^{-1}}\epsilon_h^{k,j}$
    $m_h^{k,j} = \alpha_1 m_h^{k,j-1} + (1-\alpha_1)\nabla\widetilde{L}_h^k(w_h^{k,j-1})$
    $v_h^{k,j} = \alpha_2 v_h^{k,j-1} + (1-\alpha_2)\nabla\widetilde{L}_h^k(w_h^{k,j-1}) \odot \nabla\widetilde{L}_h^k(w_h^{k,j-1})$
  end for
end for

Experiments

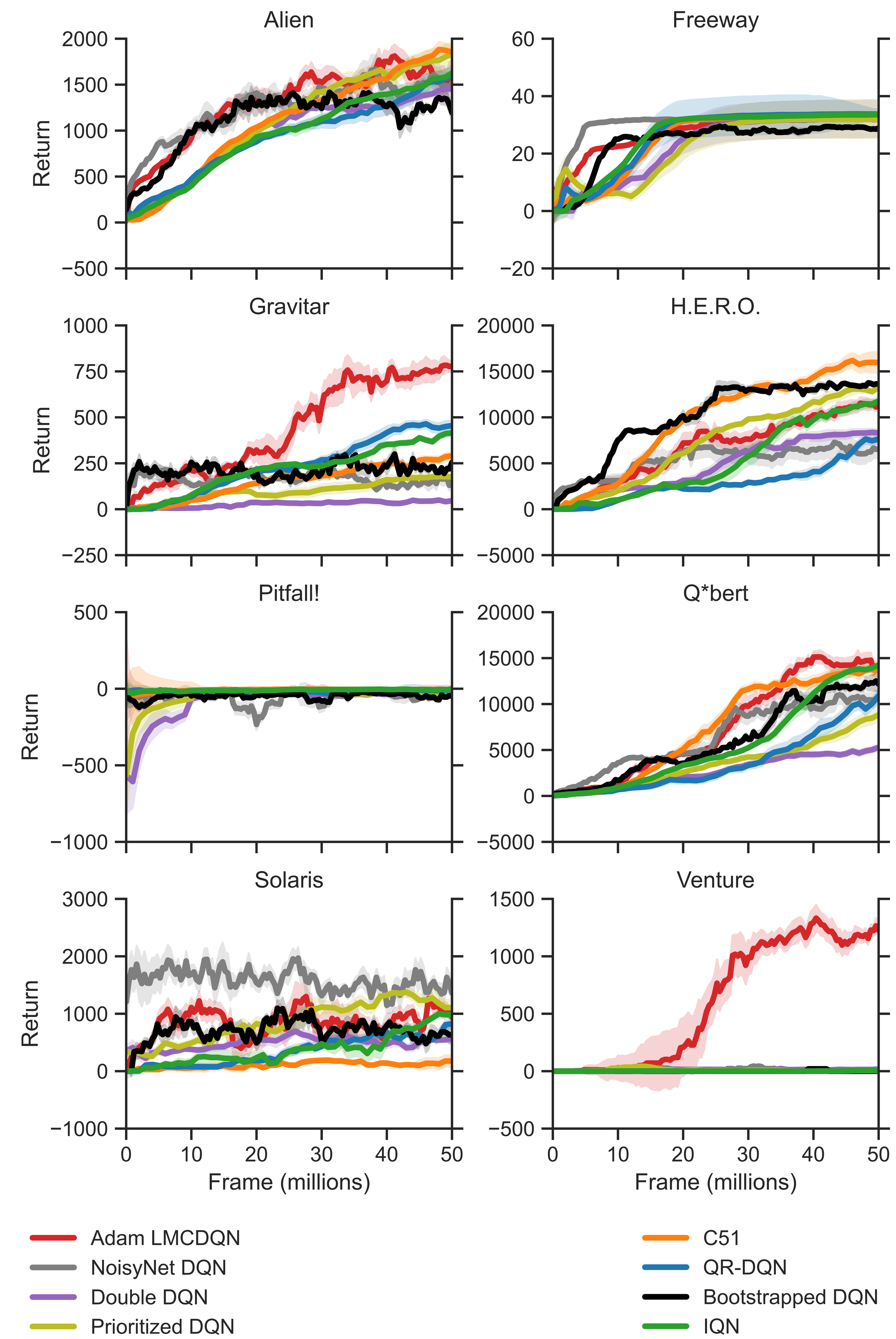

Figure 1. Return curves of various algorithms in Atari tasks over 50 million training frames. Solid lines correspond to the median performance over 5 random seeds, and the shaded areas correspond to 90% confidence interval.

✉ haque.ishfaq@mail.mcgill.ca