JOHNS HOPKINS
UNIVERSITY

VIPeR: Provably Efficient Algorithm for Offline RL with Neural Function Approximation

Thanh Nguyen-Tang & Raman Arora

Department of Computer Science, Johns Hopkins University

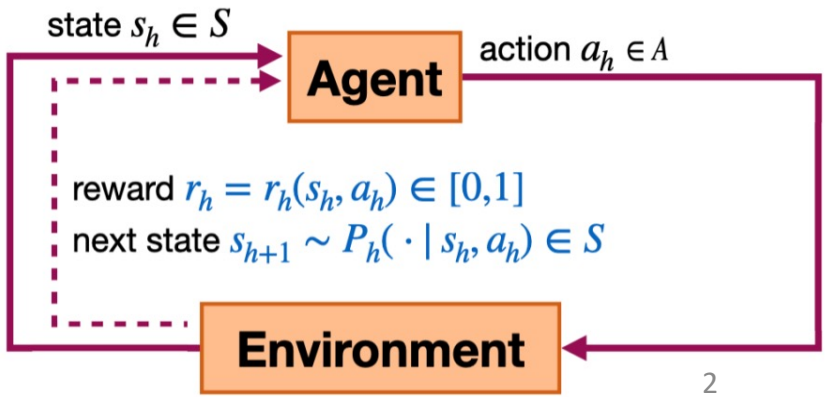
*top 25% noble papers, ICLR'23

Episodic MDP

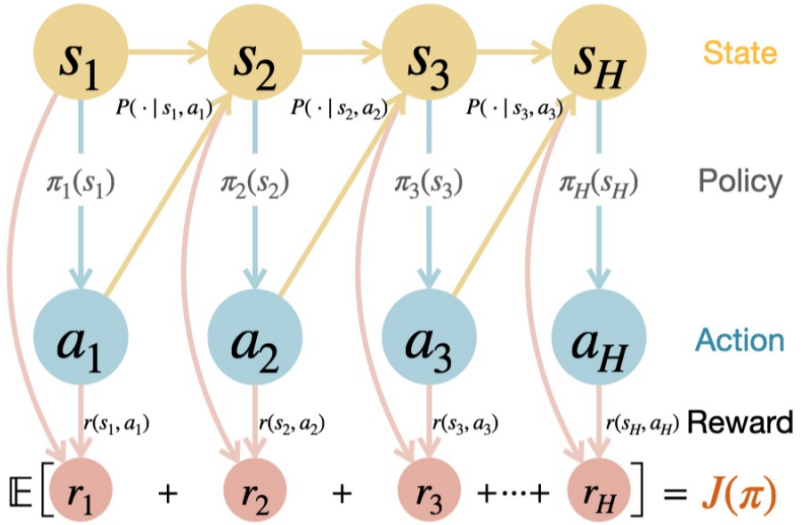
- Episodic time-inhomogeneous Markov decision process

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathrm{H}, \mathrm{P}, \mathrm{r}, \mathrm{d}_1)$$

  - State space $\mathcal{S}$: finite but exponentially large
  - Action space $\mathcal{A}$: finite but exponentially large
  - Episode length H: finite
    - Agent interacts with MDP for H steps and then restart the episode
  - Unknown Transition kernels $\mathrm{P} = (\mathrm{P}_1, \ldots, \mathrm{P}_\mathrm{H})$, where $\mathrm{P}_\mathrm{h} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
  - Unknown Reward functions $\mathrm{r} = (\mathrm{r}_1, \ldots, \mathrm{r}_\mathrm{H})$, where $\mathrm{r}_\mathrm{h} : \mathcal{S} \times \mathcal{A} \to [0,1]$
  - Initial state distribution $\mathrm{d}_1 \in \Delta(\mathcal{S})$



Episodic MDP



- A policy $\pi = \{\pi_h\}_{h\in[H]}$ where $\pi_h : \mathcal{S} \to \Delta(\mathcal{A})$
- Action-value functions:

$$Q_h^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{i=1}^H r_i | (s_h, a_h) = (s, a)\right]$$

- Value functions

$$V_h^\pi(s) = \mathbb{E}_\pi\left[\sum_{i=1}^H r_i | \, s_h = s\right]$$

- The optimal policy π^* maximizes V_1^π

Offline RL

- Offline dataset: collected a priori, $\mathcal{D} = \{(s_h^t, a_h^t, r_h^t)\}_{h\in[H]}^{t\in[K]}$
  - $a_h^t \sim \mu_h(\cdot|s_h^t)$, $s_{h+1}^t \sim P_h(\cdot|s_h^t, a_h^t)$
  - μ is the behavior policy
  - K: # number of episodes
- No further interactions with MDP
- Learning objective:

$$\mathrm{SubOpt}(\hat{\pi}; s_1) = V_1^*(s_1) - V_1^{\hat{\pi}}(s_1)$$

where $\hat{\pi} = \mathrm{OfflineRLAlgo}(\mathcal{D}, \mathcal{F})$, $\mathcal{F}$ is some function class (e.g., neural networks)

PEVI algorithm (Jin et al., ICML'21)

- **Low-rank** MDPs:

$$r_{h(s,a)} = \phi_h(s,a)^T w_h, P_h(s'|s,a) = \phi_h(s,a)^T \nu_h(s')$$

- **Pessimism** design: Computing **LCB (lower confidence bound)**

$$\widehat{Q}_h(s,a) = \phi_h(s,a)^T \widehat{w}_h - \beta||\phi_h(s,a)||_{\Lambda_h^{-1}}$$

  - $\Lambda_h = \lambda I + \sum_{k=1}^{K} \phi_h(s_h^k, a_h^k)\, \phi_h(s_h^k, a_h^k)^T$
  - $\widehat{w}_h$: ERM

- Then, extract $\widehat{\pi}_h$ greedy w.r.t. $\widehat{Q}_h$
- PEVI is both **statistically efficient** and **computationally efficient** in low-rank MDPs
- PEVI require data coverage **only over an optimal policy**

[Jin et al., ICML'21] "*Is pessimism provably efficient for offline rl?*"

Offline RL with Neural Function Approximation

However, in many **practical** settings, MDPs do **not** admit any **linear** structures

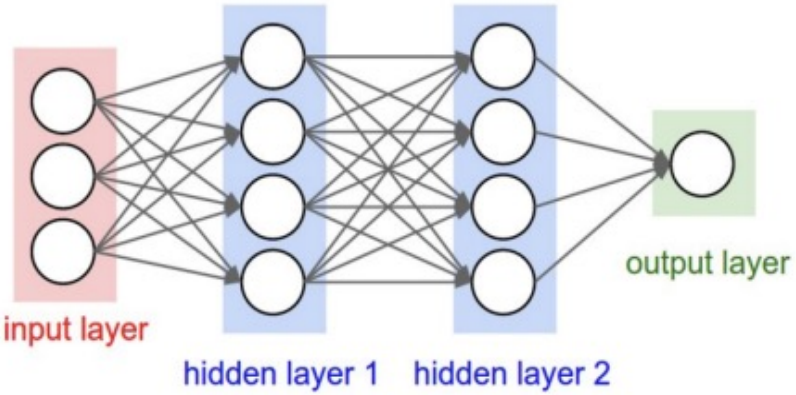
→ Desirable to use differentiable function approximator such as **neural networks**

Background: Neural networks

- Action-value functions are approximated by a two-layer neural net

$$f(x; W) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} b_i \cdot \mathrm{ReLU}(w_i^T x)$$

  - $x = (s, a) \in \mathcal{X} = \mathcal{S} \times \mathcal{A} \in \mathbb{R}^d$
  - m: network width
  - $b_i \sim \mathrm{Unif}(\{-1, 1\})$: fixed during training
  - $W = (w_1, w_2, \ldots, w_m) \in \mathbb{R}^{md}$: trainable
  - W_0: initialization of W
    - $w_i \sim \mathcal{N}(0, \frac{I_d}{d})$



NeuraLCB (Nguyen-Tang et al., ICLR)

LCB + **neural** function approximation

- Use neural network $\mathrm{f}(\mathrm{s}, \mathrm{a}; \mathrm{W})$ to approximate the action-value function
- Computing **LCB**:

$$\widehat{\mathrm{Q}}_{\mathrm{h}}(\mathrm{s}, \mathrm{a}) = \mathrm{f}\left(\mathrm{s}, \mathrm{a}; \widehat{\mathrm{W}}\right) - \beta||\nabla_{\mathrm{W}}\mathrm{f}\left(\mathrm{s}, \mathrm{a}; \widehat{\mathrm{W}}\right)||_{\Lambda_{\mathrm{h}}^{-1}}$$

  - $\widehat{\mathrm{W}}$: found by gradient descent
  - $\nabla_W f\left(s, a; \widehat{W}\right)$: the gradient of the neural network w.r.t. its parameters
  - $\Lambda_{\mathrm{h}} = \lambda \mathrm{I} + \sum_{\mathrm{k}=1}^{\mathrm{K}} \nabla_{\mathrm{W}}\mathrm{f}\left(\mathrm{s}_{\mathrm{h}}^{\mathrm{k}}, \mathrm{a}_{\mathrm{h}}^{\mathrm{k}}; \widehat{\mathrm{W}}\right)\left(\nabla_{\mathrm{W}}\mathrm{f}\left(\mathrm{s}_{\mathrm{h}}^{\mathrm{k}}, \mathrm{a}_{\mathrm{h}}^{\mathrm{k}}; \widehat{\mathrm{W}}\right)\right)^{\mathrm{T}}$ is the empirical covariance matrix
- Complexity of computing LCB: (# of network parameters)^2

[Nguyen-Tang et al., ICLR'22] "Offline neural contextual bandits: Pessimism, Optimization, and Generalization"

VIPeR: Implicit pessimism via reward perturbing

- End of Episode: $\widetilde{Q}_{H+1} \leftarrow 0$
- Bootstrapping (backward induction): for $h = H, H-1, \dots, 1$
  - Perturb the rewards with independent Gaussian noises $\xi_h^{k,i} \sim \mathcal{N}(0, \sigma^2)$ to create M perturbed datasets $\widetilde{\mathcal{D}}_h^1, \widetilde{\mathcal{D}}_h^2, \dots, \widetilde{\mathcal{D}}_h^M$

$$\widetilde{\mathcal{D}}_h^i = \{(\underbrace{s_h^k, a_h^k}_{\text{input}}, \underbrace{r_h^k + \widetilde{V}_{h+1}(s_{h+1}^k) + \xi_h^{k,i}}_{\textbf{perturbed output}})\}_{k \in [K]}$$

  - Train a different neural network in each $\widetilde{\mathcal{D}}_h^i$ to get $f(s, a; W^i)$
  - Pessimistically estimate

$$\widetilde{Q}_h(s, a) = \min_{i \in [M]} f(s, a; W^i)$$

  - Optimize $\widetilde{\pi}_h(s) \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} \widetilde{Q}_h(s, a)$ and $\widetilde{V}_h(s) = \max_a \widetilde{Q}_h(s, a)$
  - Move to step h-1 and repeat

Value suboptimality of VIPeR

- Conditions:
  - Adaptively collected data with coverage of only an optimal policy
  - Completeness assumption
  - The ensemble size M is **polylogarithmically** large
  - The noise level $\sigma = \tilde{\mathcal{O}}(H\ \mathrm{d_{eff}})$
  - Small learning rate and sufficiently large training iterations
  - The network width m is polynomially large
- Then,

$$\mathrm{SubOpt}(\widetilde{\pi}; \mathrm{s}_1) \lesssim \sigma \mathbb{E}_{\pi^*}\left[\sum_{\mathrm{h}=1}^{\mathrm{H}} ||\nabla_{\mathrm{W}} \mathrm{f}(\mathrm{s}_\mathrm{h}, \mathrm{a}_\mathrm{h}; \widehat{\mathrm{W}})||_{\Lambda_\mathrm{h}^{-1}}\right]$$

- Same guarantee as **NeuralLCB**, but **no** need to compute LCB

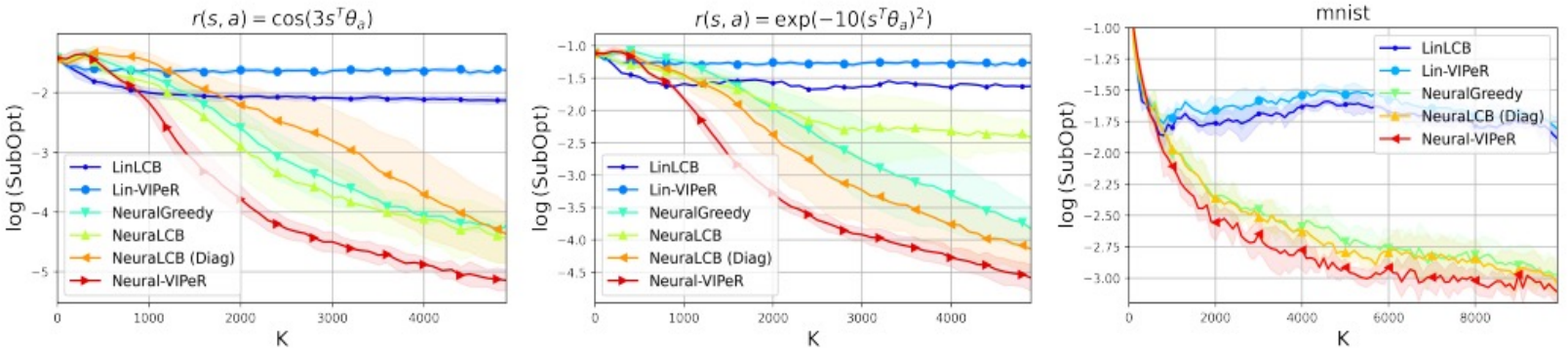
Comparison

work	bound	i.i.d?	explorative data?	finite spectrum?	matrix inverse?	opt
Jin et al. (2021)	$\tilde{\mathcal{O}}\left(\frac{d_{lin}^{3/2}H^2}{\sqrt{K}}\right)$	no	yes	yes	yes	analytical
Yang et al. (2020)	$\tilde{\mathcal{O}}\left(\frac{H^2\sqrt{\tilde{d}^2+\tilde{d}\tilde{n}}}{\sqrt{K}}\right)$	no	–	no	yes	oracle
Xu & Liang (2022)	$\tilde{\mathcal{O}}\left(\frac{\tilde{d}H^2}{\sqrt{K}}\right)$	yes	yes	yes	yes	oracle
This work	$\tilde{\mathcal{O}}\left(\frac{\kappa H^{5/2}\tilde{d}}{\sqrt{K}}\right)$	**no**	**no**	**no**	**no**	**GD**

Table 1: SOTA results for offline RL with function approximation. The third and fourth columns ask if the corresponding result needs the data to be i.i.d, and well-explored, resp.; the fifth column asks if the induced RKHS needs to have a finite spectrum, the sixth column asks if the algorithm needs to inverse a covariance matrix and the last column presents the optimizer being used. Here $\tilde{n}$ is the log covering number.

Experiment: Neural Contextual Bandits

- **LinLCB**: LCB + linear
- **Lin-VIPeR**: Reward perturbing + linear
- **NeuralGreedy**: FQI + neural
- **NeuraLCB**: LCB + neural
- **Neural-VIPeR**: reward perturbing + neural



(a) (b) (c)

Figure 3: Sub-optimality (on log-scale) vs. sample size (K) for neural contextual bandits with following reward functions: (a) $r(s,a) = \cos(3s^T\theta_a)$, (b) $r(s,a) = \exp(-10(s^T\theta_a)^2)$, and (c) MNIST.

Runtime efficiency

- **NeuralLCB** spends $O(K^2)$ time in action selection
- **Neural-VIPeR** spends $O(1)$ in action selection

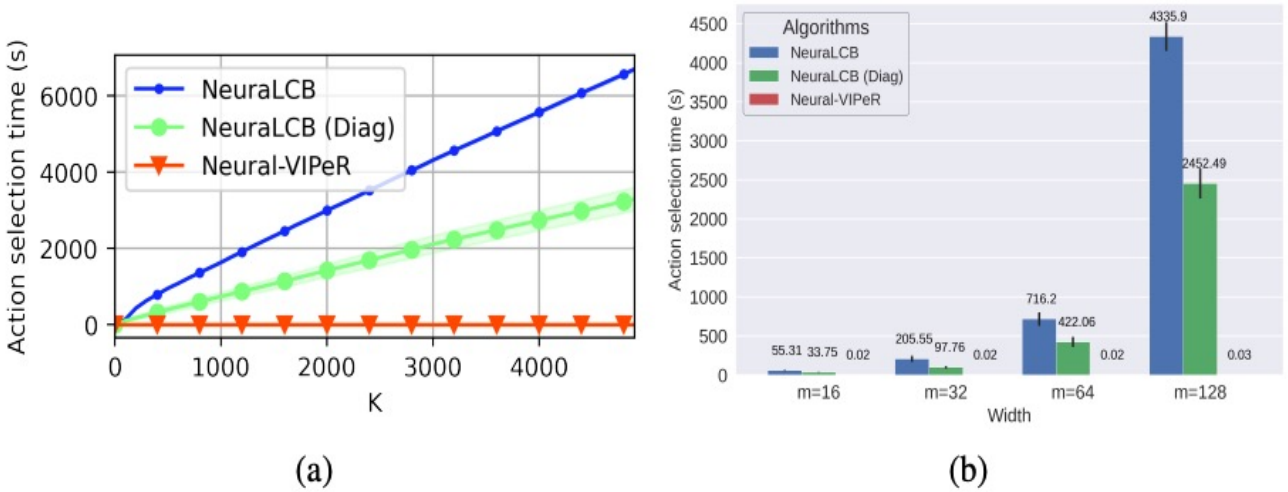


Figure 4: Elapsed time (in seconds) for action selection in the contextual bandits problem with $r(s,a) = 10(s^T\theta_a)^2$: (a) Runtime of action selection versus the number of (offline) data points K, and (b) runtime of action selection versus the network width m (for $K = 500$).

Controlling pessimism

M controls the level of pessimism and nicely correlates with the decrease of subopt in practice

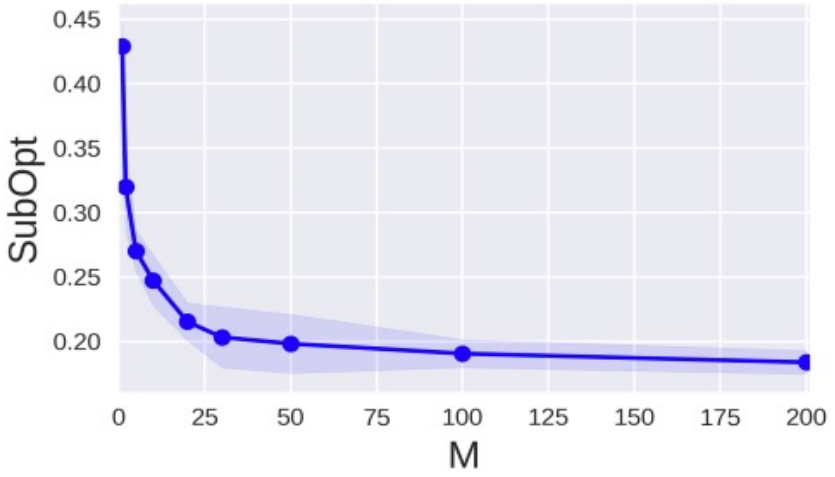


Figure 5: Sub-optimality of NeuralPER versus different values of M.

Result in D4RL

- **BEAR** (Kumar et al., 2019): use MMD distance to constraint policy to the offline data
- **UWAC** (Wu et al., 2021): improves BEAR using dropout uncertainty
- **CQL** (Kumar et al., 2020) that minimizes Q-values of OOD actions
- **MOPO** (Yu et al., 2020): uses model-based uncertainty via ensemble dynamics
- **TD3-BC** (Fujimoto & Gu, 2021): uses adaptive behavior cloning
- **PBRL** (Bai et al., 2022): use uncertainty quantification via disagreement of bootstrapped Q-functions

		BEAR	UWAC	CQL	MOPO	TD3-BC	PBRL	VIPeR
Random	HalfCheetah	2.3 ±0.0	2.3 ±0.0	17.5 ±1.5	**35.9** ±2.9	11.0 ±1.1	11.0 ±5.8	14.5 ±2.1
	Hopper	3.9 ±2.3	2.7 ±0.3	7.9 ±0.4	16.7 ±12.2	8.5 ±0.6	26.8 ±9.3	**31.4** ±0.0
	Walker2d	12.8 ±10.2	2.0 ±0.4	5.1 ±1.3	4.2 ±5.7	1.6 ±1.7	8.1 ±4.4	**20.5** ±0.5
Medium	HalfCheetah	43.0 ±0.2	42.2 ±0.4	47.0 ±0.5	**73.1** ±2.4	48.3 ±0.3	57.9 ±1.5	58.5 ±1.1
	Hopper	51.8 ±4.0	50.9 ±4.4	53.0 ±28.5	38.3 ±34.9	59.3 ±4.2	75.3 ±31.2	**99.4** ±6.2
	Walker2d	-0.2 ±0.1	75.4 ±3.0	73.3 ±17.7	41.2 ±30.8	83.7 ±2.1	**89.6** ±0.7	**89.6** ±1.2
Medium Replay	HalfCheetah	36.3 ±3.1	35.9 ±3.7	45.5 ±0.7	**69.2** ±1.1	44.6 ±0.5	45.1 ±8.0	45.0 ±8.6
	Hopper	52.2 ±19.3	25.3 ±1.7	88.7 ±12.9	32.7 ±9.4	60.9 ±18.8	**100.6** ±1.0	**100.2** ±1.0
	Walker2d	7.0 ±7.8	23.6 ±6.9	81.8 ±2.7	73.7 ±9.4	81.8 ±5.5	77.7 ±14.5	**83.1** ±4.2
Medium Expert	HalfCheetah	46.0 ±4.7	42.7 ±0.3	75.6 ±25.7	70.3 ±21.9	90.7 ±4.3	92.3 ±1.1	**94.2** ±1.2
	Hopper	50.6 ±25.3	44.9 ±8.1	105.6 ±12.9	60.6 ±32.5	98.0 ±9.4	**110.8** ±0.8	**110.6** ±1.0
	Walker2d	22.1 ±44.9	96.5 ±9.1	107.9 ±1.6	77.4 ±27.9	**110.1** ±0.5	**110.1** ±0.3	**109.8** ±0.5
Expert	HalfCheetah	92.7 ±0.6	92.9 ±0.6	96.3 ±1.3	81.3 ±21.8	96.7 ±1.1	92.4 ±1.7	**97.4** ±0.9
	Hopper	54.6 ±21.0	110.5 ±0.5	96.5 ±28.0	62.5 ±29.0	107.8 ±7	**110.5** ±0.4	**110.8** ±0.4
	Walker2d	106.6 ±6.8	108.4 ±0.4	108.5 ±0.5	62.4 ±3.2	**110.2** ±0.3	108.3 ±0.3	108.3 ±0.2
	Average	38.78 ±10.0	50.41 ±2.7	67.35 ±9.1	53.3 ±16.3	67.55 ±3.8	74.37 ±5.3	**78.2** ±1.9

Table 2: Average normalized score and standard deviation of all algorithms over five seeds in the Gym domain in the "v2" dataset of D4RL (Fu et al., 2020). The scores for all the baselines are from Table 1 of Bai et al. (2022). The highest scores are highlighted.

Summary

We now have a **provably efficient & computationally efficient algorithm** for ***neural*** function approximation with polynomial sample and runtime under *mild data coverage*

Algorithm: perturbed rewards + (Stochastic) gradient descent

Sample complexity: $\mathrm{SubOpt}(\tilde{\pi}) = \tilde{\mathcal{O}}\left(\frac{H^{2.5} \cdot \kappa \cdot d_{eff}}{\sqrt{K}}\right)$

- K: # samples
- H: horizon
- κ: single-policy concentration coefficient
- d_{eff}: effective dimension

Future Work

- Match the lower bound in terms of κ?
- Avoid the need of training an ensemble of M models?
- General-purpose offline RL with any regression oracle?
- Can randomized value iteration obtain a horizon-free rate?
- Extension to the POMDP?
- Model-free posterior sampling for offline RL with optimal frequentist rates?

Thank you

More details at our paper: https://openreview.net/forum?id=WOquZTLCBO1