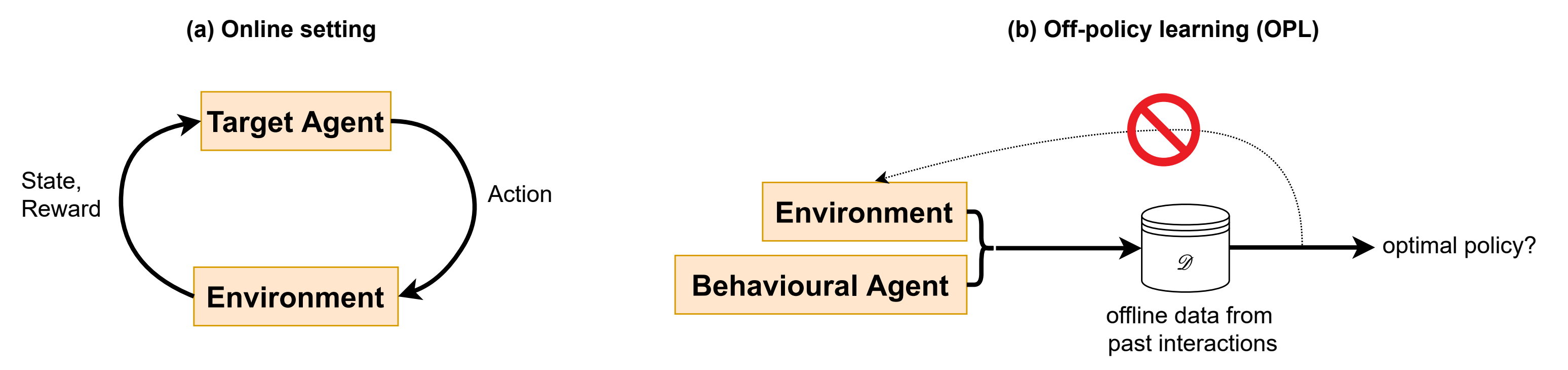
Offline Neural Contextual Bandits: Pessimism, Optimization and Generalization

Thanh Nguyen-Tang [1] Sunil Gupta [1] A.Tuan Nguyen [2] Svetha Venkatesh [1]

[1]Applied AI Institute, Deakin University, Australia [2]Department of Engineering Science, University of Oxford, UK

Motivation

Problem context: Offline policy learning (OPL) where a learner infers an optimal policy given only access to a fixed dataset collected a priori by unknown behaviour policies, without any active exploration



Key challenges

Despite the importance of OPL, theoretical and algorithmic progress on this problem has been rather limited:

- Distributional shift: The prior analyses require the offline policy to be already sufficiently explorative over the entire state space and action space, and to be stationary
- Optimization: The prior analyses rely on simple function approximations such as tabular representation or linear models with closed-form solutions. Using neural network function approximation increases the expressiveness but poses an additional challenge in optimization.
- Generalization: As we learn from the fixed offline data, the learned policy has highly correlated structures where we cannot directly use the standard concentration inequalities to derive a generalization bound

Question: Can we design **practical** OPL algorithms with neural network function approximation with **stronger** theoretical guarantees and **milder** conditions?

Contributions

In this paper, we consider the problem of OPL with neural network function approximation on the axes of distributional shift, optimization and generalization via studying the setting of stochastic contextual bandits with overparameterized neural networks. Specifically, we make three contributions toward enabling OPL in more practical settings:

- Algorithmically, we proposed an algorithm that uses a neural network to model any bounded reward function without assuming any functional form (e.g., linear models) and uses a pessimistic formulation to deal with distributional shifts.
- Theoretically, we proved that our algorithm learns the optimal policy with an expected error of $\tilde{\mathcal{O}}(\kappa \tilde{d}^{1/2} n^{-1/2})$, where n is the number of sample, κ is a measure of distributional shift, and $\tilde{d}$ is the "effective dimension" of the neural network.
  - Notably, this result was established under milder data coverage condition than the prior OPL works, with a reduction in computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, and an improved the dependency on $\tilde{d}$, as compared to an online counterpart.
- Empirically, we evaluated our algorithm in a number of synthetic and real-world OPL benchmark problems, verifying its empirical effectiveness against the representative methods of OPL.

Background

Stochastic Contextual Bandits: We consider a stochastic K-armed contextual bandit where at each round t, an online learner observes a full context $x_t := \{x_{t,a} \in \mathbb{R}^d : a \in [K]\}$ sampled from a context distribution ρ, takes an action $a_t \in [K]$, and receives a reward $r_t \sim P(\cdot|x_{t,a_t})$.

A policy π maps a full context (and possibly other past information) to a distribution over the action space $[K]$. For each full context $x := \{x_a \in \mathbb{R}^d : a \in [K]\}$, we define $v^\pi(x) = \mathbb{E}_{a \sim \pi(\cdot|x), r \sim P(\cdot|x_a)}[r]$ and $v^*(x) = \max_\pi v^\pi(x)$

Offline Contextual Bandit Setting: The goal in this setting is to learn an optimal policy only from an offline data $\mathcal{D}_n = \{(x_t, a_t, r_t)\}_{t=1}^n$ collected a priori by a behaviour policy μ. The goodness of a learned policy $\hat{\pi}$ is measured by the (expected) sub-optimality the policy achieves in the entire (unknown) context distribution ρ:

$$\text{subopt}(\hat{\pi}) := \mathbb{E}_{x \sim \rho}[\text{subopt}(\hat{\pi}; x)], \text{ where } \text{subopt}(\hat{\pi}; x) := v^*(x) - v^{\hat{\pi}}(x).$$

Neural networks: We consider fully connected neural networks with depth $L \geq 2$ defined on $\mathbb{R}^d$ as

$$f_W(u) = \sqrt{m} W_L \sigma(W_{L-1}\sigma(\ldots\sigma(W_1 u)\ldots)), \forall u \in \mathbb{R}^d, \quad (1)$$

where $\sigma(\cdot) = \max\{\cdot, 0\}$ is the rectified linear unit (ReLU) activation function, $W_1 \in \mathbb{R}^{m \times d}, W_i \in \mathbb{R}^{m \times m}, \forall i \in [2, L-1], W_L \in \mathbb{R}^{m \times 1}$, and $W := (W_1, \ldots, W_L)$ with $(W) \in \mathbb{R}^p$ where $p = md + m + m^2(L-2)$.

Algorithm

Key highlights

- We used a neural network $f_W(x_a)$ to learn the reward function $h(x_a)$
- We constructed a lower confidence bound (LCB) of the reward function based on which decision-making is guided
- The offline data is processed sequentially in an online-like manner

Algorithm 1 NeuraLCB

```
Input: Offline data $\mathcal{D}_n = \{(\boldsymbol{x}_t, a_t, r_t)\}_{t=1}^n$, step sizes $\{\eta_t\}_{t=1}^n$ , regularization parameter $\lambda > 0$,
    confidence parameters $\{\beta_t\}_{t=1}^n$.
1: Initialize $\boldsymbol{W}^{(0)}$ as follows: set $\boldsymbol{W}_l^{(0)} = [\bar{\boldsymbol{W}}_l, \mathbf{0}; \mathbf{0}, \bar{\boldsymbol{W}}_l], \forall l \in [L-1]$ where each entry of $\bar{\boldsymbol{W}}_l$
    is generated independently from $\mathcal{N}(0, 4/m)$, and set $\boldsymbol{W}_L^{(0)} = [\boldsymbol{w}^T, -\boldsymbol{w}^T]$ where each entry of
    $\boldsymbol{w}$ is generated independently from $\mathcal{N}(0, 2/m)$.
2: $\boldsymbol{\Lambda}_0 \leftarrow \lambda \boldsymbol{I}$.
3: for $t = 1, \ldots, n$ do
4:     Retrieve $(\boldsymbol{x}_t, a_t, r_t)$ from $\mathcal{D}_n$.
5:     $\hat{\pi}_t(\boldsymbol{x}) \leftarrow \arg\max_{a \in [K]} L_t(\boldsymbol{x}_a)$, for all $\boldsymbol{x} = \{\boldsymbol{x}_a \in \mathbb{R}^d : a \in [K]\}$ where $L_t(\boldsymbol{u}) =$
    $f_{\boldsymbol{W}^{(t-1)}}(\boldsymbol{u}) - \beta_{t-1} \|\nabla f_{\boldsymbol{W}^{(t-1)}}(\boldsymbol{u}) \cdot m^{-1/2}\|_{\boldsymbol{\Lambda}_{t-1}^{-1}}, \forall \boldsymbol{u} \in \mathbb{R}^d$
6:     $\boldsymbol{\Lambda}_t \leftarrow \boldsymbol{\Lambda}_{t-1} + \text{vec}(\nabla f_{\boldsymbol{W}^{(t-1)}}(\boldsymbol{x}_{t,a_t})) \cdot \text{vec}(\nabla f_{\boldsymbol{W}^{(t-1)}}(\boldsymbol{x}_{t,a_t}))^T / m$.
7:     $\boldsymbol{W}^{(t)} \leftarrow \boldsymbol{W}^{(t-1)} - \eta_t \nabla \mathcal{L}_t(\boldsymbol{W}^{(t-1)})$ where $\mathcal{L}_t(\boldsymbol{W}) = \frac{1}{2}(f_{\boldsymbol{W}}(\boldsymbol{x}_{t,a_t}) - r_t)^2 + \frac{m\lambda}{2}\|\boldsymbol{W} -$
    $\boldsymbol{W}^{(0)}\|_F^2$.
8: end for
Output: Randomly sample $\hat{\pi}$ uniformly from $\{\hat{\pi}_1, \ldots, \hat{\pi}_n\}$.
```

Theoretical Analysis

Assumption 1: $\exists \lambda_0 > 0, H \succeq \lambda_0 I$ where H is the neural tangent kernel (NTK) matrix of the neural network in Algorithm 1.

Assumption 2: $\forall t, x_t$ is independent of $\mathcal{D}_{t-1}$, and $\exists \kappa \in (0, \infty)$, $\left\| \frac{\pi^*(\cdot|x_t)}{\mu(\cdot|\mathcal{D}_{t-1}, x_t)} \right\|_\infty \leq \kappa, \forall t \in [n]$.

Theoretical Analysis (con't)

Table 1: The SOTA generalization theory of OPL with function approximation. Here the distributional shift measure κ can be defined differently in different works.

Work	Function	Type	Optimization	Sub-optimality	Data Coverage	Data Gen.
Yin & Wang (2020)[a]	Tabular	Greedy	Analytical	$\tilde{\mathcal{O}}\left(\sqrt{\lvert\mathcal{X}\rvert \cdot K} \cdot n^{-1/2}\right)$	Uniform	I
Rashidinejad et al. (2021)	Tabular	Pessimism	Analytical	$\tilde{\mathcal{O}}\left(\sqrt{\lvert\mathcal{X}\rvert \cdot \kappa} \cdot n^{-1/2}\right)$	SPC	I
Duan & Wang (2020)[b]	Linear	Greedy	Analytical	$\tilde{\mathcal{O}}\left(\kappa \cdot n^{-1/2} + d \cdot n^{-1}\right)$	Uniform	I
Jin et al. (2020)	Linear	Pessimism	Analytical	$\tilde{\mathcal{O}}\left(d \cdot n^{-1/2}\right)$	Uniform	I
Nguyen-Tang et al. (2021)	Narrow ReLU	Greedy	Oracle	$\tilde{\mathcal{O}}\left(\sqrt{\kappa} \cdot n^{-\frac{\alpha}{2(\alpha+d)}}\right)$	Uniform	I
This work	Wide ReLU	Pessimism	SGD	$\tilde{\mathcal{O}}(\kappa \cdot \sqrt{\tilde{d}} \cdot n^{-1/2})$	eSPC	I/D

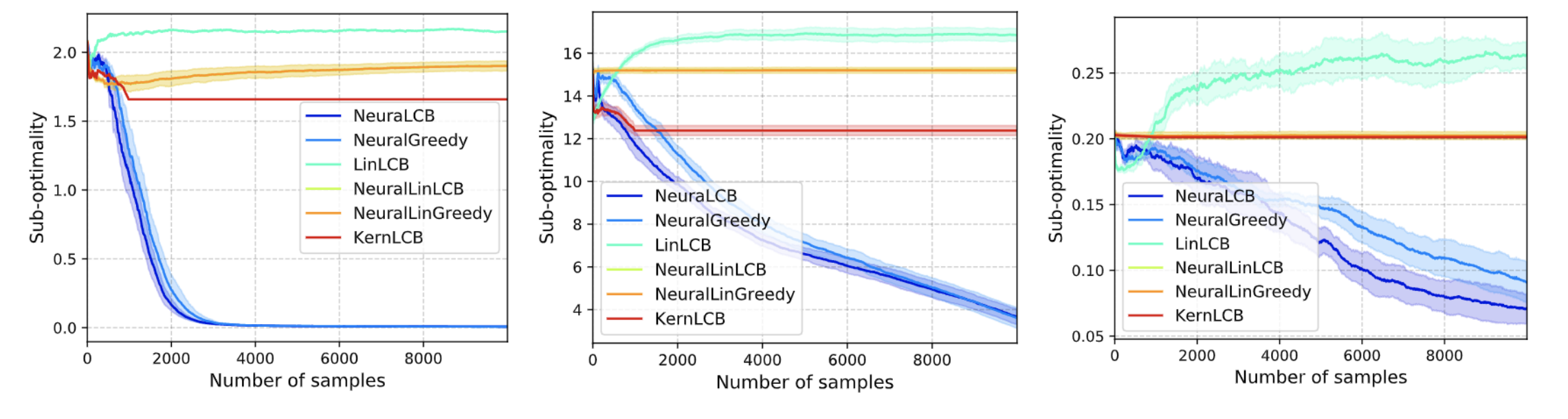
[a,b] The bounds of these works are for off-policy evaluation which is generally easier than OPL problem.

Key highlights

- Our bound does not scale with p, but linearly with $\sqrt{\tilde{d}}$, where $\tilde{d} = \frac{\log\det(I + H/\lambda)}{\log(1 + nK/\lambda)}$
- Our bound does not require the offline policy to be uniformly explorative or stationary
- We reduced computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$, and our bound improves by a factor of $\sqrt{\tilde{d}}$, as compared to its online counterpart.

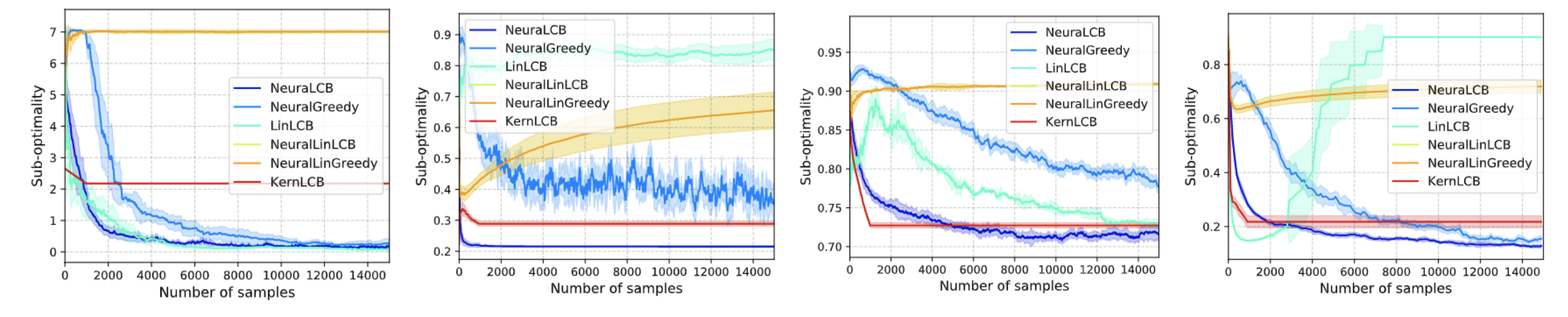
Experiments

We evaluated NeuraLCB and representative baseline methods on both synthetic and real-world datasets. (1) **LinLCB** uses LCB but relies on linear models, (2) **KernLCB** uses RKHS as function approximation, (3) **NeuralLinLCB** is the same as LinLCB except that it uses $\phi(x_a) = (\nabla f_{W^{(0)}}(x_a))$ as the feature extractor for the linear model, (4) **NeuralLinGreedy** is the same as NeuralLinLCB except that it relies on the empirical estimate of the reward function for decision-making, and (5) **NeuralGreedy** is the same as NeuraLCB except it makes decision based on the empirical estimate of the reward.



(a) $h(u) = 10(a^T u)^2$ (b) $h(u) = u^T A^T A u$ (c) $h(u) = cos(3a^T u)$

Figure 1: The sub-optimality of NeuraLCB versus the baseline algorithms on synthetic datasets.



(a) Mushroom (b) Statlog (c) Adult (d) MNIST

Figure 2: The sub-optimality of NeuraLCB versus the baseline algorithms on real-world datasets.

Observation: NeuraLCB outperforms the representative baselines in both synthetic and real-world experiments.

For more details, visit https://arxiv.org/abs/2111.13807