Neural Contextual Bandits with UCB-based Exploration

Dongruo Zhou [1] **Lihong Li** [2] **Quanquan Gu** [1]

Abstract

We study the stochastic contextual bandit problem, where the reward is generated from an unknown function with additive noise. No assumption is made about the reward function other than boundedness. We propose a new algorithm, NeuralUCB, which leverages the representation power of deep neural networks and uses a neural network-based random feature mapping to construct an upper confidence bound (UCB) of reward for efficient exploration. We prove that, under standard assumptions, NeuralUCB achieves $\widetilde{O}(\sqrt{T})$ regret, where T is the number of rounds. To the best of our knowledge, it is the first neural network-based contextual bandit algorithm with a near-optimal regret guarantee. We also show the algorithm is empirically competitive against representative baselines in a number of benchmarks.

1. Introduction

The stochastic contextual bandit problem has been extensively studied in machine learning (Langford & Zhang, 2008; Bubeck & Cesa-Bianchi, 2012; Lattimore & Szepesvári, 2019): at round $t \in \{1, 2, \ldots, T\}$, an agent is presented with a set of K actions, each of which is associated with a d-dimensional feature vector. After choosing an action, the agent will receive a stochastic reward generated from some unknown distribution conditioned on the action's feature vector. The goal of the agent is to maximize the expected cumulative rewards over T rounds. Contextual bandit algorithms have been applied to many real-world applications, such as personalized recommendation, advertising and Web search.

The most studied model in the literature is linear contextual bandits (Auer, 2002; Abe et al., 2003; Dani et al., 2008; Rusmevichientong & Tsitsiklis, 2010), which assumes that the expected reward at each round is linear in the feature vector. While successful in both theory and practice (Li et al., 2010; Chu et al., 2011; Abbasi-Yadkori et al., 2011), the linear-reward assumption it makes often fails to hold in practice, which motivates the study of nonlinear or nonparametric contextual bandits (Filippi et al., 2010; Srinivas et al., 2010; Bubeck et al., 2011; Valko et al., 2013). However, they still require fairly restrictive assumptions on the reward function. For instance, Filippi et al. (2010) make a generalized linear model assumption on the reward, Bubeck et al. (2011) require it to have a Lipschitz continuous property in a proper metric space, and Valko et al. (2013) assume the reward function belongs to some Reproducing Kernel Hilbert Space (RKHS).

In order to overcome the above shortcomings, deep neural networks (DNNs) (Goodfellow et al., 2016) have been introduced to learn the underlying reward function in contextual bandit problem, thanks to their strong representation power. We call these approaches collectively as *neural contextual bandit algorithms*. Given the fact that DNNs enable the agent to make use of nonlinear models with less domain knowledge, existing work (Riquelme et al., 2018; Zahavy & Mannor, 2019) study *neural-linear bandits*. That is, they use all but the last layers of a DNN as a feature map, which transforms contexts from the raw input space to a low-dimensional space, usually with better representation and less frequent updates. Then they learn a linear exploration policy on top of the last hidden layer of the DNN with more frequent updates. These attempts have achieved great empirical success, but no regret guarantees are provided.

In this paper, we consider *provably efficient* neural contextual bandit algorithms. The new algorithm, NeuralUCB, uses a neural network to learn the unknown reward function, and follows a UCB strategy for exploration. At the core of the algorithm is the novel use of DNN-based random feature mappings to construct the UCB. Its regret analysis is built on recent advances on optimization and generalization of deep neural networks (Jacot et al., 2018; Arora et al., 2019; Cao & Gu, 2019). Crucially, the analysis makes no modeling assumptions about the reward function, other than that it be bounded. While the main focus of our paper is theoretical, we also show in a few benchmark problems the effectiveness of NeuralUCB, and demonstrate its benefits against several representative baselines.

[1]Department of Computer Science, University of California, Los Angeles, CA 90095, USA [2]Google Research, USA. Correspondence to: Quanquan Gu <qgu@cs.ucla.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Our main contributions are as follows:

- We propose a neural contextual bandit algorithm that can be regarded as an extension of existing (generalized) linear bandit algorithms (Abbasi-Yadkori et al., 2011; Filippi et al., 2010; Li et al., 2010; 2017) to the case of arbitrary bounded reward functions.
- We prove that, under standard assumptions, our algorithm is able to achieve $\widetilde{O}(\widetilde{d}\sqrt{T})$ regret, where $\widetilde{d}$ is the effective dimension of a neural tangent kernel matrix and T is the number of rounds. The bound recovers the existing $\widetilde{O}(d\sqrt{T})$ regret for linear contextual bandit as a special case (Abbasi-Yadkori et al., 2011), where d is the dimension of context.
- We demonstrate empirically the effectiveness of the algorithm in both synthetic and benchmark problems.

Notation: Scalars are denoted by lower case letters, vectors by lower case bold face letters, and matrices by upper case bold face letters. For a positive integer k, $[k]$ denotes $\{1, \ldots, k\}$. For a vector $\boldsymbol{\theta} \in \mathbb{R}^d$, we denote its ℓ_2 norm by $\|\boldsymbol{\theta}\|_2 = \sqrt{\sum_{i=1}^d \theta_i^2}$ and its j-th coordinate by $[\boldsymbol{\theta}]_j$. For a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$, we denote its spectral norm, Frobenius norm, and (i,j)-th entry by $\|\mathbf{A}\|_2$, $\|\mathbf{A}\|_F$, and $[\mathbf{A}]_{i,j}$, respectively. We denote a sequence of vectors by $\{\boldsymbol{\theta}_j\}_{j=1}^t$, and similarly for matrices. For two sequences $\{a_n\}$ and $\{b_n\}$, we use $a_n = O(b_n)$ to denote that there exists some constant $C > 0$ such that $a_n \leq Cb_n$; similarly, $a_n = \Omega(b_n)$ means there exists some constant $C' > 0$ such that $a_n \geq C'b_n$. In addition, we use $\widetilde{O}(\cdot)$ to hide logarithmic factors. We say a random variable X is ν-sub-Gaussian if $\mathbb{E}\exp(\lambda(X - \mathbb{E}X)) \leq \exp(\lambda^2\nu^2/2)$ for any $\lambda > 0$.

2. Problem Setting

We consider the stochastic K-armed contextual bandit problem, where the total number of rounds T is known. At round $t \in [T]$, the agent observes the context consisting of K feature vectors: $\{\mathbf{x}_{t,a} \in \mathbb{R}^d \mid a \in [K]\}$. The agent selects an action a_t and receives a reward r_{t,a_t}. For brevity, we denote by $\{\mathbf{x}^i\}_{i=1}^{TK}$ the collection of $\{\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \ldots, \mathbf{x}_{T,K}\}$. Our goal is to maximize the following *pseudo regret* (or *regret* for short):

$$R_T = \mathbb{E}\bigg[\sum_{t=1}^{T}(r_{t,a_t^*} - r_{t,a_t})\bigg], \tag{2.1}$$

where $a_t^* = \mathrm{argmax}_{a\in[K]}\, \mathbb{E}[r_{t,a}]$ is the optimal action at round t that maximizes the expected reward.

This work makes the following assumption about reward generation: for any round t,

$$r_{t,a_t} = h(\mathbf{x}_{t,a_t}) + \xi_t, \tag{2.2}$$

where h is an unknown function satisfying $0 \leq h(\mathbf{x}) \leq 1$ for any $\mathbf{x}$, and ξ_t is ν-sub-Gaussian noise conditioned on $\mathbf{x}_{1,a_1}, \ldots, \mathbf{x}_{t-1,a_{t-1}}$ satisfying $\mathbb{E}\xi_t = 0$. The ν-sub-Gaussian assumption for ξ_t is standard in the stochastic bandit literature (e.g., Abbasi-Yadkori et al., 2011; Li et al., 2017), and is satisfied by, for example, any bounded noise. The bounded h assumption holds true when h belongs to linear functions, generalized linear functions, Gaussian processes, and kernel functions with bounded RKHS norm over a bounded domain, among others.

In order to learn the reward function h in (2.2), we propose to use a fully connected neural networks with depth $L \geq 2$:

$$f(\mathbf{x};\boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_L\sigma\Big(\mathbf{W}_{L-1}\sigma\big(\cdots\sigma(\mathbf{W}_1\mathbf{x})\big)\Big), \tag{2.3}$$

where $\sigma(x) = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function, $\mathbf{W}_1 \in \mathbb{R}^{m\times d}, \mathbf{W}_i \in \mathbb{R}^{m\times m}, 2 \leq i \leq L-1, \mathbf{W}_L \in \mathbb{R}^{m\times 1}$, and $\boldsymbol{\theta} = [\mathrm{vec}(\mathbf{W}_1)^\top, \ldots, \mathrm{vec}(\mathbf{W}_L)^\top]^\top \in \mathbb{R}^p$ with $p = m + md + m^2(L-1)$. Without loss of generality, we assume that the width of each hidden layer is the same (i.e., m) for convenience in analysis. We denote the gradient of the neural network function by $\mathbf{g}(\mathbf{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta}) \in \mathbb{R}^p$.

3. The NeuralUCB Algorithm

The key idea of NeuralUCB (Algorithm 1) is to use a neural network $f(\mathbf{x};\boldsymbol{\theta})$ to predict the reward of context $\mathbf{x}$, and upper confidence bounds computed from the network to guide exploration (Auer, 2002).

Initialization It initializes the network by randomly generating each entry of $\boldsymbol{\theta}$ from an appropriate Gaussian distribution: for $1 \leq l \leq L-1$, $\mathbf{W}_l$ is set to be $\begin{pmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{pmatrix}$, where each entry of $\mathbf{W}$ is generated independently from $N(0, 4/m)$; $\mathbf{W}_L$ is set to $(\mathbf{w}^\top, -\mathbf{w}^\top)$, where each entry of $\mathbf{w}$ is generated independently from $N(0, 2/m)$.

Learning At round t, Algorithm 1 observes the contexts for all actions, $\{\mathbf{x}_{t,a}\}_{a=1}^K$. First, it computes an upper confidence bound $U_{t,a}$ for each action a, based on $\mathbf{x}_{t,a}$, θ_{t-1} (the current neural network parameter), and a positive scaling factor γ_{t-1}. It then chooses action a_t with the largest $U_{t,a}$, and receives the corresponding reward r_{t,a_t}. At the end of round t, NeuralUCB updates $\boldsymbol{\theta}_t$ by applying Algorithm 2 to (approximately) minimize $L(\boldsymbol{\theta})$ using gradient descent, and updates γ_t. We choose gradient descent in Algorithm 2 for the simplicity of analysis, although the training method can be replaced by stochastic gradient descent with a more involved analysis (Allen-Zhu et al., 2019; Zou et al., 2019).

Algorithm 1 NeuralUCB

Input: Number of rounds T, regularization parameter λ, exploration parameter ν, confidence parameter δ, norm parameter S, step size η, number of gradient descent steps J, network width m, network depth L.
Initialization: Randomly initialize $\boldsymbol{\theta}_0$ as described in the text
Initialize $\mathbf{Z}_0 = \lambda \mathbf{I}$
for $t = 1, \ldots, T$ **do**
  Observe $\{\mathbf{x}_{t,a}\}_{a=1}^K$
  for $a = 1, \ldots, K$ **do**
    Compute $U_{t,a} = f(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1}) + \gamma_{t-1} \sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})^\top \mathbf{Z}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \boldsymbol{\theta}_{t-1})/m}$
    Let $a_t = \operatorname{argmax}_{a \in [K]} U_{t,a}$
  end for
  Play a_t and observe reward r_{t,a_t}
  Compute $\mathbf{Z}_t = \mathbf{Z}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1}) \mathbf{g}(\mathbf{x}_{t,a_t}; \boldsymbol{\theta}_{t-1})^\top / m$
  Let $\boldsymbol{\theta}_t = \text{TrainNN}(\lambda, \eta, J, m, \{\mathbf{x}_{i,a_i}\}_{i=1}^t, \{r_{i,a_i}\}_{i=1}^t, \boldsymbol{\theta}_0)$
  Compute

$$\gamma_t = \sqrt{1 + C_1 m^{-1/6} \sqrt{\log m} L^4 t^{7/6} \lambda^{-7/6}} \cdot \Bigg(\nu \sqrt{\log \frac{\det \mathbf{Z}_t}{\det \lambda \mathbf{I}} + C_2 m^{-1/6} \sqrt{\log m} L^4 t^{5/3} \lambda^{-1/6} - 2 \log \delta} + \sqrt{\lambda} S \Bigg)$$
$$+ (\lambda + C_3 t L) \Big[(1 - \eta m \lambda)^{J/2} \sqrt{t/\lambda} + m^{-1/6} \sqrt{\log m} L^{7/2} t^{5/3} \lambda^{-5/3} (1 + \sqrt{t/\lambda}) \Big].$$

end for

Algorithm 2 TrainNN$(\lambda, \eta, U, m, \{\mathbf{x}_{i,a_i}\}_{i=1}^t, \{r_{i,a_i}\}_{i=1}^t, \boldsymbol{\theta}^{(0)})$

Input: Regularization parameter λ, step size η, number of gradient descent steps U, network width m, contexts $\{\mathbf{x}_{i,a_i}\}_{i=1}^t$, rewards $\{r_{i,a_i}\}_{i=1}^t$, initial parameter $\boldsymbol{\theta}^{(0)}$.
Define $\mathcal{L}(\boldsymbol{\theta}) = \sum_{i=1}^t (f(\mathbf{x}_{i,a_i}; \boldsymbol{\theta}) - r_{i,a_i})^2/2 + m\lambda \|\boldsymbol{\theta} - \boldsymbol{\theta}^{(0)}\|_2^2/2$.
for $j = 0, \ldots, J-1$ **do**
  $\boldsymbol{\theta}^{(j+1)} = \boldsymbol{\theta}^{(j)} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}^{(j)})$
end for
Return $\boldsymbol{\theta}^{(J)}$.

Comparison with Existing Algorithms We compare NeuralUCB with other neural contextual bandit algorithms. Allesiardo et al. (2014) proposed NeuralBandit which consists of K neural networks. It uses a committee of networks to compute the score of each action and chooses an action with the ϵ-greedy strategy. In contrast, our NeuralUCB uses upper confidence bound-based exploration, which is more effective than ϵ-greedy. In addition, our algorithm only uses one neural network instead of K networks, thus can be computationally more efficient.

Lipton et al. (2018) used Thompson sampling on deep neural networks (through variational inference) in reinforcement learning; a variant is proposed by Azizzadenesheli et al. (2018) that works well on a set of Atari benchmarks. Riquelme et al. (2018) proposed NeuralLinear, which uses the first $L-1$ layers of a L-layer DNN to learn a representation, then applies Thompson sampling on the last layer to choose action. Zahavy & Mannor (2019) proposed a NeuralLinear with limited memory (NeuralLinearLM), which also uses the first $L-1$ layers of a L-layer DNN to learn a representation and applies Thompson sampling on the last layer. Instead of computing the exact mean and variance in Thompson sampling, NeuralLinearLM only computes their approximation. Unlike NeuralLinear and NeuralLinearLM, NeuralUCB uses the entire DNN to learn the representation and constructs the upper confidence bound based on the random feature mapping defined by the neural network gradient. Finally, Kveton et al. (2020) studied the use of reward perturbation for exploration in neural network-based bandit algorithms.

A Variant of NeuralUCB called NeuralUCB$_0$ is described in Appendix E. It can be viewed as a simplified version of NeuralUCB where only the first-order Taylor approximation of the neural network around the initialized parameter is updated through online ridge regression. In this sense, NeuralUCB$_0$ can be seen as KernelUCB (Valko et al., 2013) specialized to the Neural Tangent Kernel (Jacot et al., 2018), or LinUCB (Li et al., 2010) with Neural Tangent Random Features (Cao & Gu, 2019).

While this variant has a comparable regret bound as NeuralUCB, we expect the latter to be stronger in practice. Indeed, as shown by Allen-Zhu & Li (2019), the Neural Tangent Kernel does not seem to completely realize the representation power of neural networks in supervised learning. A similar phenomenon will be demonstrated for contextual bandit learning in Section 7.

4. Regret Analysis

This section analyzes the regret of NeuralUCB. Recall that $\{\mathbf{x}^i\}_{i=1}^{TK}$ is the collection of all $\{\mathbf{x}_{t,a}\}$. Our regret analysis is built upon the recently proposed neural tangent kernel matrix (Jacot et al., 2018):

Definition 4.1 (Jacot et al. (2018); Cao & Gu (2019))**.** Let $\{\mathbf{x}^i\}_{i=1}^{TK}$ be a set of contexts. Define

$$\widetilde{\mathbf{H}}_{i,j}^{(1)} = \mathbf{\Sigma}_{i,j}^{(1)} = \langle \mathbf{x}^i, \mathbf{x}^j \rangle, \qquad \mathbf{A}_{i,j}^{(l)} = \begin{pmatrix} \mathbf{\Sigma}_{i,i}^{(l)} & \mathbf{\Sigma}_{i,j}^{(l)} \\ \mathbf{\Sigma}_{i,j}^{(l)} & \mathbf{\Sigma}_{j,j}^{(l)} \end{pmatrix},$$
$$\mathbf{\Sigma}_{i,j}^{(l+1)} = 2\mathbb{E}_{(u,v)\sim N(\mathbf{0},\mathbf{A}_{i,j}^{(l)})}\left[\sigma(u)\sigma(v)\right],$$
$$\widetilde{\mathbf{H}}_{i,j}^{(l+1)} = 2\widetilde{\mathbf{H}}_{i,j}^{(l)}\mathbb{E}_{(u,v)\sim N(\mathbf{0},\mathbf{A}_{i,j}^{(l)})}\left[\sigma'(u)\sigma'(v)\right] + \mathbf{\Sigma}_{i,j}^{(l+1)}.$$

Then, $\mathbf{H} = (\widetilde{\mathbf{H}}^{(L)} + \mathbf{\Sigma}^{(L)})/2$ is called the *neural tangent kernel (NTK)* matrix on the context set.

In the above definition, the Gram matrix $\mathbf{H}$ of the NTK on the contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$ for L-layer neural networks is defined recursively from the input layer all the way to the output layer of the network. Interested readers are referred to Jacot et al. (2018) for more details about neural tangent kernels.

With Definition 4.1, we may state the following assumption on the contexts: $\{\mathbf{x}^i\}_{i=1}^{TK}$.

Assumption 4.2. $\mathbf{H} \succeq \lambda_0 \mathbf{I}$. Moreover, for any $1 \le i \le TK$, $\|\mathbf{x}^i\|_2 = 1$ and $[\mathbf{x}^i]_j = [\mathbf{x}^i]_{j+d/2}$.

The first part of the assumption says that the neural tangent kernel matrix is non-singular, a mild assumption commonly made in the related literature (Du et al., 2019a; Arora et al., 2019; Cao & Gu, 2019). It can be satisfied as long as *no* two contexts in $\{\mathbf{x}^i\}_{i=1}^{TK}$ are parallel. The second part is also mild and is just for convenience in analysis: for any context $\mathbf{x}$, $\|\mathbf{x}\|_2 = 1$, we can always construct a new context $\mathbf{x}' = [\mathbf{x}^\top, \mathbf{x}^\top]^\top/\sqrt{2}$ to satisfy Assumption 4.2. It can be verified that if $\boldsymbol{\theta}_0$ is initialized as in NeuralUCB, then $f(\mathbf{x}^i; \boldsymbol{\theta}_0) = 0$ for any $i \in [TK]$.

Next we define the effective dimension of the neural tangent kernel matrix.

Definition 4.3. The effective dimension $\widetilde{d}$ of the neural tangent kernel matrix on contexts $\{\mathbf{x}^i\}_{i=1}^{TK}$ is defined as

$$\widetilde{d} = \frac{\log\det(\mathbf{I} + \mathbf{H}/\lambda)}{\log(1 + TK/\lambda)}. \tag{4.1}$$

Remark 4.4. The notion of effective dimension was first introduced by Valko et al. (2013) for analyzing kernel contextual bandits, which was defined by the eigenvalues of any kernel matrix restricted to the given contexts. We adapt a similar but different definition of Yang & Wang (2019), which was used for the analysis of kernel-based Q-learning. Suppose the dimension of the reproducing kernel Hilbert space induced by the given kernel is $\widehat{d}$ and the feature mapping $\boldsymbol{\psi} : \mathbb{R}^d \to \mathbb{R}^{\widehat{d}}$ induced by the given kernel satisfies $\|\boldsymbol{\psi}(\mathbf{x})\|_2 \le 1$ for any $\mathbf{x} \in \mathbb{R}^d$. Then, it can be verified that $\widetilde{d} \le \widehat{d}$, as shown in Appendix A.1. Intuitively, $\widetilde{d}$ measures how quickly the eigenvalues of $\mathbf{H}$ diminish, and only depends on T logarithmically in several special cases (Valko et al., 2013).

Now we are ready to present the main result, which provides the regret bound R_T of Algorithm 1.

Theorem 4.5. Let $\widetilde{d}$ be the effective dimension, and $\mathbf{h} = [h(\mathbf{x}^i)]_{i=1}^{TK} \in \mathbb{R}^{TK}$. There exist constant $C_1, C_2 > 0$, such that for any $\delta \in (0,1)$, if

$$m \ge \mathrm{poly}(T, L, K, \lambda^{-1}, \lambda_0^{-1}, S^{-1}, \log(1/\delta)), \tag{4.2}$$
$$\eta = C_1(mTL + m\lambda)^{-1},$$

$\lambda \ge \max\{1, S^{-2}\}$, and $S \ge \sqrt{2\mathbf{h}^\top \mathbf{H}^{-1}\mathbf{h}}$, then with probability at least $1-\delta$, the regret of Algorithm 1 satisfies

$$\begin{aligned} R_T \le\ & 3\sqrt{T}\sqrt{\widetilde{d}\log(1+TK/\lambda)+2} \\ & \cdot \Bigg[\nu\sqrt{\widetilde{d}\log(1+TK/\lambda)+2-2\log\delta} \\ & + (\lambda + C_2TL)(1-\lambda/(TL))^{J/2}\sqrt{T/\lambda} \\ & + +2\sqrt{\lambda}S\Bigg] + 1. \end{aligned} \tag{4.3}$$

Remark 4.6. It is worth noting that, simply applying results for linear bandits to our algorithm would lead to a linear dependence of p or $\sqrt{p}$ in the regret. Such a bound is vacuous since in our setting p would be very large compared with the number of rounds T and the input context dimension d. In contrast, our regret bound only depends on $\widetilde{d}$, which can be much smaller than p.

Remark 4.7. Our regret bound (4.3) has a term $(\lambda + C_2TL)(1-\lambda/(TL))^{J/2}\sqrt{T/\lambda}$, which characterizes the optimization error of Algorithm 2 after J iterations. Setting

$$J = 2\log\frac{\lambda S}{\sqrt{T}(\lambda + C_2TL)}\frac{TL}{\lambda} = \widetilde{O}(TL/\lambda), \tag{4.4}$$

which is independent of m, we have $(\lambda + C_2TL)(1-\lambda/(TL))^{J/2}\sqrt{T/\lambda} \le \sqrt{\lambda}S$, so the optimization error is dominated by $\sqrt{\lambda}S$. Hence, the order of the regret bound is not affected by the error of optimization.

Remark 4.8. With ν and λ treated as constants, $S = \sqrt{2\mathbf{h}^\top\mathbf{H}^{-1}\mathbf{h}}$, and J given in (4.4), the regret bound (4.3) becomes $R_T = \widetilde{O}\Big(\sqrt{\widetilde{d}T}\sqrt{\max\{\widetilde{d}, \mathbf{h}^\top\mathbf{H}^{-1}\mathbf{h}\}}\Big)$. Specifically, if h belongs to the RKHS $\mathcal{H}$ induced by the neural

tangent kernel with bounded RKHS norm $\|h\|_{\mathcal{H}}$, we have $\|h\|_{\mathcal{H}} \geq \sqrt{\mathbf{h}^\top \mathbf{H}^{-1}\mathbf{h}}$; see Appendix A.2 for more details. Thus our regret bound can be further written as

$$R_T = \widetilde{O}\Big(\sqrt{\widetilde{d}T}\sqrt{\max\{\widetilde{d}, \|h\|_{\mathcal{H}}\}}\Big). \tag{4.5}$$

The high-probability result in Theorem 4.5 can be used to obtain a bound on the expected regret.

Corollary 4.9. Under the same conditions in Theorem 4.5, there exists a positive constant C such that

$$\begin{aligned}
&\mathbb{E}[R_T] \\
&\leq 2 + 3\sqrt{T}\sqrt{\widetilde{d}\log(1+TK/\lambda)+2} \\
&\quad \cdot \Bigg[\nu\sqrt{\widetilde{d}\log(1+TK/\lambda)+2+2\log T} \\
&\quad + 2\sqrt{\lambda}S + (\lambda + CTL)(1-\lambda/(TL))^{J/2}\sqrt{T/\lambda}\Bigg].
\end{aligned}$$

5. Proof of Main Result

This section outlines the proof of Theorem 4.5, which has to deal with the following technical challenges:

- We do not make parametric assumptions on the reward function as some previous work (Filippi et al., 2010; Chu et al., 2011; Abbasi-Yadkori et al., 2011).
- To avoid strong parametric assumptions, we use overparameterized neural networks, which implies m (and thus p) is very large. Therefore, we need to make sure the regret bound is independent of m.
- Unlike the *static* feature mapping used in kernel bandit algorithms (Valko et al., 2013), NeuralUCB uses a neural network $f(\mathbf{x};\boldsymbol{\theta}_t)$ and its gradient $\mathbf{g}(\mathbf{x};\boldsymbol{\theta}_t)$ as a *dynamic* feature mapping depending on $\boldsymbol{\theta}_t$. This difference makes the analysis of NeuralUCB more difficult.

These challenges are addressed by the following technical lemmas, whose proofs are gathered in the appendix.

Lemma 5.1. There exists a positive constant $\bar{C}$ such that for any $\delta \in (0,1)$, if $m \geq \bar{C}T^4K^4L^6\log(T^2K^2L/\delta)/\lambda_0^4$, then with probability at least $1-\delta$, there exists a $\boldsymbol{\theta}^* \in \mathbb{R}^p$ such that

$$\begin{aligned}
&h(\mathbf{x}^i) = \langle \mathbf{g}(\mathbf{x}^i;\boldsymbol{\theta}_0), \boldsymbol{\theta}^* - \boldsymbol{\theta}_0\rangle, \\
&\sqrt{m}\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_0\|_2 \leq \sqrt{2\mathbf{h}^\top\mathbf{H}^{-1}\mathbf{h}},
\end{aligned} \tag{5.1}$$

for all $i \in [TK]$.

Lemma 5.1 suggests that with high probability, the reward function restricted to $\{\mathbf{x}^i\}_{i=1}^{TK}$ can be regarded as a linear function of $\mathbf{g}(\mathbf{x}^i;\boldsymbol{\theta}_0)$ parameterized by $\boldsymbol{\theta}^* - \boldsymbol{\theta}_0$, where $\boldsymbol{\theta}^*$ lies in a ball centered at $\boldsymbol{\theta}_0$. Note that here $\boldsymbol{\theta}^*$ is not a ground truth parameter for the reward function. Instead, it is introduced only for the sake of analysis. Equipped with Lemma 5.1, we can utilize existing results on linear bandits (Abbasi-Yadkori et al., 2011) to show that with high probability, $\boldsymbol{\theta}^*$ lies in the sequence of confidence sets.

Lemma 5.2. There exist positive constants $\bar{C}_1$ and $\bar{C}_2$ such that for any $\delta \in (0,1)$, if $\eta \leq \bar{C}_1(TmL+m\lambda)^{-1}$ and

$$\begin{aligned}
m \geq \bar{C}_2 \max\big\{&T^7\lambda^{-7}L^{21}(\log m)^3, \\
&\lambda^{-1/2}L^{-3/2}(\log(TKL^2/\delta))^{3/2}\big\},
\end{aligned}$$

then with probability at least $1-\delta$, we have $\|\boldsymbol{\theta}_t - \boldsymbol{\theta}_0\|_2 \leq 2\sqrt{t/(m\lambda)}$ and $\|\boldsymbol{\theta}^* - \boldsymbol{\theta}_t\|_{\mathbf{Z}_t} \leq \gamma_t/\sqrt{m}$ for all $t \in [T]$, where γ_t is defined in Algorithm 1.

Lemma 5.3. Let $a_t^* = \mathrm{argmax}_{a\in[K]} h(\mathbf{x}_{t,a})$. There exists a positive constant $\bar{C}$ such that for any $\delta \in (0,1)$, if η and m satisfy the same conditions as in Lemma 5.2, then with probability at least $1-\delta$, we have

$$\begin{aligned}
&h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t}) \\
&\leq 2\gamma_{t-1}\min\Big\{\|\mathbf{g}(\mathbf{x}_{t,a_t};\boldsymbol{\theta}_{t-1})/\sqrt{m}\|_{\mathbf{Z}_{t-1}^{-1}}, 1\Big\} \\
&\quad + \bar{C}\big(Sm^{-1/6}\sqrt{\log m}T^{7/6}\lambda^{-1/6}L^{7/2} \\
&\quad + m^{-1/6}\sqrt{\log m}T^{5/3}\lambda^{-2/3}L^3\big).
\end{aligned}$$

Lemma 5.3 gives an upper bound for $h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t})$, which can be used to bound the regret R_T. It is worth noting that γ_t has a term $\log\det\mathbf{Z}_t$. A trivial upper bound of $\log\det\mathbf{Z}_t$ would result in a quadratic dependence on the network width m, since the dimension of $\mathbf{Z}_t$ is $p = md + m^2(L-2) + m$. Instead, we use the next lemma to establish an m-independent upper bound. The dependence on $\widetilde{d}$ is similar to Valko et al. (2013, Lemma 4), but the proof is different as our notion of effective dimension is different.

Lemma 5.4. There exist positive constants $\{\bar{C}_i\}_{i=1}^3$ such that for any $\delta \in (0,1)$, if $m \geq \bar{C}_1\max\{T^7\lambda^{-7}L^{21}(\log m)^3, T^6K^6L^6(\log(TKL^2/\delta))^{3/2}\}$ and $\eta \leq \bar{C}_2(TmL+m\lambda)^{-1}$, then with probability at least $1-\delta$, we have

$$\begin{aligned}
&\sqrt{\sum_{t=1}^T \gamma_{t-1}^2 \min\Big\{\|\mathbf{g}(\mathbf{x}_{t,a_t};\boldsymbol{\theta}_{t-1})/\sqrt{m}\|_{\mathbf{Z}_{t-1}^{-1}}^2, 1\Big\}} \\
&\leq \sqrt{\widetilde{d}\log(1+TK/\lambda)+\Gamma_1} \\
&\quad \Bigg[\Gamma_2\Big(\nu\sqrt{\widetilde{d}\log(1+TK/\lambda)+\Gamma_1-2\log\delta}+\sqrt{\lambda}S\Big) \\
&\quad + (\lambda + \bar{C}_3tL)\Big[(1-\eta m\lambda)^{J/2}\sqrt{T/\lambda}+\Gamma_3(1+\sqrt{T/\lambda})\Big]\Bigg],
\end{aligned}$$

where

$$\Gamma_1 = 1 + \bar{C}_3 m^{-1/6}\sqrt{\log m} L^4 T^{5/3}\lambda^{-1/6},$$
$$\Gamma_2 = \sqrt{1 + \bar{C}_3 m^{-1/6}\sqrt{\log m} L^4 T^{7/6}\lambda^{-7/6}},$$
$$\Gamma_3 = m^{-1/6}\sqrt{\log m} L^{7/2} T^{5/3}\lambda^{-5/3}.$$

We are now ready to prove the main result.

Proof of Theorem 4.5. Lemma 5.3 implies that the total regret R_T can be bounded as follows with a constant $C_1 > 0$:

$$\begin{aligned}
R_T &= \sum_{t=1}^{T}\big[h(\mathbf{x}_{t,a_t^*}) - h(\mathbf{x}_{t,a_t})\big] \\
&\le 2\sum_{t=1}^{T}\gamma_{t-1}\min\Big\{\|\mathbf{g}(\mathbf{x}_{t,a_t};\boldsymbol{\theta}_{t-1})/\sqrt{m}\|_{\mathbf{Z}_{t-1}^{-1}}, 1\Big\} \\
&\quad + C_1\big(Sm^{-1/6}\sqrt{\log m}T^{13/6}\lambda^{-1/6}L^{7/2} \\
&\quad + m^{-1/6}\sqrt{\log m}T^{8/3}\lambda^{-2/3}L^3\big).
\end{aligned}$$

It can be further bounded as follows:

$$\begin{aligned}
R_T &\le 2\sqrt{T\sum_{t=1}^{T}\gamma_{t-1}^2\min\Big\{\|\mathbf{g}(\mathbf{x}_{t,a_t};\boldsymbol{\theta}_{t-1})/\sqrt{m}\|_{\mathbf{Z}_{t-1}^{-1}}^2, 1\Big\}} \\
&\quad + C_1\big(Sm^{-1/6}\sqrt{\log m}T^{13/6}\lambda^{-1/6}L^{7/2} \\
&\quad + m^{-1/6}\sqrt{\log m}T^{8/3}\lambda^{-2/3}L^3\big) \\
&\le 2\sqrt{T}\cdot\sqrt{\widetilde{d}\log(1+TK/\lambda) + \Gamma_1} \\
&\quad \Bigg[\Gamma_2\bigg(\nu\sqrt{\widetilde{d}\log(1+TK/\lambda) + \Gamma_1 - 2\log\delta} + \sqrt{\lambda}S\bigg) \\
&\quad + (\lambda + C_2 TL)\Big[(1-\eta m\lambda)^{J/2}\sqrt{T/\lambda} \\
&\quad + \Gamma_3(1+\sqrt{T/\lambda})\Big]\Bigg] \\
&\quad + C_1\big(Sm^{-1/6}\sqrt{\log m}T^{13/6}\lambda^{-1/6}L^{7/2} \\
&\quad + m^{-1/6}\sqrt{\log m}T^{8/3}\lambda^{-2/3}L^3\big) \\
&\le 3\sqrt{T}\sqrt{\widetilde{d}\log(1+TK/\lambda) + 2} \\
&\quad \cdot\Bigg[\nu\sqrt{\widetilde{d}\log(1+TK/\lambda) + 2 - 2\log\delta} \\
&\quad + (\lambda + C_3 TL)(1-\eta m\lambda)^{J/2}\sqrt{T/\lambda} \\
&\quad + 2\sqrt{\lambda}S\Bigg] + 1,
\end{aligned}$$

where C_1, C_2, C_3 are positive constants, the first inequality is due to Cauchy-Schwarz inequality, the second inequality due to Lemma 5.4, and the third inequality holds for sufficiently large m. This completes our proof. □

6. Related Work

Contextual Bandits There is a line of extensive work on linear bandits (e.g., Abe et al., 2003; Auer, 2002; Abe et al., 2003; Dani et al., 2008; Rusmevichientong & Tsitsiklis, 2010; Li et al., 2010; Chu et al., 2011; Abbasi-Yadkori et al., 2011). Many of these algorithms are based on the idea of upper confidence bounds, and are shown to achieve near-optimal regret bounds. Our algorithm is also based on UCB exploration, and the regret bound reduces to that of Abbasi-Yadkori et al. (2011) in the linear case.

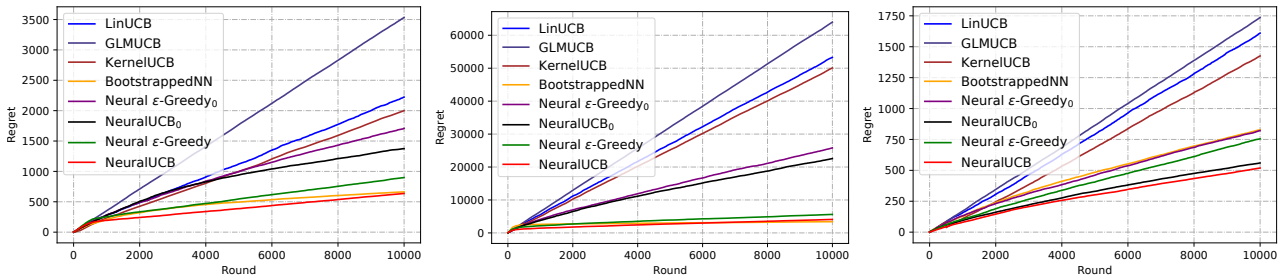
To deal with nonlinearity, a few authors have considered generalized linear bandits (Filippi et al., 2010; Li et al., 2017; Jun et al., 2017), where the reward function is a composition of a linear function and a (nonlinear) link function. Such models are special cases of what we study in this work.

More general nonlinear bandits without making strong modeling assumptions have also be considered. One line of work is the family of expert learning algorithms (Auer et al., 2002; Beygelzimer et al., 2011) that typically have a time complexity linear in the number of experts (which in many cases can be exponential in the number of parameters).

A second approach is to reduce a bandit problem to supervised learning, such as the epoch-greedy algorithm (Langford & Zhang, 2008) that has a non-optimal $O(T^{2/3})$ regret. Later, Agarwal et al. (2014) develop an algorithm that enjoys a near-optimal regret, but relies on an oracle, whose implementation still requires proper modeling assumptions.

A third approach uses nonparametric modeling, such as perceptrons (Kakade et al., 2008), random forests (Féraud et al., 2016), Gaussian processes and kernels (Kleinberg et al., 2008; Srinivas et al., 2010; Krause & Ong, 2011; Bubeck et al., 2011). The most relevant is by Valko et al. (2013), who assumed that the reward function lies in an RKHS with bounded RKHS norm and developed a UCB-based algorithm. They also proved an $\widetilde{O}(\sqrt{\widetilde{d}T})$ regret, where $\widetilde{d}$ is a form of effective dimension similar to ours. Compared with these interesting works, our neural network-based algorithm avoids the need to carefully choose a good kernel or metric, and can be computationally more efficient in large-scale problems. Recently, Foster & Rakhlin (2020) proposed contextual bandit algorithms with regression oracles which achieve a dimension-independent $O(T^{3/4})$ regret. Compared with Foster & Rakhlin (2020), NeuralUCB achieves a dimension-dependent $\widetilde{O}(\widetilde{d}\sqrt{T})$ regret with a better dependence on the time horizon.

Neural Networks Substantial progress has been made to understand the expressive power of DNNs, in connection to the network depth (Telgarsky, 2015; 2016; Liang & Srikant, 2016; Yarotsky, 2017; 2018; Hanin, 2017), as well as network width (Lu et al., 2017; Hanin & Sellke, 2017).



(a) $h_1(\mathbf{x}) = 10(\mathbf{x}^\top \mathbf{a})^2$ (b) $h_2(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x}$ (c) $h_3(\mathbf{x}) = \cos(3\mathbf{x}^\top \mathbf{a})$

Figure 1. Comparison of NeuralUCB and baseline algorithms on synthetic datasets.

The present paper on neural contextual bandit algorithms is inspired by these theoretical justifications and empirical evidence in the literature.

Our regret analysis for NeuralUCB makes use of recent advances in optimizing a DNN. A series of works show that (stochastic) gradient descent can find global minima of the training loss (Li & Liang, 2018; Du et al., 2019b; Allen-Zhu et al., 2019; Du et al., 2019a; Zou et al., 2019; Zou & Gu, 2019). For the generalization of DNNs, a number of authors (Daniely, 2017; Cao & Gu, 2019; 2020; Arora et al., 2019; Chen et al., 2019) show that by using (stochastic) gradient descent, the parameters of a DNN are located in a particular regime and the generalization bound of DNNs can be characterized by the best function in the corresponding neural tangent kernel space (Jacot et al., 2018).

7. Experiments

In this section, we evaluate NeuralUCB empirically and compare it with seven representative baselines: (1) LinUCB, which is also based on UCB but adopts a linear representation; (2) GLMUCB (Filippi et al., 2010), which applies a nonlinear link function over a linear function; (3) KernelUCB (Valko et al., 2013), a kernelised UCB algorithm which makes use of a predefined kernel function; (4) BootstrappedNN (Efron, 1982; Riquelme et al., 2018), which simultaneously trains a set of neural networks using bootstrapped samples and at every round chooses an action based on the prediction of a randomly picked model; (5) Neural ϵ-Greedy, which replaces the UCB-based exploration in Algorithm 1 by ϵ-greedy; (6) NeuralUCB$_0$, as described in Section 3; and (7) Neural ϵ-Greedy$_0$, same as NeuralUCB$_0$ but with ϵ-greedy exploration. We use the cumulative regret as the performance metric.

7.1. Synthetic Datasets

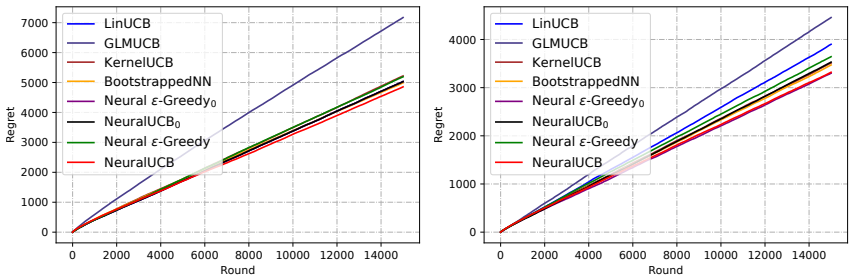
In the first set of experiments, we use contextual bandits with context dimension $d = 20$ and $K = 4$ actions. The number of rounds $T = 10\ 000$. The contextual vectors $\{\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{T,K}\}$ are chosen uniformly at random from the unit ball. The reward function h is one of the following:

$$h_1(\mathbf{x}) = 10(\mathbf{x}^\top \mathbf{a})^2,$$
$$h_2(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x},$$
$$h_3(\mathbf{x}) = \cos(3\mathbf{x}^\top \mathbf{a}),$$

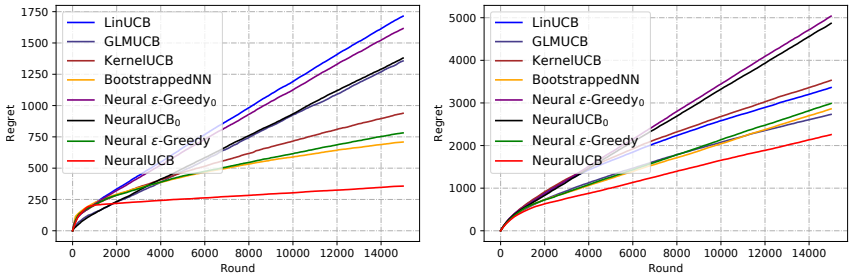
where each entry of $\mathbf{A} \in \mathbb{R}^{d\times d}$ is randomly generated from $N(0, 1)$, $\mathbf{a}$ is randomly generated from uniform distribution over unit ball. For each $h_i(\cdot)$, the reward is generated by $r_{t,a} = h_i(\mathbf{x}_{t,a}) + \xi_t$, where $\xi_t \sim N(0, 1)$.

Following Li et al. (2010), we implement LinUCB using a constant α (for the variance term in the UCB). We do a grid search for α over $\{0.01, 0.1, 1, 10\}$. For GLMUCB, we use the sigmoid function as the link function and adapt the online Newton step method to accelerate the computation (Zhang et al., 2016; Jun et al., 2017). We do grid searches over $\{0.1, 1, 10\}$ for regularization parameter, $\{1, 10, 100\}$ for step size, $\{0.01, 0.1, 1\}$ for exploration parameter. For KernelUCB, we use the radial basis function (RBF) kernel with parameter σ, and set the regularization parameter to 1. Grid searches over $\{0.1, 1, 10\}$ for σ and $\{0.01, 0.1, 1, 10\}$ for the exploration parameter are done. To accelerate the calculation, we stop adding contexts to KernelUCB after 1000 rounds, following the same setting for Gaussian Process in Riquelme et al. (2018). For all five neural algorithms, we choose a two-layer neural network $f(\mathbf{x}; \boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x})$ with network width $m = 20$, where $\boldsymbol{\theta} = [\text{vec}(\mathbf{W}_1)^\top, \text{vec}(\mathbf{W}_2)^\top] \in \mathbb{R}^p$ and $p = md + m = 420$.[1] Moreover, we set $\gamma_t = \gamma$ in NeuralUCB, and do a grid search over $\{0.01, 0.1, 1, 10\}$. For NeuralUCB$_0$, we do grid searches for ν over $\{0.1, 1, 10\}$, for λ over $\{0.1, 1, 10\}$, for δ over $\{0.01, 0.1, 1\}$, for S over $\{0.01, 0.1, 1, 10\}$. For Neural ϵ-Greedy and Neural ϵ-Greedy$_0$, we do a grid search for ϵ over $\{0.001, 0.01, 0.1, 0.2\}$. For BootstrappedNN, we follow Riquelme et al. (2018) to set the number of models to be 10 and the transition probability to be 0.8. To accelerate

[1] Note that the bound on the required network width m is likely not tight. Therefore, in experiments we choose m to be relatively large, but not as large as theory suggests.



(a) `covertype` (b) `magic`



(c) `statlog` (d) `mnist`

Figure 2. Comparison of NeuralUCB and baseline algorithms on real-world datasets.

Table 1. Dataset statistics

DATASET	COVER-TYPE	MAGIC	STATLOG	MNIST
FEATURE DIMENSION	54	10	8	784
NUMBER OF CLASSES	7	2	7	10
NUMBER OF INSTANCES	581012	19020	58000	60000

the training process, for BootstrappedNN, NeuralUCB and Neural ϵ-Greedy, we update the parameter $\boldsymbol{\theta}_t$ by TrainNN every 50 rounds. We use stochastic gradient descent with batch size 50, $J = t$ at round t, and do a grid search for step size η over $\{0.001, 0.01, 0.1\}$. For all grid-searched parameters, we choose the best of them for the comparison. All experiments are repeated 10 times, and the averaged results reported for comparison.

7.2. Real-world Datasets

We evaluate our algorithms on real-world datasets from the UCI Machine Learning Repository (Dua & Graff, 2017): `covertype`, `magic`, and `statlog`. We also evaluate our algorithms on `mnist` dataset (LeCun et al., 1998). These are all K-class classification datasets (Table 1), and are converted into K-armed contextual bandits (Beygelzimer & Langford, 2009). The number of rounds is set as $T = 15000$. Following Riquelme et al. (2018), we create contextual bandit problems based on the prediction accuracy. In detail, to transform a classification problem with k-classes into a bandit problem, we adapts the disjoint model (Li et al., 2010) which transforms each contextual vector $\mathbf{x} \in \mathbb{R}^d$ into k vectors $\mathbf{x}^{(1)} = (\mathbf{x}, \mathbf{0}, \ldots, \mathbf{0}), \ldots, \mathbf{x}^{(k)} = (\mathbf{0}, \ldots, \mathbf{0}, \mathbf{x}) \in \mathbb{R}^{dk}$. The agent received regret 0 if he classifies the context correctly, and 1 otherwise. For all the algorithms, We reshuffle the order of contexts and repeat the experiment for 10 runs. Averaged results are reported for comparison.

For LinUCB, GLMUCB and KernelUCB, we tune their parameters as Section 7.1 suggests. For BootstrappedNN, NeuralUCB, NeuralUCB$_0$, Neural ϵ-Greedy and Neural ϵ-Greedy$_0$, we choose a two-layer neural network with width $m = 100$. For NeuralUCB and NeuralUCB$_0$, since it is computationally expensive to store and compute a whole matrix $\mathbf{Z}_t$, we use a diagonal matrix which consists of the diagonal elements of $\mathbf{Z}_t$ to approximate $\mathbf{Z}_t$. To accelerate the training process, for BootstrappedNN, NeuralUCB and Neural ϵ-Greedy, we update the parameter $\boldsymbol{\theta}_t$ by TrainNN every 100 rounds starting from round 2000. We do grid searches for λ over $\{10^{-i}\}, i = 1, 2, 3, 4$, for η over $\{2 \times 10^{-i}, 5 \times 10^{-i}\}, i = 1, 2, 3, 4$. We set $J = 1000$ and use stochastic gradient descent with batch size 500 to train the networks. For the rest of parameters, we tune them as those in Section 7.1 and choose the best of them for comparison.

7.3. Results

Figures 1 and 2 show the cumulative regret of all algorithms. First, due to the nonlinearity of reward functions h, LinUCB fails to learn them for nearly all tasks. GLMUCB is only able to learn the true reward functions for certain tasks due to its simple link function. In contrast, thanks to the neural network representation and efficient exploration, NeuralUCB achieves a substantially lower regret. The performance of Neural ϵ-Greedy is between the two. This suggests that while Neural ϵ-Greedy can capture the nonlinearity of the underlying reward function, ϵ-Greedy based exploration is not as effective as UCB based exploration. This confirms the effectiveness of NeuralUCB for contextual bandit problems with nonlinear reward functions. Second, it is worth noting that NeuralUCB and Neural ϵ-Greedy outperform NeuralUCB$_0$ and Neural ϵ-Greedy$_0$. This suggests that using deep neural networks to predict the reward function is better than using a fixed feature mapping associated with the Neural Tangent Kernel, which mirrors similar findings in supervised learning (Allen-Zhu & Li, 2019). Furthermore, we can see that KernelUCB is not as good as NeuralUCB, which suggests the limitation of simple kernels like RBF compared to flexible neural networks. What's more, BootstrappedNN can be competitive, approaching the performance of NeuralUCB in some datasets. However, it requires to maintain and train multiple neural networks, so is computationally more expensive than our approach, especially in large-scale problems.

8. Conclusion

In this paper, we proposed NeuralUCB, a new algorithm for stochastic contextual bandits based on neural networks and upper confidence bounds. Building on recent advances in optimization and generalization of deep neural networks, we showed that for an arbitrary bounded reward function, our algorithm achieves an $\widetilde{O}(\widetilde{d}\sqrt{T})$ regret bound. Promising empirical results on both synthetic and real-world data corroborated our theoretical findings, and suggested the potential of the algorithm in practice.

We conclude the paper with a suggested direction for future research. Given the focus on UCB exploration in this work, a natural open question is provably efficient exploration based on randomized strategies, when DNNs are used. These methods are effective in practice, but existing regret analyses are mostly for shallow (i.e., linear or generalized linear) models (Chapelle & Li, 2011; Agrawal & Goyal, 2013; Russo et al., 2018; Kveton et al., 2020). Extending them to DNNs will be interesting. Meanwhile, our current analysis of NeuralUCB is based on the NTK theory. While NTK facilitates the analysis, it has its own limitations, and we will leave the analysis of NeuralUCB beyond NTK as future work.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful comments. This research was sponsored in part by the National Science Foundation IIS-1904183 and IIS-1906169. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pp. 2312–2320, 2011.

Abe, N., Biermann, A. W., and Long, P. M. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37(4):263–293, 2003.

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R. E. Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 1638–1646, 2014.

Agrawal, S. and Goyal, N. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pp. 127–135, 2013.

Allen-Zhu, Z. and Li, Y. What can ResNet learn efficiently, going beyond kernels? In *Advances in Neural Information Processing Systems*, 2019.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pp. 242–252, 2019.

Allesiardo, R., Féraud, R., and Bouneffouf, D. A neural networks committee for the contextual bandit problem. In *International Conference on Neural Information Processing*, pp. 374–381. Springer, 2014.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, 2019.

Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

Azizzadenesheli, K., Brunskill, E., and Anandkumar, A. Efficient exploration through Bayesian deep Q-networks.

In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.

Beygelzimer, A. and Langford, J. The offset tree for learning with partial labels. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 129–138, 2009.

Beygelzimer, A., Langford, J., Li, L., Reyzin, L., and Schapire, R. E. Contextual bandit algorithms with supervised learning guarantees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 19–26, 2011.

Bubeck, S. and Cesa-Bianchi, N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. X-armed bandits. *Journal of Machine Learning Research*, 12(May):1655–1695, 2011.

Cao, Y. and Gu, Q. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Cao, Y. and Gu, Q. Generalization error bounds of gradient descent for learning over-parameterized deep relu networks. In *the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Chapelle, O. and Li, L. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pp. 2249–2257, 2011.

Chen, Z., Cao, Y., Zou, D., and Gu, Q. How much over-parameterization is sufficient to learn deep relu networks? *arXiv preprint arXiv:1911.12360*, 2019.

Chu, W., Li, L., Reyzin, L., and Schapire, R. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 208–214, 2011.

Dani, V., Hayes, T. P., and Kakade, S. M. Stochastic linear optimization under bandit feedback. 2008.

Daniely, A. SGD learns the conjugate kernel class of the network. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2017.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pp. 1675–1685, 2019a.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=S1eK3i09YQ.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Efron, B. *The jackknife, the bootstrap, and other resampling plans*, volume 38. Siam, 1982.

Féraud, R., Allesiardo, R., Urvoy, T., and Clérot, F. Random forest for the contextual bandit problem. In *Artificial Intelligence and Statistics*, pp. 93–101, 2016.

Filippi, S., Cappe, O., Garivier, A., and Szepesvári, C. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2010.

Foster, D. J. and Rakhlin, A. Beyond ucb: Optimal and efficient contextual bandits with regression oracles. *arXiv preprint arXiv:2002.04926*, 2020.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Hanin, B. Universal function approximation by deep neural nets with bounded width and ReLU activations. *arXiv preprint arXiv:1708.02691*, 2017.

Hanin, B. and Sellke, M. Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Jun, K.-S., Bhargava, A., Nowak, R. D., and Willett, R. Scalable generalized linear bandits: Online computation and hashing. In *Advances in Neural Information Processing Systems 30 (NIPS)*, pp. 99–109, 2017.

Kakade, S. M., Shalev-Shwartz, S., and Tewari, A. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th international conference on Machine learning*, pp. 440–447, 2008.

Kleinberg, R., Slivkins, A., and Upfal, E. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 681–690. ACM, 2008.

Krause, A. and Ong, C. S. Contextual Gaussian process bandit optimization. In *Advances in neural information processing systems*, pp. 2447–2455, 2011.

Kveton, B., Zaheer, M., Szepesvári, C., Li, L., Ghavamzadeh, M., and Boutilier, C. Randomized exploration in generalized linear bandits. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*, 2020.

Langford, J. and Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pp. 1096–1103, 2008.

Lattimore, T. and Szepesvári, C. *Bandit Algorithms*. Cambridge University Press, 2019. In press.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670. ACM, 2010.

Li, L., Lu, Y., and Zhou, D. Provably optimal algorithms for generalized linear contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2071–2080. JMLR. org, 2017.

Li, Y. and Liang, Y. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pp. 8157–8166, 2018.

Liang, S. and Srikant, R. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*, 2016.

Lipton, Z., Li, X., Gao, J., Li, L., Ahmed, F., and Deng, L. BBQ-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pp. 6231–6239, 2017.

Riquelme, C., Tucker, G., and Snoek, J. Deep Bayesian bandits showdown. In *International Conference on Learning Representations*, 2018.

Rusmevichientong, P. and Tsitsiklis, J. N. Linearly parameterized bandits. *Mathematics of Operations Research*, 35 (2):395–411, 2010.

Russo, D., Roy, B. V., Kazerouni, A., Osband, I., and Wen, Z. A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 1015–1022. Omnipress, 2010.

Telgarsky, M. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.

Telgarsky, M. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*, 2016.

Valko, M., Korda, N., Munos, R., Flaounas, I., and Cristianini, N. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

Yang, L. F. and Wang, M. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. *arXiv preprint arXiv:1905.10389*, 2019.

Yarotsky, D. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

Yarotsky, D. Optimal approximation of continuous functions by very deep ReLU networks. *arXiv preprint arXiv:1802.03620*, 2018.

Zahavy, T. and Mannor, S. Deep neural linear bandits: Overcoming catastrophic forgetting through likelihood matching. *arXiv preprint arXiv:1901.08612*, 2019.

Zhang, L., Yang, T., Jin, R., Xiao, Y., and Zhou, Z.-H. Online stochastic linear optimization under one-bit feedback. In *International Conference on Machine Learning*, pp. 392–401, 2016.

Zou, D. and Gu, Q. An improved analysis of training over-parameterized deep neural networks. In *Advances in Neural Information Processing Systems*, 2019.

Zou, D., Cao, Y., Zhou, D., and Gu, Q. Stochastic gradient descent optimizes over-parameterized deep ReLU networks. *Machine Learning*, 2019.