Neural Contextual Bandits with Deep Representation and Shallow Exploration

Pan Xu* and Zheng Wen† and Handong Zhao‡ and Quanquan Gu§

Abstract

We study a general class of contextual bandits, where each context-action pair is associated with a raw feature vector, but the reward generating function is unknown. We propose a novel learning algorithm that transforms the raw feature vector using the last hidden layer of a deep ReLU neural network (deep representation learning), and uses an upper confidence bound (UCB) approach to explore in the last linear layer (shallow exploration). We prove that under standard assumptions, our proposed algorithm achieves $\widetilde{O}(\sqrt{T})$ finite-time regret, where T is the learning time horizon. Compared with existing neural contextual bandit algorithms, our approach is computationally much more efficient since it only needs to explore in the last layer of the deep neural network.

1 Introduction

Multi-armed bandits (MAB) (Auer et al., 2002; Audibert et al., 2009; Lattimore and Szepesvári, 2020) are a class of online decision-making problems where an agent needs to learn to maximize its expected cumulative reward while repeatedly interacting with a partially known environment. Based on a bandit algorithm (also called a strategy or policy), in each round, the agent adaptively chooses an arm, and then observes and receives a reward associated with that arm. Since only the reward of the chosen arm will be observed (bandit information feedback), a good bandit algorithm has to deal with the exploration-exploitation dilemma: trade-off between pulling the best arm based on existing knowledge/history data (exploitation) and trying the arms that have not been fully explored (exploration).

In many real-world applications, the agent will also be able to access detailed contexts associated with the arms. For example, when a company wants to choose an advertisement to present to a user, the recommendation will be much more accurate if the company takes into consideration the contents, specifications, and other features of the advertisements in the arm set as well as the profile of the user. To encode the contextual information, contextual bandit models and algorithms have been developed, and widely studied both in theory and in practice (Dani et al., 2008; Rusmevichientong

*Department of Computer Science, University of California, Los Angeles, Los Angeles, CA 90095; e-mail: panxu@cs.ucla.edu

†DeepMind, Mountain View, CA 94043; e-mail: zhengwen@google.com

‡Adobe Research, San Jose, CA 95110; e-mail: hazhao@adobe.com

§Department of Computer Science, University of California, Los Angeles, Los Angeles, CA 90095; e-mail: qgu@cs.ucla.edu

and Tsitsiklis, 2010; Li et al., 2010; Chu et al., 2011; Abbasi-Yadkori et al., 2011). Most existing contextual bandit algorithms assume that the expected reward of an arm at a context is a linear function in a known context-action feature vector, which leads to many useful algorithms such as LinUCB (Chu et al., 2011), OFUL (Abbasi-Yadkori et al., 2011), etc. The representation power of the linear model can be limited in applications such as marketing, social networking, clinical studies, etc., where the rewards are usually counts or binary variables. The linear contextual bandit problem has also been extended to richer classes of parametric bandits such as the generalized linear bandits (Filippi et al., 2010; Li et al., 2017) and kernelised bandits (Valko et al., 2013; Chowdhury and Gopalan, 2017).

With the resurgence of deep neural networks and their phenomenal performances in many machine learning tasks (LeCun et al., 2015; Goodfellow et al., 2016), there has emerged a line of work that employs deep neural networks to increase the representation power of contextual bandit algorithms. Zhou et al. (2020) developed the NeuralUCB algorithm, which can be viewed as a direct extension of linear contextual bandits (Abbasi-Yadkori et al., 2011), where they use the output of a deep neural network with the feature vector as input to approximate the reward. Zhang et al. (2020) adapted neural networks in Thompson Sampling (Thompson, 1933; Chapelle and Li, 2011) for both exploration and exploitation and proposed the NeuralTS algorithm. For a fixed time horizon T, it has been proved that both NeuralUCB and NeuralTS achieve a $O(\widetilde{d}\sqrt{T})$ regret bound, where $\widetilde{d}$ is the effective dimension of a neural tangent kernel matrix which can potentially scale with $O(TK)$ for K-armed bandits. This high complexity is mainly due to their exploration over the entire neural network parameter space. A more realistic and efficient way of using deep neural networks in contextual bandits may be to just explore different arms using the last layer as the exploration parameter. More specifically, Riquelme et al. (2018) provided an extensive empirical study of benchmark algorithms for contextual-bandits through the lens of Thompson Sampling (Thompson, 1933; Chapelle and Li, 2011). They found that decoupling representation learning and uncertainty estimation improves performance.

In this paper, we study a new neural contextual bandit algorithm, which learns a mapping to transform the raw features associated with each context-action pair using a deep neural network, and then performs an upper confidence bound (UCB)-type (shallow) exploration over the last layer of the neural network. We prove a sublinear regret of the proposed algorithm by exploiting the UCB exploration techniques in linear contextual bandits (Abbasi-Yadkori et al., 2011) and the analysis of deep overparameterized neural networks using neural tangent kernel (Jacot et al., 2018). Our theory confirms the effectiveness of decoupling the deep representation learning and the UCB exploration in contextual bandits (Riquelme et al., 2018; Zahavy and Mannor, 2019).

Contributions we summarize the main contributions of this paper as follows.

- We propose a contextual bandit algorithm, Neural-LinUCB, for solving a general class of contextual bandit problems without any assumption on the structure of the reward generating function. The proposed algorithm learns a deep representation to transform the raw feature vectors and performs UCB-type exploration in the last layer of the neural network, which we refer to as deep representation and shallow exploration. Compared with Zhou et al. (2020), our algorithm is much more computationally efficient in practice.

- We prove a $\widetilde{O}(\sqrt{T})$ regret for the proposed Neural-LinUCB algorithm, which matches the sublinear regret of linear contextual bandits (Chu et al., 2011; Abbasi-Yadkori et al., 2011). To the best

of our knowledge, this is the first work that theoretically shows the Neural-Linear schemes of contextual bandits are able to converge, which validates the empirical observation by Riquelme et al. (2018).

- We conduct experiments on contextual bandit problems based on real-world datasets, which demonstrates the good performance and computational efficiency of Neural-LinUCB over NeuralUCB and well aligns with our theory.

1.1 Additional related work

There is a line of related work to ours on the recent advance in the optimization and generalization analysis of deep neural networks. In particular, Jacot et al. (2018) first introduced the neural tangent kernel (NTK) to characterize the training dynamics of network outputs in the infinite width limit. From the notion of NTK, a fruitful line of research emerged and showed that loss functions of deep neural networks trained by (stochastic) gradient descent can converge to the global minimum (Du et al., 2019b; Allen-Zhu et al., 2019b; Du et al., 2019a; Zou et al., 2018; Zou and Gu, 2019). The generalization bounds for overparameterized deep neural networks are also established in Arora et al. (2019a,b); Allen-Zhu et al. (2019a); Cao and Gu (2019b,a). Recently, the NTK based analysis is also extended to the study of sequential decision problems including NeuralUCB (Zhou et al., 2020), and reinforcement learning algorithms (Cai et al., 2019; Liu et al., 2019; Wang et al., 2020; Xu and Gu, 2020).

Our algorithm is also different from Langford and Zhang (2008); Agarwal et al. (2014) which reduce the bandit problem to supervised learning. Moreover, their algorithms need to access an oracle that returns the optimal policy in a policy class given a sequence of context and reward vectors, whose regret depends on the VC-dimension of the policy class.

Notation We use $[k]$ to denote a set $\{1, \ldots, k\}$, $k \in \mathbb{N}^+$. $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^\top \mathbf{x}}$ is the Euclidean norm of a vector $\mathbf{x} \in \mathbb{R}^d$. For a matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, we denote by $\|\mathbf{W}\|_2$ and $\|\mathbf{W}\|_F$ its operator norm and Frobenius norm respectively. For a semi-definite matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ and a vector $\mathbf{x} \in \mathbb{R}^d$, we denote the Mahalanobis norm as $\|\mathbf{x}\|_{\mathbf{A}} = \sqrt{\mathbf{x}^\top \mathbf{A} \mathbf{x}}$. Throughout this paper, we reserve the notations $\{C_i\}_{i=0,1,\ldots}$ to represent absolute positive constants that are independent of problem parameters such as dimension, sample size, iteration number, step size, network length and so on. The specific values of $\{C_i\}_{i=0,1,\ldots}$ can be different in different context. For a parameter of interest T and a function $f(T)$, we use notations such as $O(f(T))$ and $\Omega(f(T))$ to hide constant factors and $\widetilde{O}(f(T))$ to hide constant and logarithmic dependence of T.

2 Preliminaries

In this section, we provide the background of contextual bandits and deep neural networks.

2.1 Linear contextual bandits

A contextual bandit is characterized by a tuple $(\mathcal{S}, \mathcal{A}, r)$, where $\mathcal{S}$ is the context (state) space, $\mathcal{A}$ is the arm (action) space, and r encodes the unknown *reward generating function* at all context-arm pairs. A learning agent, who knows $\mathcal{S}$ and $\mathcal{A}$ but does not know the true reward r (values bounded in $(0, 1)$ for simplicity), needs to interact with the contextual bandit for T rounds. At each round

$t = 1, \ldots, T$, the agent first observes a context $s_t \in \mathcal{S}$ chosen by the environment; then it needs to adaptively select an arm $a_t \in \mathcal{A}$ based on its past observations; finally it receives a reward $\widehat{r}_t(\mathbf{x}_{s,a_t}) = r(\mathbf{x}_{s,a_t}) + \xi_t$, where $\mathbf{x}_{s,a} \in \mathbb{R}^d$ is a known feature vector for context-arm pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, and ξ_t is a random noise with zero mean. The agent's objective is to maximize its expected total reward over these T rounds, which is equivalent to minimizing the pseudo regret (Audibert et al., 2009):

$$R_T = \mathbb{E}\bigg[\sum_{t=1}^{T} \big(\widehat{r}(\mathbf{x}_{s_t,a_t^*}) - \widehat{r}(\mathbf{x}_{s_t,a_t})\big)\bigg], \tag{2.1}$$

where $a_t^* \in \operatorname{argmax}_{a \in \mathcal{A}}\{r(\mathbf{x}_{s_t,a}) = \mathbb{E}[\widehat{r}(\mathbf{x}_{s_t,a})]\}$. To simplify the exposition, we use $\mathbf{x}_{t,a}$ to denote $\mathbf{x}_{s_t,a}$ since the context only depends on the round index t in most bandit problems, and we assume $\mathcal{A} = [K]$.

In some practical problems, the agent has a prior knowledge that the reward-generating function r has some specific parametric form. For instance, in linear contextual bandits, the agent knows that $r(\mathbf{x}_{s,a}) = \mathbf{x}_{s,a}^\top \boldsymbol{\theta}^*$ for some unknown weight vector $\boldsymbol{\theta}^* \in \mathbb{R}^d$. One provably sample efficient algorithm for linear contextual bandits is Linear Upper Confidence Bound (LinUCB) (Abbasi-Yadkori et al., 2011). Specifically, at each round t, LinUCB chooses action by the following strategy

$$a_t = \operatorname*{argmax}_{a \in [K]} \Big\{\mathbf{x}_{t,a}^\top \boldsymbol{\theta}_t + \alpha_t \|\mathbf{x}_{t,a}\|_{\mathbf{A}_t^{-1}}\Big\},$$

where $\boldsymbol{\theta}_t$ is a point estimate of $\boldsymbol{\theta}^*$, $\mathbf{A}_t = \lambda \mathbf{I} + \sum_{i=1}^{t} \mathbf{x}_{i,a_i} \mathbf{x}_{i,a_i}^\top$ with some $\lambda > 0$ is a matrix defined based on the historical context-arm pairs, and $\alpha_t > 0$ is a tuning parameter that controls the exploration rate in LinUCB.

2.2 Deep neural networks

In this paper, we use $f(\mathbf{x})$ to denote a neural network with input data $\mathbf{x} \in \mathbb{R}^d$. Let L be the number of hidden layers and $\mathbf{W}_l \in \mathbb{R}^{m_l \times m_{l-1}}$ be the weight matrices in the l-th layer, where $l = 1, \ldots, L$, $m_1 = \ldots = m_{L-1} = m$ and $m_0 = m_L = d$. Then a L-hidden layer neural network is defined as

$$f(\mathbf{x}) = \sqrt{m} \boldsymbol{\theta}^{*\top} \sigma_L(\mathbf{W}_L \sigma_{L-1}(\mathbf{W}_{L-1} \cdots \sigma_1(\mathbf{W}_1 \mathbf{x}) \cdots)), \tag{2.2}$$

where σ_l is an activation function and $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is the weight of the output layer. To simplify the presentation, we will assume $\sigma_1 = \sigma_2 = \ldots = \sigma_L = \sigma$ is the ReLU activation function, i.e., $\sigma(x) = \max\{0, x\}$ for $x \in \mathbb{R}$. We denote $\mathbf{w} = (\operatorname{vec}(\mathbf{W}_1)^\top, \ldots, \operatorname{vec}(\mathbf{W}_L)^\top)^\top$, which is the concatenation of the vectorized weight parameters of all hidden layers of the neural network. We also write $f(\mathbf{x}; \boldsymbol{\theta}^*, \mathbf{w}) = f(\mathbf{x})$ in order to explicitly specify the weight parameters of neural network f. It is easy to show that the dimension p of vector $\mathbf{w}$ satisfies $p = (L-2)m^2 + 2md$. To simplify the notation, we define $\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})$ as the output of the L-th hidden layer of neural network f.

$$\boldsymbol{\phi}(\mathbf{x}; \mathbf{w}) = \sqrt{m} \sigma(\mathbf{W}_L \sigma(\mathbf{W}_{L-1} \cdots \sigma(\mathbf{W}_1 \mathbf{x}) \cdots)). \tag{2.3}$$

Note that $\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})$ itself can also be viewed as a neural network with vector-valued outputs.

3 Deep Representation and Shallow Exploration

The linear parametric form in linear contextual bandit might produce biased estimates of the reward due to the lack of representation power (Snoek et al., 2015; Riquelme et al., 2018). In contrast, it is well known that deep neural networks are powerful enough to approximate an arbitrary function (Cybenko, 1989). Therefore, it would be a natural extension for us to guess that the reward generating function $r(\cdot)$ can be represented by a deep neural network. Nonetheless, deep neural networks usually have a prohibitively large dimension for weight parameters, which makes the exploration in neural networks based UCB algorithm inefficient (Kveton et al., 2020; Zhou et al., 2020).

In this work, we study a more realistic setting, where the hidden layers of a deep neural network are used to represent the features and the exploration is only performed in the last layer of the neural network (Riquelme et al., 2018; Zahavy and Mannor, 2019). In particular, we assume that the reward generating function $r(\cdot)$ can be expressed as the inner product between a deep represented feature vector and an exploration weight parameter, namely, $r(\cdot) = \langle \boldsymbol{\theta}^*, \boldsymbol{\psi}(\cdot) \rangle$, where $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is some weight parameter and $\boldsymbol{\psi}(\cdot)$ is an unknown feature mapping. This decoupling of the representation and the exploration will achieve the best of both worlds: efficient exploration in shallow (linear) models and high expressive power of deep models. To learn the unknown feature mapping, we propose to use a neural network to approximate it. In what follows, we will describe a neural contextual bandit algorithm that uses the output of the last hidden layer of a neural network to transform the raw feature vectors (*deep representation*) and performs UCB-type exploration in the last layer of the neural network (*shallow exploration*). Since the exploration is performed only in the last linear layer, we call this procedure Neural-LinUCB, which is displayed in Algorithm 1.

Specifically, in round t, the agent receives an action set with raw features $\mathcal{X}_t = \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,K}\}$. Then the agent chooses an arm a_t that maximizes the following upper confidence bound:

$$a_t = \operatorname*{argmax}_{k \in [K]} \Big\{ \langle \boldsymbol{\phi}(\mathbf{x}_{t,k}; \mathbf{w}_{t-1}), \boldsymbol{\theta}_{t-1} \rangle + \alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,k}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} \Big\}, \tag{3.1}$$

where $\boldsymbol{\theta}_{t-1}$ is a point estimate of the unknown weight in the last layer, $\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})$ is defined as in (2.3), $\mathbf{w}_{t-1}$ is an estimate of all the weight parameters in the hidden layers of the neural network, $\alpha_t > 0$ is the algorithmic parameter controlling the exploration, and $\mathbf{A}_t$ is a matrix defined based on historical transformed features:

$$\mathbf{A}_t = \lambda \mathbf{I} + \sum_{i=1}^{t} \boldsymbol{\phi}(\mathbf{x}_{i,a_i}; \mathbf{w}_{i-1}) \boldsymbol{\phi}(\mathbf{x}_{i,a_i}; \mathbf{w}_{i-1})^\top, \tag{3.2}$$

and $\lambda > 0$. After pulling arm a_t, the agent will observe a noisy reward $\widehat{r}_t := \widehat{r}(\mathbf{x}_{t,a_t})$ defined as

$$\widehat{r}(\mathbf{x}_{t,k}) = r(\mathbf{x}_{t,k}) + \xi_t, \tag{3.3}$$

where ξ_t is an independent ν-subGaussian random noise for some $\nu > 0$ and $r(\cdot)$ is an unknown reward function. In this paper, we will interchangeably use notation $\widehat{r}_t$ to denote the reward received at the t-th step and an equivalent notation $\widehat{r}(\mathbf{x})$ to express its dependence on the feature vector $\mathbf{x}$.

Upon receiving the reward $\widehat{r}_t$, the agent updates its estimate $\boldsymbol{\theta}_t$ of the output layer weight by

using the same ℓ^2-regularized least-squares estimate in linear contextual bandits (Abbasi-Yadkori et al., 2011). In particular, we have

$$\boldsymbol{\theta}_t = \mathbf{A}_t^{-1}\mathbf{b}_t, \tag{3.4}$$

where $\mathbf{b}_t = \sum_{i=1}^{t} \widehat{r}_i \phi(\mathbf{x}_{i,a_i}; \mathbf{w}_{i-1})$.

To save the computation, the neural network $\phi(\cdot; \mathbf{w}_t)$ will be updated once every H steps. Therefore, we have $\mathbf{w}_{(q-1)H+1} = \ldots = \mathbf{w}_{qH}$ for $q = 1, 2, \ldots$. We call the time steps $\{(q-1)H + 1, \ldots, qH\}$ an epoch with length H. At time step $t = Hq$, for any $q = 1, 2, \ldots$, Algorithm 1 will retrain the neural network based on all the historical data. In Algorithm 2, our goal is to minimize the following empirical loss function:

$$\mathcal{L}_q(\mathbf{w}) = \sum_{i=1}^{qH} \big(\boldsymbol{\theta}_i^\top \phi(\mathbf{x}_{i,a_i}; \mathbf{w}) - \widehat{r}_i\big)^2. \tag{3.5}$$

In practice, one can further save computational cost by only feeding data $\{\mathbf{x}_{i,a_i}, \widehat{r}_i, \boldsymbol{\theta}_i\}_{i=(q-1)H+1}^{qH}$ from the q-th epoch into Algorithm 2 to update the parameter $\mathbf{w}_t$, which does not hurt the performance since the historical information has been encoded into the estimate of $\boldsymbol{\theta}_i$. In this paper, we will perform the following gradient descent step

$$\mathbf{w}_q^{(s)} = \mathbf{w}_q^{(s-1)} - \eta_q \nabla_{\mathbf{w}} \mathcal{L}_q(\mathbf{w}^{(s-1)}).$$

for $s = 1, \ldots, n$, where $\mathbf{w}_q^{(0)} = \mathbf{w}^{(0)}$ is chosen as the same random initialization point. We will discuss more about the initial point $\mathbf{w}^{(0)}$ in the next paragraph. Then Algorithm 2 outputs $\mathbf{w}_q^{(n)}$ and we set it as the updated weight parameter $\mathbf{w}_{Hq+1}$ in Algorithm 1. In the next round, the agent will receive another action set $\mathcal{X}_{t+1}$ with raw feature vectors and repeat the above steps to choose the sub-optimal arm and update estimation for contextual parameters.

Initialization: Recall that $\mathbf{w}$ is the collection of all hidden layer weight parameters of the neural network. We will follow the same initialization scheme as used in Zhou et al. (2020), where each entry of the weight matrices follows some Gaussian distribution. Specifically, for any $l \in \{1, \ldots, L-1\}$, we set $\mathbf{W}_l = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} \end{bmatrix}$, where each entry of $\mathbf{W}$ follows distribution $N(0, 4/m)$ independently; for $\mathbf{W}_L$, we set it as $\begin{bmatrix} \mathbf{V} & -\mathbf{V} \end{bmatrix}$, where each entry of $\mathbf{V}$ follows distribution $N(0, 2/m)$ independently.

Comparison with LinUCB and NeuralUCB: Compared with linear contextual bandits in Section 2.1, Algorithm 1 has a distinct feature that it learns a deep neural network to obtain a deep representation of the raw data vectors and then performs UCB exploration. This deep representation allows our algorithm to characterize more intrinsic and latent information about the raw data $\{\mathbf{x}_{t,k}\}_{t\in[T],k\in[K]} \subset \mathbb{R}^d$. However, the increased complexity of the feature mapping $\phi(\cdot; \mathbf{w})$ also introduces great hardness in training. For instance, a recent work by Zhou et al. (2020) also studied the neural contextual bandit problem, but different from (3.1), their algorithm (NeuralUCB) performs the UCB exploration on the entire network parameter space, which is $\mathbb{R}^{p+d}$. Note that in Zhou et al. (2020), they need to compute the inverse of a matrix $\mathbf{Z}_t \in \mathbb{R}^{(p+d)\times(p+d)}$, which is defined in a similar way to the matrix $\mathbf{A}_t$ in our paper except that $\mathbf{Z}_t$ is defined based on the gradient of the network instead of the output of the last hidden layer as in (3.2). In sharp contrast, $\mathbf{A}_t$ in our

Algorithm 1 Deep Representation and Shallow Exploration (Neural-LinUCB)

Input: regularization parameter $\lambda > 0$, number of total steps T, episode length H, exploration parameters $\{\alpha_t > 0\}_{t\in[T]}$
Initialization: $\mathbf{A}_0 = \lambda\mathbf{I}$, $\mathbf{b}_0 = \mathbf{0}$; entries of $\boldsymbol{\theta}_0$ follow $N(0, 1/d)$, and $\mathbf{w}^{(0)}$ is initialized as described in Section 3; $q = 1$; $\mathbf{w}_0 = \mathbf{w}^{(0)}$
for $t = 1, \ldots, T$ **do**
  receive feature vectors $\{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,K}\}$
  choose arm $a_t = \text{argmax}_{k\in[K]}\, \boldsymbol{\theta}_{t-1}^\top \boldsymbol{\phi}(\mathbf{x}_{t,k}; \mathbf{w}_{t-1}) + \alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,k}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}}$, and obtain reward $\widehat{r}_t$
  update $\mathbf{A}_t$ and $\mathbf{b}_t$ as follows:
    $\mathbf{A}_t = \mathbf{A}_{t-1} + \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})^\top$,
    $\mathbf{b}_t = \mathbf{b}_{t-1} + \widehat{r}_t \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})$,
  update $\boldsymbol{\theta}_t = \mathbf{A}_t^{-1}\mathbf{b}_t$
  if $\text{mod}(t, H) = 0$ **then**
    $\mathbf{w}_t \leftarrow$ output of Algorithm 2
    $q = q + 1$
  else
    $\mathbf{w}_t = \mathbf{w}_{t-1}$
  end if
end for
Output

paper is only of size $d \times d$ and thus is much more efficient and practical in implementation, which will be seen from our experiments in later sections.

We note that there is also a similar algorithm to our Neural-LinUCB presented in Deshmukh et al. (2020), where they studied the self-supervised learning loss in contextual bandits with neural network representation for computer vision problems. However, no regret analysis has been provided. When the feature mapping $\boldsymbol{\phi}(\cdot; \mathbf{w})$ is an identity function, the problem reduces to linear contextual bandits where we directly use $\mathbf{x}_t$ as the feature vector. In this case, it is easy to see that Algorithm 1 reduces to LinUCB (Chu et al., 2011) since we do not need to learn the representation parameter $\mathbf{w}$ anymore.

Comparison with Neural-Linear: Our algorithm is also similar to the Neural-Linear algorithm studied in Riquelme et al. (2018), which trains a deep neural network to learn a representation of the raw feature vectors, and then uses a Bayesian linear regression to estimate the uncertainty in the bandit problem. The difference between Neural-Linear (Riquelme et al., 2018) and Neural-LinUCB in this paper lies in the specific exploration strategies: Neural-Linear uses posterior sampling to estimate the weight parameter $\boldsymbol{\theta}^*$ via Bayesian linear regression, whereas Neural-LinUCB adopts upper confidence bound based techniques to estimate the weight $\boldsymbol{\theta}^*$. Nevertheless, both algorithms share the same idea of deep representation and shallow exploration, and we view our Neural-LinUCB algorithm as one instantiation of the Neural-Linear scheme proposed in Riquelme et al. (2018).

Algorithm 2 Update Weight Parameters with Gradient Descent

Input: initial point $\mathbf{w}_q^{(0)} = \mathbf{w}^{(0)}$, maximum iteration number n, step size η_q, and loss function defined in (3.5).
for $s = 1, \ldots, n$ **do**
  $\mathbf{w}_q^{(s)} = \mathbf{w}_q^{(s-1)} - \eta_q \nabla_{\mathbf{w}} \mathcal{L}_q(\mathbf{w}_q^{(s-1)})$.
end for
return $\mathbf{w}_q^{(n)}$

4 Main Theory

To analyze the regret bound of Algorithm 1, we first lay down some important assumptions on the neural contextual bandit model.

Assumption 4.1. For all $i \geq 1$ and $k \in [K]$, we assume that $\|\mathbf{x}_{i,k}\|_2 = 1$ and its entries satisfy $[\mathbf{x}_{i,k}]_j = [\mathbf{x}_{j,k}]_{j+d/2}$.

The assumption that $\|\mathbf{x}_{i,k}\|_2 = 1$ is not essential and is only imposed for simplicity, which is also used in Zou and Gu (2019); Zhou et al. (2020). Finally, the condition on the entries of $\mathbf{x}_{i,k}$ is also mild since otherwise we could always construct $\mathbf{x}'_{i,k} = [\mathbf{x}_{i,k}^\top, \mathbf{x}_{i,k}^\top]^\top / \sqrt{2}$ to replace it. An implication of Assumption 4.1 is that the initialization scheme in Algorithm 1 results in $\boldsymbol{\phi}(\mathbf{x}_{i,k}; \mathbf{w}^{(0)}) = \mathbf{0}$ for all $i \in [T]$ and $k \in [K]$.

We further impose the following stability condition on the spectral norm of the neural network gradient:

Assumption 4.2. There is a constant $\ell_{\text{Lip}} > 0$ such that it holds

$$\left\| \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{w}}(\mathbf{x}; \mathbf{w}_0) - \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{w}}(\mathbf{x}'; \mathbf{w}_0) \right\|_2 \leq \ell_{\text{Lip}} \|\mathbf{x} - \mathbf{x}'\|_2,$$

for all $\mathbf{x}, \mathbf{x}' \in \{\mathbf{x}_{i,k}\}_{i \in [T], k \in [K]}$.

The inequality in Assumption 4.2 looks like some Lipschitz condition. However, it is worth noting that here the gradient is taken with respect to the neural network weights while the Lipschitz condition is imposed on the feature parameter $\mathbf{x}$. Similar conditions are widely made in nonconvex optimization (Wang et al., 2014; Balakrishnan et al., 2017; Xu et al., 2017), in the name of first-order stability, which is essential to derive the convergence of alternating optimization algorithms. Furthermore, Assumption 4.2 is only required on the TK training data points and a specific weight parameter $\mathbf{w}_0$. Therefore, the condition will hold if the raw feature data lie in a benign subspace. A more thorough study of this stability condition is out of the scope of this paper, though it would be an interesting open direction in the theory of deep neural networks.

In order to analyze the regret bound of Algorithm 1, we need to characterize the properties of the deep neural network in (2.2) that is used to represent the feature vectors. Following a recent line of research (Jacot et al., 2018; Cao and Gu, 2019b; Arora et al., 2019b; Zhou et al., 2020), we

define the covariance between two data point $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ as follows.

$$\begin{aligned}
&\widetilde{\mathbf{\Sigma}}^{(0)}(\mathbf{x}, \mathbf{y}) = \mathbf{\Sigma}^{(0)}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}, \\
&\mathbf{\Lambda}^{(l)}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \mathbf{\Sigma}^{l-1}(\mathbf{x}, \mathbf{x}) & \mathbf{\Sigma}^{l-1}(\mathbf{x}, \mathbf{y}) \\ \mathbf{\Sigma}^{l-1}(\mathbf{y}, \mathbf{x}) & \mathbf{\Sigma}^{l-1}(\mathbf{y}, \mathbf{y}) \end{bmatrix}, \\
&\mathbf{\Sigma}^{(l)}(\mathbf{x}, \mathbf{y}) = 2\mathbb{E}_{(u,v)\sim N(\mathbf{0}, \mathbf{\Lambda}^{(l-1)}(\mathbf{x},\mathbf{y}))}[\sigma(u)\sigma(v)], \\
&\widetilde{\mathbf{\Sigma}}^{(l)}(\mathbf{x}, \mathbf{y}) = 2\widetilde{\mathbf{\Sigma}}^{(l-1)}(\mathbf{x}, \mathbf{y})\mathbb{E}_{u,v}[\dot{\sigma}(u)\dot{\sigma}(v)] + \mathbf{\Sigma}^{(l)}(\mathbf{x}, \mathbf{y}),
\end{aligned} \tag{4.1}$$

where $(u, v) \sim N(\mathbf{0}, \mathbf{\Lambda}^{(l-1)}(\mathbf{x}, \mathbf{y}))$, and $\dot{\sigma}(\cdot)$ is the derivative of activation function $\sigma(\cdot)$. We denote the neural tangent kernel (NTK) matrix $\mathbf{H}$ by a $\mathbb{R}^{TK\times TK}$ matrix defined on the dataset of all feature vectors $\{\mathbf{x}_{t,k}\}_{t\in[T],k\in[K]}$. Renumbering $\{\mathbf{x}_{t,k}\}_{t\in[T],k\in[K]}$ as $\{\mathbf{x}_i\}_{i=1,\ldots,TK}$, then each entry $\mathbf{H}_{ij}$ is defined as

$$\mathbf{H}_{ij} = \frac{1}{2}\big(\widetilde{\mathbf{\Sigma}}^{(L)}(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{\Sigma}^{(L)}(\mathbf{x}_i, \mathbf{x}_j)\big), \tag{4.2}$$

for all $i, j \in [TK]$. Based on the above definition, we impose the following assumption on $\mathbf{H}$.

Assumption 4.3. The neural tangent kernel defined in (4.2) is positive definite, i.e., $\lambda_{\min}(\mathbf{H}) \geq \lambda_0$ for some constant $\lambda_0 > 0$.

Assumption 4.3 essentially requires the neural tangent kernel matrix $\mathbf{H}$ to be non-singular, which is a mild condition and also imposed in other related work (Du et al., 2019a; Arora et al., 2019b; Cao and Gu, 2019b; Zhou et al., 2020). Moreover, it is shown that Assumption 4.3 can be easily derived from Assumption 4.1 for two-layer ReLU networks (Oymak and Soltanolkotabi, 2020; Zou and Gu, 2019). Therefore, Assumption 4.3 is mild or even negligible given the non-degeneration assumption on the feature vectors. Also note that matrix $\mathbf{H}$ is only defined based on layers $l = 1, \ldots, L$ of the neural network, and does not depend on the output layer $\boldsymbol{\theta}$. It is easy to extend the definition of $\mathbf{H}$ to the NTK matrix defined on all layers including the output layer $\boldsymbol{\theta}$, which would also be positive definite by Assumption 4.3 and the recursion in (4.2).

Before we present the regret analysis of the neural contextual bandit, we need to modify the regret defined in (2.1) to account for the randomness of the neural network initialization. For a fixed time horizon T, we define the regret of Algorithm 1 as follows.

$$R_T = \mathbb{E}\bigg[\sum_{t=1}^{T}\big(\widehat{r}(\mathbf{x}_{t,a_t^*}) - \widehat{r}(\mathbf{x}_{t,a_t})\big)|\mathbf{w}^{(0)}\bigg], \tag{4.3}$$

where the expectation is taken over the randomness of the reward noise. Note that R_T defined in (4.3) is still a random variable since the initialization of Algorithm 2 is randomly generated.

Now we are going to present the regret bound of the proposed algorithm.

Theorem 4.4. Suppose Assumptions 4.1, 4.2 and 4.3 hold. Assume that $\|\boldsymbol{\theta}^*\|_2 \leq M$ for some positive constant $M > 0$. For any $\delta \in (0, 1)$, let us choose α_t in Neural-LinUCB as

$$\alpha_t = \nu\sqrt{2\big(d\log(1 + t\log(HK)/\lambda) + \log(1/\delta)\big)} + \lambda^{1/2}M.$$

We choose the step size η_q of Algorithm 2 as

$$\eta_q \leq C_0\big(d^2mnT^{5.5}L^6\log(TK/\delta)\big)^{-1},$$

and the width of the neural network satisfies $m = \mathrm{poly}(L, d, 1/\delta, H, \log(TK/\delta))$. With probability at least $1-\delta$ over the randomness of the initialization of the neural network, it holds that

$$R_T \leq C_1\alpha_T\sqrt{Td\log\Big(1+\frac{TG^2}{\lambda d}\Big)} + \frac{C_2\ell_{\mathrm{Lip}}L^3d^{5/2}T\sqrt{\log m\log(\frac{1}{\delta})\log(\frac{TK}{\delta})}\|\mathbf{r}-\widetilde{\mathbf{r}}\|_{\mathbf{H}^{-1}}}{m^{1/6}},$$

where $\{C_i\}_{i=0,1,2}$ are absolute constants independent of the problem parameters, $\|\mathbf{r}\|_{\mathbf{A}} = \sqrt{\mathbf{r}^\top\mathbf{A}\mathbf{r}}$, $\mathbf{r} = (r(\mathbf{x}_1), r(\mathbf{x}_2), \ldots, r(\mathbf{x}_{TK}))^\top \in \mathbb{R}^{TK}$ and $\widetilde{\mathbf{r}} = (f(\mathbf{x}_1;\boldsymbol{\theta}_0,\mathbf{w}_0),\ldots,f(\mathbf{x}_{TK};\boldsymbol{\theta}_{T-1},\mathbf{w}_{T-1}))^\top \in \mathbb{R}^{TK}$.

Remark 4.5. Theorem 4.4 shows that the regret of Algorithm 1 can be bounded by two parts: the first part is of order $\widetilde{O}(\sqrt{T})$, which resembles the regret bound of linear contextual bandits (Abbasi-Yadkori et al., 2011); the second part is of order $\widetilde{O}(m^{-1/6}T\sqrt{(\mathbf{r}-\widetilde{\mathbf{r}})^\top\mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})})$, which depends on the estimation error of the neural network f for the reward generating function r and the neural tangent kernel $\mathbf{H}$.

Remark 4.6. For the ease of presentation, let us denote $\mathcal{E} := \|\mathbf{r}-\widetilde{\mathbf{r}}\|_{\mathbf{H}^{-1}}$. If we have $\mathcal{E} = O(1)$, the total regret in Theorem 4.4 becomes $\widetilde{O}(m^{-1/6}T)$. If we further choose a sufficiently overparameterized neural network with $m \geq T^3$, then the regret reduces to $\widetilde{O}(\sqrt{T})$ which matches the regret of linear contextual bandits (Abbasi-Yadkori et al., 2011). We remark that there is a similar assumption in Zhou et al. (2020) where they assume that $\mathbf{r}^\top\mathbf{H}^{-1}\mathbf{r}$ can be upper bounded by a constant. They show that this term can be bounded by the RKHS norm of $\mathbf{r}$ if it belongs to the RKHS induced by the neural tangent kernel (Arora et al., 2019a,b; Lee et al., 2019). In addition, $\mathcal{E}$ here is the difference between the true reward function and the neural network function, which can also be small if the deep neural network function well approximates the reward generating function $r(\cdot)$.

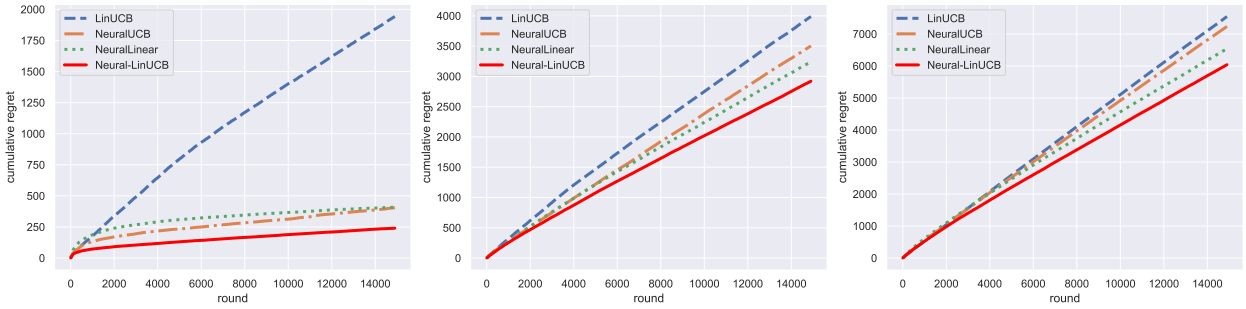
5 Experiments

In this section, we provide empirical evaluations of the proposed Neural-LinUCB algorithm. As we have discussed in Section 3, Neural-LinUCB could be viewed as an instantiation of the Neural-Linear scheme studied in Riquelme et al. (2018) except that we use the UCB exploration instead of the posterior sampling exploration therein. Note that there has been an extensive comparison (Riquelme et al., 2018) of the Neural-Linear methods with many other baselines such as greedy algorithms, Variational Inference, Expectation-Propagation, Bayesian Non-parametrics and so on. Therefore, we do not seek a thorough empirical comparison of Neural-LinUCB with all existing bandits algorithms. We refer readers who are interested in the performance of Neural-Linear methods with deep representation and shallow exploration compared with a vast of baselines in the literature to the benchmark study by Riquelme et al. (2018). In this experiment, we only aim to show the advantages of our algorithm over the following baselines: (1) Neural-Linear (Riquelme et al., 2018); (2) LinUCB (Chu et al., 2011), which does not have a *deep representation* of the feature vectors; and (3) NeuralUCB (Zhou et al., 2020), which performs UCB exploration on all the parameters of the neural network instead of the *shallow exploration* used in our paper.

Datasets: we evaluate the performances of all algorithms on bandit problems created from real-world data. Specifically, we use datasets *(Shuttle) Statlog*, *Magic* and *Covertype* from UCI machine learning repository (Dua and Graff, 2017), of which the details are presented in Table 1. In Table 1, each instance represents a feature vector $\mathbf{x} \in \mathbb{R}^d$ that is associated with one of the K arms, and dimension d is the number of attributes in each instance.

Table 1: Specifications of datasets from the UCI machine learning repository used in this paper.

	Statlog	*Magic*	*Covertype*
Number of attributes	9	11	54
Number of arms	7	2	7
Number of instances	58,000	19,020	581,012



(a) Statlog (b) Magic (c) Covertype

Figure 1: The cumulative regrets of LinUCB, NeuralUCB, Neural-Linear and Neural-LinUCB over 15,000 rounds. Experiments are averaged over 10 repetitions.

Implementations: for LinUCB, we follow the setting in Li et al. (2010) to use disjoint models for different arms. For NeuralUCB, Neural-Linear and Neural-LinUCB, we use a ReLU neural network defined as in (2.2), where we set the width $m = 2000$ and the hidden layer length $L = 2$. We set the time horizon $T = 15,000$, which is the total number of rounds for each algorithm on each dataset. We use gradient decent to optimize the network weights, with a step size η_q =1e-5 and maximum iteration number $n = 1,000$. To speed up the training process, the network parameter $\mathbf{w}$ is updated every $H = 100$ rounds. We also apply early stopping when the loss difference of two consecutive iterations is smaller than a threshold of 1e-6. We set $\lambda = 1$ and $\alpha_t = 0.02$ for all algorithms, $t \in [T]$. Following the setting in Riquelme et al. (2018), we use round-robin to independently select each arm for 3 times at the beginning of each algorithm. For NeuralUCB, since it is computationally unaffordable to perform the original UCB exploration as displayed in Zhou et al. (2020), we follow their experimental setting to replace the matrix $\mathbf{Z}_t \in \mathbb{R}^{(d+p)\times(d+p)}$ in Zhou et al. (2020) with its diagonal matrix.

Results: we plot the cumulative regret of all algorithms versus round in Figures 1(a), 1(b) and 1(c). The results are reported based on the average of 10 repetitions over different random shuffles of the datasets. It can be seen that algorithms based on neural network representations (NeuralUCB,

Neural-Linear and Neural-LinUCB) consistently outperform the linear contextual bandit method LinUCB, which shows that linear models may lack representation power and find biased estimates for the underlying reward generating function. Furthermore, our proposed Neural-LinUCB consistently achieves a significantly lower cumulative regret than the NeuralUCB algorithm. The possible reasons for this improvement are two-fold: (1) the replacement of the feature matrix with its diagonal matrix in NeuralUCB causes the UCB exploration to be biased and not accurate enough; (2) the exploration over the whole weight space of a deep neural network may potentially lead to spurious local minima that generalize poorly to unseen data. In addition, our algorithm also achieves a lower regret than Neural-Linear on the tested datasets.

The results in our experiment are well aligned with our theory that deep representation and shallow exploration are sufficient to guarantee a good performance of neural contextual bandit algorithms, which is also consistent with the findings in existing literature (Riquelme et al., 2018) that decoupling the representation learning and uncertainty estimation improves the performance. Moreover, we also find that our Neural-LinUCB algorithm is much more computationally efficient than NeuralUCB since we only perform the UCB exploration on the last layer of the neural network. In specific, on the *Statlog* dataset, it takes $19,028$ seconds for NeuralUCB to finish $15,000$ rounds and achieve the regret in Figure 1(a), while it only takes 783 seconds for Neural-LinUCB. On the *Magic* dataset, the runtimes of NeuralUCB and Neural-LinUCB are $18,579$ seconds and $7,263$ seconds respectively. And on the *Covertype* dataset, the runtimes of NeuralUCB and Neural-LinUCB are $11,941$ seconds and $3,443$ seconds respectively. For practical applications in the real-world with larger problem sizes, we believe that the improvement of our algorithm in terms of the computational efficiency will be more significant.

6 Conclusions

In this paper, we propose a new neural contextual bandit algorithm called Neural-LinUCB, which uses the hidden layers of a ReLU neural network as a deep representation of the raw feature vectors and performs UCB type exploration on the last layer of the neural network. By incorporating techniques in liner contextual bandits and neural tangent kernels, we prove that the proposed algorithm achieves a sublinear regret when the width of the network is sufficiently large. This is the first regret analysis of neural contextual bandit algorithms with deep representation and shallow exploration, which have been observed in practice to work well on many benchmark bandit problems (Riquelme et al., 2018). We also conducted experiments on real-world datasets to demonstrate the advantage of the proposed algorithm over linear contextual bandits and existing neural contextual bandit algorithms.

A Proof of the Main Theory

In this section, we provide the proof of the regret bound for Neural-LinUCB. Recall that in neural contextual bandits, we do not assume a specific formulation of the underlying reward generating function $r(\cdot)$. Instead, we use deep neural networks defined in Section 2.2 to approximate $r(\cdot)$. We will first show that the reward generating function $r(\cdot)$ can be approximated by the local linearization of the overparameterized neural network near the initialization weight $\mathbf{w}^{(0)}$. In particular, we denote

the gradient of $\boldsymbol{\phi}(\mathbf{x};\mathbf{w})$ with respect to $\mathbf{w}$ by $\mathbf{g}(\mathbf{x};\mathbf{w})$, namely,

$$\mathbf{g}(\mathbf{x};\mathbf{w}) = \nabla_{\mathbf{w}}\boldsymbol{\phi}(\mathbf{x};\mathbf{w}), \tag{A.1}$$

which is a matrix in $\mathbb{R}^{d\times p}$. We define $\phi_j(\mathbf{x};\mathbf{w})$ to be the j-th entry of vector $\boldsymbol{\phi}(\mathbf{x};\mathbf{w})$, for any $j\in[d]$. Then, we can prove the following lemma.

Lemma A.1. Suppose Assumptions 4.3 hold. Then there exists $\mathbf{w}^*\in\mathbb{R}^p$ such that $\|\mathbf{w}^*-\mathbf{w}^{(0)}\|_2 \le 1/\sqrt{m}\sqrt{(\mathbf{r}-\widetilde{\mathbf{r}})^\top\mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})}$ and it holds that

$$r(\mathbf{x}_{t,k}) = \boldsymbol{\theta}^{*\top}\boldsymbol{\phi}(\mathbf{x}_{t,k};\mathbf{w}_{t-1}) + \boldsymbol{\theta}_0^\top\mathbf{g}(\mathbf{x}_{t,k};\mathbf{w}^{(0)})\big(\mathbf{w}^*-\mathbf{w}^{(0)}\big),$$

for all $k\in[K]$ and $t=1,\ldots,T$.

Lemma A.1 implies that the reward generating function $r(\cdot)$ at points $\{\mathbf{x}_{i,k}\}_{i\in[T],k\in[K]}$ can be approximated by a linear function around the initial point $\mathbf{w}^{(0)}$. Note that a similar lemma is also proved in Zhou et al. (2020) for NeuralUCB.

The next lemma shows the upper bounds of the output of the neural network $\boldsymbol{\phi}$ and its gradient.

Lemma A.2. Suppose Assumptions 4.1 and 4.3 hold. For any round index $t\in[T]$, suppose it is in the q-th epoch of Algorithm 2, i.e., $t=(q-1)H+i$ for some $i\in[H]$. If the step size η_q in Algorithm 2 satisfies

$$\eta \le \frac{C_0}{d^2mnT^{5.5}L^6\log(TK/\delta)},$$

and the width of the neural network satisfies

$$m \ge \max\{L\log(TK/\delta), dL^2\log(m/\delta), \delta^{-6}H^{18}L^{16}\log^3(TK)\}, \tag{A.2}$$

then, with probability at least $1-\delta$ we have

$$\begin{aligned}
\|\mathbf{w}_t-\mathbf{w}^{(0)}\|_2 &\le \frac{\delta^{3/2}}{m^{1/2}Tn^{9/2}L^6\log^3(m)},\\
\|\mathbf{g}(\mathbf{x}_{t,k};\mathbf{w}^{(0)})\|_F &\le C_1\sqrt{dLm},\\
\|\boldsymbol{\phi}(\mathbf{x};\mathbf{w}_t)\|_2 &\le \sqrt{d\log(n)\log(TK/\delta)},
\end{aligned}$$

for all $t\in[T]$, $k\in[K]$, where the neural network $\boldsymbol{\phi}$ is defined in (2.3) and its gradient is defined in (A.1).

The next lemma shows that the neural network $\boldsymbol{\phi}(\mathbf{x};\mathbf{w})$ is close to a linear function in terms of the weight $\mathbf{w}$ parameter around a small neighborhood of the initialization point $\mathbf{w}^{(0)}$.

Lemma A.3 (Theorems 5 in Cao and Gu (2019a))**.** Let $\mathbf{w},\mathbf{w}'$ be in the neighborhood of $\mathbf{w}_0$, i.e., $\mathbf{w},\mathbf{w}'\in\mathbb{B}(\mathbf{w}_0,\omega)$ for some $\omega>0$. Consider the neural network defined in (2.3), if the width m and the radius ω of the neighborhood satisfy

$$m \ge C_0\max\{dL^2\log(m/\delta), \omega^{-4/3}L^{-8/3}\log(TK)\log(m/(\omega\delta))\},$$

$$\omega \leq C_1 L^{-5} (\log m)^{-3/2},$$

then for all $\mathbf{x} \in \{\mathbf{x}_{t,k}\}_{t\in[T],k\in[K]}$, with probability at least $1-\delta$ it holds that

$$|\phi_j(\mathbf{x};\mathbf{w}) - \widehat{\phi}_j(\mathbf{x};\mathbf{w})| \leq C_2 \omega^{4/3} L^3 d^{-1/2} \sqrt{m \log m},$$

where $\widehat{\phi}_j(\mathbf{x};\mathbf{w})$ is the linearization of $\phi_j(\mathbf{x};\mathbf{w})$ at $\mathbf{w}'$ defined as follow:

$$\widehat{\phi}_j(\mathbf{x};\mathbf{w}) = \phi_j(\mathbf{x};\mathbf{w}') + \langle \nabla_{\mathbf{w}} \phi_j(\mathbf{x};\mathbf{w}'), \mathbf{w} - \mathbf{w}' \rangle. \tag{A.3}$$

Similar results on the local linearization of an overparameterized neural network are also presented in Allen-Zhu et al. (2019b); Cao and Gu (2019a).

For the output layer $\boldsymbol{\theta}^*$, we perform a UCB type exploration and thus we need to characterize the uncertainty of the estimation. The next lemma shows the confidence bound of the estimate $\boldsymbol{\theta}_t$ in Algorithm 1.

Lemma A.4. Suppose Assumption and 4.3 hold. For any $\delta \in (0,1)$, with probability at least $1-\delta$, the distance between the estimated weight vector $\boldsymbol{\theta}_t$ by Algorithm 1 and $\boldsymbol{\theta}^*$ can be bounded as follows:

$$\begin{aligned}
&\bigg\| \boldsymbol{\theta}_t - \boldsymbol{\theta}^* - \mathbf{A}_t^{-1} \sum_{s=1}^{t} \boldsymbol{\phi}(\mathbf{x}_{s,a_s}; \mathbf{w}_{s-1}) \boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{s,a_s}; \mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)}) \bigg\|_{\mathbf{A}_t} \\
&\leq \nu \sqrt{2\big(d \log(1 + t(\log HK)/\lambda) + \log 1/\delta\big)} + \lambda^{1/2} M,
\end{aligned}$$

for any $t \in [T]$.

Note that the confidence bound in Lemma A.4 is different from the standard result for linear contextual bandits in Abbasi-Yadkori et al. (2011). The additional term on the left hand side of the confidence bound is due to the bias caused by the representation learning using a deep neural network. To deal with this extra term, we need the following technical lemma.

Lemma A.5. Assume that $\mathbf{A}_t = \lambda \mathbf{I} + \sum_{s=1}^{t} \boldsymbol{\phi}_s \boldsymbol{\phi}_s^\top$, where $\boldsymbol{\phi}_t \in \mathbb{R}^d$ and $\|\boldsymbol{\phi}_t\|_2 \leq G$ for all $t \geq 1$ and some constants $\lambda, G > 0$. Let $\{\zeta_t\}_{t=1,\ldots}$ be a real-value sequence such that $|\zeta_t| \leq U$ for some constant $U > 0$. Then we have

$$\bigg\| \mathbf{A}_t^{-1} \sum_{s=1}^{t} \boldsymbol{\phi}_s \zeta_s \bigg\|_2 \leq 2Ud, \quad \forall t = 1, 2, \ldots$$

The next lemma provides some standard bounds on the feature matrix $\mathbf{A}_t$, which is a combination of Lemma 10 and Lemma 11 in Abbasi-Yadkori et al. (2011).

Lemma A.6. Let $\{\mathbf{x}_t\}_{t=1}^{\infty}$ be a sequence in $\mathbb{R}^d$ and $\lambda > 0$. Suppose $\|\mathbf{x}_t\|_2 \leq G$ and $\lambda \geq \max\{1, G^2\}$ for some $G > 0$. Let $\mathbf{A}_t = \lambda \mathbf{I} + \sum_{s=1}^{t} \mathbf{x}_t \mathbf{x}_t^\top$. Then we have

$$\det(\mathbf{A}_t) \leq (\lambda + tG^2/d)^d, \quad \text{and} \quad \sum_{t=1}^{T} \|\mathbf{x}_t\|^2_{\mathbf{A}_{t-1}^{-1}} \leq 2 \log \frac{\det(\mathbf{A}_T)}{\det(\lambda \mathbf{I})} \leq 2d \log(1 + TG^2/(\lambda d)).$$

Now we are ready to prove the regret bound of Algorithm 1.

Proof of Theorem 4.4. For a time horizon T, without loss of generality, we assume $T = QH$ for some epoch number Q. By the definition of regret in (4.3), we have

$$R_T = \mathbb{E}\bigg[\sum_{t=1}^{T}(\widehat{r}(\mathbf{x}_{t,a_t^*}) - \widehat{r}(\mathbf{x}_{t,a_t}))\bigg] = \mathbb{E}\bigg[\sum_{q=1}^{Q}\sum_{i=1}^{H}(\widehat{r}(\mathbf{x}_{qH+i,a_{qH+i}^*}) - \widehat{r}(\mathbf{x}_{qH+i,a_{qH+i}}))\bigg].$$

Note that for the simplicity of presentation, we omit the conditional expectation notation of $\mathbf{w}^{(0)}$ in the rest of the proof when the context is clear. In the second equation, we rewrite the time index $t = qH + i$ as the i-th iteration in the q-th epoch.

By the definition in (3.3), we have $\mathbb{E}[\widehat{r}(\mathbf{x}_{t,k})|\mathbf{x}_{t,k}] = r(\mathbf{x}_{t,k})$ for all $t \in [T]$ and $k \in K$. Based on the linearization of reward generating function, we can decompose the instaneous regret into different parts and upper bound them individually. In particular, by Lemma A.1, there exists a vector $\mathbf{w}^* \in \mathbb{R}^p$ such that we can write the expectation of the reward generating function as a linear function. Then it holds that

$$\begin{aligned}
r(\mathbf{x}_{t,a_t^*}) - r(\mathbf{x}_{t,a_t}) &= \boldsymbol{\theta}_0^\top\big[\mathbf{g}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}^{(0)}\big) - \mathbf{g}\big(\mathbf{x}_{t,a_t};\mathbf{w}^{(0)}\big)\big]\big(\mathbf{w}^* - \mathbf{w}^{(0)}\big)\\
&\quad + \boldsymbol{\theta}^{*\top}\big[\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) - \boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\big]\\
&= \boldsymbol{\theta}_0^\top\big[\mathbf{g}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}^{(0)}\big) - \mathbf{g}\big(\mathbf{x}_{t,a_t};\mathbf{w}^{(0)}\big)\big]\big(\mathbf{w}^* - \mathbf{w}^{(0)}\big)\\
&\quad + \boldsymbol{\theta}_{t-1}^\top\big[\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) - \boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\big]\\
&\quad - \big(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\big)^\top\big[\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) - \boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\big]. \qquad \text{(A.4)}
\end{aligned}$$

The first term in (A.4) can be easily bounded using the first order stability in Assumption 4.2 and the distance between $\mathbf{w}^*$ and $\mathbf{w}^{(0)}$ in Lemma A.1. The second term in (A.4) is related to the optimistic rule of choosing arms in Line 5 of Algorithm 1, which can be bounded using the same technique for LinUCB (Abbasi-Yadkori et al., 2011). For the last term in (A.4), we need to prove that the estimate of weight parameter $\boldsymbol{\theta}_{t-1}$ lies in a confidence ball centered at $\boldsymbol{\theta}^*$. For the ease of notation, we define

$$\mathbf{M}_t = \mathbf{A}_t^{-1}\sum_{s=1}^{t}\boldsymbol{\phi}(\mathbf{x}_{s,a_s};\mathbf{w}_{s-1})\boldsymbol{\theta}_0^\top\mathbf{g}(\mathbf{x}_{s,a_s};\mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)}). \qquad \text{(A.5)}$$

Then the second term in (A.4) can be bounded in the following way:

$$\begin{aligned}
&-\big(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\big)^\top\big[\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) - \boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\big]\\
&= -\big(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^* - \mathbf{M}_{t-1}\big)^\top\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) + \big(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^* - \mathbf{M}_{t-1}\big)^\top\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\\
&\quad - \mathbf{M}_{t-1}^\top\big[\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) - \boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\big]\\
&\le \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^* - \mathbf{M}_{t-1}\|_{\mathbf{A}_{t-1}}\cdot\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} + \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^* - \mathbf{M}_{t-1}\|_{\mathbf{A}_{t-1}}\cdot\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}}\\
&\quad + \big\|\mathbf{M}_{t-1}^\top\big[\boldsymbol{\phi}\big(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1}\big) - \boldsymbol{\phi}\big(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1}\big)\big]\big\|_2\\
&\le \alpha_t\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*};\mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} + \alpha_t\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}}
\end{aligned}$$

$$+ \|\mathbf{M}_{t-1}\|_2 \cdot \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_2. \tag{A.6}$$

where the last inequality is due to Lemma A.4 and the choice of α_t. Plugging (A.6) back into (A.4) yields

$$\begin{aligned}
r(\mathbf{x}_{t,a_t^*}) - r(\mathbf{x}_{t,a_t}) \leq{}& \alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} - \alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} \\
&+ \alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} + \alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} \\
&+ \|\mathbf{M}_{t-1}\|_2 \cdot \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_2 \\
&+ \|\boldsymbol{\theta}_0\|_2 \cdot \|\mathbf{g}(\mathbf{x}_{t,a_t^*}; \mathbf{w}^{(0)}) - \mathbf{g}(\mathbf{x}_{t,a_t}; \mathbf{w}^{(0)})\|_F \cdot \|\mathbf{w}^* - \mathbf{w}^{(0)}\|_2 \\
\leq{}& 2\alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} + \|\mathbf{M}_{t-1}\|_2 \cdot \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_2 \\
&+ \ell_{\mathrm{Lip}} \|\boldsymbol{\theta}_0\|_2 \cdot \|\mathbf{x}_{t,a_t^*} - \mathbf{x}_{t,a_t}\|_2 \cdot \|\mathbf{w}^* - \mathbf{w}^{(0)}\|_2,
\end{aligned} \tag{A.7}$$

where in the first inequality we used the definition of upper confidence bound in Algorithm 1 and the second inequality is due to Assumption 4.2. Recall the linearization of ϕ_j in Lemma A.3, we have

$$\widehat{\boldsymbol{\phi}}(\mathbf{x}; \mathbf{w}_{t-1}) = \boldsymbol{\phi}(\mathbf{x}; \mathbf{w}_0) + \mathbf{g}(\mathbf{x}; \mathbf{w}_0)(\mathbf{w}_{t-1} - \mathbf{w}_0).$$

Note that by the initialization, we have $\boldsymbol{\phi}(\mathbf{x}; \mathbf{w}_0) = \mathbf{0}$ for any $\mathbf{x} \in \mathbb{R}^d$. Thus, it holds that

$$\begin{aligned}
\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1}) &= \boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_0) + \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_0) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1}) \\
&= \boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \widehat{\boldsymbol{\phi}}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) + \mathbf{g}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_0)(\mathbf{w}_{t-1} - \mathbf{w}_0) \\
&\quad + \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1}) - \mathbf{g}(\mathbf{x}_{t,a_t}; \mathbf{w}_0)(\mathbf{w}_{t-1} - \mathbf{w}_0),
\end{aligned} \tag{A.8}$$

which immediately implies that

$$\begin{aligned}
&\big\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\big\|_2 \\
&\leq \big\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \widehat{\boldsymbol{\phi}}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1})\big\|_2 + \big\|\boldsymbol{\phi}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1}) - \widehat{\boldsymbol{\phi}}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_{t-1})\big\|_2 \\
&\quad + \big\|\big(\mathbf{g}(\mathbf{x}_{t,a_t^*}; \mathbf{w}_0) - \mathbf{g}(\mathbf{x}_{t,a_t}; \mathbf{w}_0)\big)(\mathbf{w}_{t-1} - \mathbf{w}_0)\big\|_2 \\
&\leq C_0 \omega^{4/3} L^3 d^{1/2} \sqrt{m \log m} + \ell_{\mathrm{Lip}} \|\mathbf{x}_{t,a_t^*} - \mathbf{x}_{t,a_t}\|_2 \|\mathbf{w}_{t-1} - \mathbf{w}^{(0)}\|_2,
\end{aligned} \tag{A.9}$$

where the second inequality is due to Lemma A.3 and Assumption 4.2. Therefore, the instaneous regret can be further upper bounded as follows.

$$\begin{aligned}
r(\mathbf{x}_{t,a_t^*}) - r(\mathbf{x}_{t,a_t}) \leq{}& 2\alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} + \ell_{\mathrm{Lip}} \|\boldsymbol{\theta}_0\|_2 \cdot \|\mathbf{x}_{t,a_t^*} - \mathbf{x}_{t,a_t}\|_2 \cdot \|\mathbf{w}^* - \mathbf{w}^{(0)}\|_2 \\
&+ \|\mathbf{M}_{t-1}\|_2 \cdot \big(C_0 \omega^{4/3} L^3 d^{1/2} \sqrt{m \log m} + \ell_{\mathrm{Lip}} \|\mathbf{x}_{t,a_t^*} - \mathbf{x}_{t,a_t}\|_2 \|\mathbf{w}_{t-1} - \mathbf{w}^{(0)}\|_2\big).
\end{aligned} \tag{A.10}$$

By Assumption 4.1 we have $\|\mathbf{x}_{t,a_t^*} - \mathbf{x}_{t,a_t}\|_2 \leq 2$. By Lemma A.1 and Lemma A.2, we have

$$\|\mathbf{w}^* - \mathbf{w}^{(0)}\|_2 \leq \sqrt{1/m(\mathbf{r}-\widetilde{\mathbf{r}})^\top \mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})}, \quad \|\mathbf{w}_t - \mathbf{w}^{(0)}\|_2 \leq \frac{\delta^{3/2}}{m^{1/2}Tn^{9/2}L^6\log^3(m)}. \tag{A.11}$$

In addition, since the entries of $\boldsymbol{\theta}_0$ are i.i.d. generated from $N(0, 1/d)$, we have $\|\boldsymbol{\theta}_0\|_2 \leq 2(2 + \sqrt{d^{-1}\log(1/\delta)})$ with probability at least $1-\delta$ for any $\delta > 0$. By Lemma A.2, we have $\|\mathbf{g}(\mathbf{x}_{t,a_t}; \mathbf{w}^{(0)})\|_F \leq C_1\sqrt{dm}$. Therefore,

$$\big|\boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{s,a_s}; \mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)})\big| \leq C_2 d\sqrt{\log(1/\delta)(\mathbf{r}-\widetilde{\mathbf{r}})^\top \mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})}.$$

Then, by the definition of $\mathbf{M}_t$ in (A.5) and Lemma A.5, we have

$$\|\mathbf{M}_{t-1}\|_2 \leq C_3 d^2\sqrt{\log(1/\delta)(\mathbf{r}-\widetilde{\mathbf{r}})^\top \mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})}. \tag{A.12}$$

Substituting (A.12) and the above results on $\|\mathbf{x}_{t,a_t} - \mathbf{x}_{t,a_t^*}\|_2$, $\|\boldsymbol{\theta}_0\|_2$, $\|\mathbf{w}^* - \mathbf{w}^{(0)}\|_2$ and $\|\mathbf{w}_{t-1} - \mathbf{w}^{(0)}\|_2$ back into (A.10) further yields

$$\begin{aligned}
&r(\mathbf{x}_{t,a_t^*}) - r(\mathbf{x}_{t,a_t}) \\
&\leq 2\alpha_t \|\boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1})\|_{\mathbf{A}_{t-1}^{-1}} + C_4\ell_{\mathrm{Lip}} m^{-1/2}\sqrt{\log(1/\delta)(\mathbf{r}-\widetilde{\mathbf{r}})^\top \mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})} \\
&\quad + \bigg(C_0\omega^{4/3}L^3 d^{1/2}\sqrt{m\log m} + \frac{2\ell_{\mathrm{Lip}}\delta^{3/2}}{m^{1/2}Tn^{9/2}L^6\log^3(m)}\bigg) C_3 d^2\sqrt{\log(1/\delta)(\mathbf{r}-\widetilde{\mathbf{r}})^\top \mathbf{H}^{-1}(\mathbf{r}-\widetilde{\mathbf{r}})}.
\end{aligned}$$

Note that we have $\omega = O(m^{-1/2}\|\mathbf{r} - \widetilde{\mathbf{r}}\|_{\mathbf{H}^{-1}})$ by Lemma A.1. Therefore, the regret of the Neural-LinUCB is

$$\begin{aligned}
R_T &\leq \sqrt{QH \max_{t\in[T]} \alpha_t^2 \sum_{q=1}^{Q}\sum_{i=1}^{H} \|\boldsymbol{\phi}(\mathbf{x}_{i,a_i}; \mathbf{w}_{qH+i})\|^2_{\mathbf{A}_i^{-1}}} + C_4\ell_{\mathrm{Lip}} m^{-1/2} T\sqrt{\log(1/\delta)}\|\mathbf{r} - \widetilde{\mathbf{r}}\|_{\mathbf{H}^{-1}} \\
&\quad + \bigg(\frac{C_0 TL^3 d^{1/2}\sqrt{\log m}\|\mathbf{r} - \widetilde{\mathbf{r}}\|^{4/3}_{\mathbf{H}^{-1}}}{m^{1/6}} + \frac{2\ell_{\mathrm{Lip}}\delta^{3/2}}{m^{1/2}n^{9/2}L^6\log^3(m)}\bigg) C_3 d^2\sqrt{\log(1/\delta)}\|\mathbf{r} - \widetilde{\mathbf{r}}\|_{\mathbf{H}^{-1}} \\
&\leq C_5\sqrt{Td\log(1 + TG^2/(\lambda d))}\big(\nu\sqrt{d\log(1 + T(\log TK)/\lambda) + \log 1/\delta} + \lambda^{1/2}M\big) \\
&\quad + C_6\ell_{\mathrm{Lip}} L^3 d^{5/2} m^{-1/6} T\sqrt{\log m\log(1/\delta)\log(TK/\delta)}\|\mathbf{r} - \widetilde{\mathbf{r}}\|_{\mathbf{H}^{-1}},
\end{aligned}$$

where the first inequality is due to Cauchy's inequality, the second inequality comes from the upper bound of α_t in Lemma A.4 and Lemma A.6. $\{C_j\}_{j=0,\ldots,6}$ are absolute constants that are independent of problem parameters. □

B Proof of Technical Lemmas

In this section, we provide the proof of technical lemmas used in the regret analysis of Algorithm 1.

B.1 Proof of Lemma A.1

Before we prove the lemma, we first present some notations and a supporting lemma for simplification. Let $\boldsymbol{\beta} = (\boldsymbol{\theta}^\top, \mathbf{w}^\top)^\top \in \mathbb{R}^{d+p}$ be the concatenation of the exploration parameter and the hidden layer parameter of the neural network $f(\mathbf{x}; \boldsymbol{\beta}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}; \mathbf{w})$. Note that for any input data vector $\mathbf{x} \in \mathbb{R}^d$, we have

$$\frac{\partial}{\partial \boldsymbol{\beta}} f(\mathbf{x}; \boldsymbol{\beta}) = \bigg(\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})^\top, \boldsymbol{\theta}^\top \frac{\partial}{\partial \mathbf{w}} \boldsymbol{\phi}(\mathbf{x}; \mathbf{w})\bigg)^\top = \big(\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})^\top, \boldsymbol{\theta}^\top \mathbf{g}(\mathbf{x}; \mathbf{w})\big)^\top, \tag{B.1}$$

where $\mathbf{g}(\mathbf{x}; \mathbf{w})$ is the partial gradient of $\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})$ with respect to $\mathbf{w}$ defined in (A.1), which is a matrix in $\mathbb{R}^{d \times p}$. Similar to (4.2), we define $\mathbf{H}_{L+1}$ to be the neural tangent kernel matrix based on all $L+1$ layers of the neural network $f(\mathbf{x}; \boldsymbol{\beta})$. Note that by the definition of $\mathbf{H}$ in (4.2), we must have $\mathbf{H}_{L+1} = \mathbf{H} + \mathbf{B}$ for some positive definite matrix $\mathbf{B} \in \mathbb{R}^{TK \times TK}$. The following lemma shows that the NTK matrix is close to the matrix defined based on the gradients of the neural network on TK data points.

Lemma B.1 (Theorem 3.1 in Arora et al. (2019b))**.** Let $\epsilon > 0$ and $\delta \in (0, 1)$. Suppose the activation function in (2.2) is ReLU, i.e., $\sigma_l(x) = \max(0, x)$, and the width of the neural network satisfies

$$m \geq \boldsymbol{\Omega}\bigg(\frac{L^{14}}{\epsilon^4} \log\bigg(\frac{L}{\delta}\bigg)\bigg). \tag{B.2}$$

Then for any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ with $\|\mathbf{x}\|_2 = \|\mathbf{x}'\|_2 = 1$, with probability at least $1 - \delta$ over the randomness of the initialization of the network weight $\mathbf{w}$ it holds that

$$\bigg|\bigg\langle \frac{1}{\sqrt{m}} \frac{\partial f(\boldsymbol{\beta}, \mathbf{x})}{\partial \boldsymbol{\beta}}, \frac{1}{\sqrt{m}} \frac{\partial f(\boldsymbol{\beta}, \mathbf{x}')}{\partial \boldsymbol{\beta}} \bigg\rangle - \mathbf{H}_{L+1}(\mathbf{x}, \mathbf{x}')\bigg| \leq \epsilon.$$

Note that in the above lemma, there is a factor $1/\sqrt{m}$ before the gradient. This is due to the additional $\sqrt{m}$ factor in the definition of the neural network in (2.2), which ensures the value of the neural network function evaluated at the initialization is of the order $O(1)$.

Proof of Lemma A.1. Recall that we renumbered the feature vectors $\{\mathbf{x}_{t,k}\}_{t \in [T], k \in [K]}$ for all arms from round 1 to round T as $\{\mathbf{x}_i\}_{i=1,\ldots,TK}$. By concatenating the gradients at different inputs and the gradient in (B.1), we define $\boldsymbol{\Psi} \in \mathbb{R}^{TK \times (d+p)}$ as follows.

$$\boldsymbol{\Psi} = \frac{1}{\sqrt{m}} \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\beta}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_1; \mathbf{w}) \\ \vdots \\ \frac{\partial}{\partial \boldsymbol{\beta}} \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x}_{TK}; \mathbf{w}) \end{bmatrix} = \frac{1}{\sqrt{m}} \begin{bmatrix} \boldsymbol{\phi}(\mathbf{x}_1; \mathbf{w}^{(0)})^\top & \boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_1; \mathbf{w}^{(0)}) \\ \vdots & \vdots \\ \boldsymbol{\phi}(\mathbf{x}_i; \mathbf{w}^{(0)})^\top & \boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_i; \mathbf{w}^{(0)}) \\ \vdots & \vdots \\ \boldsymbol{\phi}(\mathbf{x}_{TK}; \mathbf{w}^{(0)})^\top & \boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{TK}; \mathbf{w}^{(0)}) \end{bmatrix}.$$

By Applying Lemma B.1, we know with probability at least $1 - \delta$ it holds that

$$|\langle \boldsymbol{\Psi}_{j*}, \boldsymbol{\Psi}_{l*} \rangle - \mathbf{H}_{L+1}(\mathbf{x}_j, \mathbf{x}_l)| \leq \epsilon$$

for any $\epsilon > 0$ as long as the width m satisfies the condition in (B.2). By applying union bound over all data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_{TK}\}$, we further have

$$\|\boldsymbol{\Psi}\boldsymbol{\Psi}^\top - \mathbf{H}_{L+1}\|_F \leq TK\epsilon.$$

Note that $\mathbf{H}$ is the neural tangent kernel (NTK) matrix defined in (4.2) and $\mathbf{H}_{L+1}$ is the NTK matrix defined based on all $L+1$ layers. By Assumption 4.3, $\mathbf{H}$ has a minimum eigenvalue $\lambda_0 > 0$, which is defined based on the first L layers of f. Furthermore, by the definition of NTK matrix in (4.2), we know that $\mathbf{H}_{L+1} = \mathbf{H} + \mathbf{B}$ for some semi-positive definite matrix $\mathbf{B}$. Therefore, the NTK matrix $\mathbf{H}_{L+1}$ defined based on all $L+1$ layers is also positive definite and its minimum eigenvalue is lower bounded by λ_0. Let $\epsilon = \lambda_0/(2TK)$. By triangle equality we have $\boldsymbol{\Psi}\boldsymbol{\Psi}^\top \succ \mathbf{H}_{L+1} - \|\boldsymbol{\Psi}\boldsymbol{\Psi}^\top - \mathbf{H}_{L+1}\|_2\mathbf{I} \succ \mathbf{H}_{L+1} - \|\boldsymbol{\Psi}\boldsymbol{\Psi}^\top - \mathbf{H}_{L+1}\|_F\mathbf{I} \succ \mathbf{H}_{L+1} - \lambda_0/2\mathbf{I} \succ 1/2\mathbf{H}_{L+1}$, which means that $\boldsymbol{\Psi}$ is semi-definite positive and thus $\text{rank}(\boldsymbol{\Psi}) = TK$ since $m > TK$.

We assume that $\boldsymbol{\Psi}$ can be decomposed as $\boldsymbol{\Psi} = \mathbf{P}\mathbf{D}\mathbf{Q}^\top$, where $\mathbf{P} \in \mathbb{R}^{TK\times TK}$ is the eigenvectors of $\boldsymbol{\Psi}\boldsymbol{\Psi}^\top$ and thus $\mathbf{P}\mathbf{P}^\top = \mathbf{I}_{TK}$, $\mathbf{D} \in \mathbb{R}^{TK\times TK}$ is a diagonal matrix with the square root of eigenvalues of $\boldsymbol{\Psi}\boldsymbol{\Psi}^\top$, and $\mathbf{Q}^\top \in \mathbb{R}^{TK\times(d+p)}$ is the eigenvectors of $\boldsymbol{\Psi}^\top\boldsymbol{\Psi}$ and thus $\mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_{TK}$. We use $\mathbf{Q}_1 \in \mathbb{R}^{d\times TK}$ and $\mathbf{Q}_2 \in \mathbb{R}^{p\times TK}$ to denote the two blocks of $\mathbf{Q}$ such that $\mathbf{Q}^\top = [\mathbf{Q}_1^\top, \mathbf{Q}_2^\top]$. By definition, we have

$$\mathbf{Q}^\top\mathbf{Q} = [\mathbf{Q}_1^\top, \mathbf{Q}_2^\top]\begin{bmatrix}\mathbf{Q}_1\\ \mathbf{Q}_2\end{bmatrix} = \mathbf{Q}_1^\top\mathbf{Q}_1 + \mathbf{Q}_2^\top\mathbf{Q}_2 = \mathbf{I}_{TK}.$$

Note that the minimum singular value of $\mathbf{Q}_1 \in \mathbb{R}^{d\times TK}$ is zero since d is a fixed number and $TK > d$. Therefore, it must hold that $\text{rank}(\mathbf{Q}_2) = TK$ and thus $\mathbf{Q}_2^\top\mathbf{Q}_2$ is positive definite. Let $\mathbf{r} = (r(\mathbf{x}_1), \ldots, r(\mathbf{x}_i), \ldots, r(\mathbf{x}_{TK}))^\top \in \mathbb{R}^{TK}$ denote the vector of all possible rewards. We further define $\mathbf{G} \in \mathbb{R}^{TKd\times p}$ and $\boldsymbol{\Phi} \in \mathbb{R}^{TKd}$ as follows

$$\mathbf{G} = \frac{1}{\sqrt{m}}\begin{bmatrix}\mathbf{g}(\mathbf{x}_1; \mathbf{w}^{(0)})\\ \vdots\\ \mathbf{g}(\mathbf{x}_i; \mathbf{w}^{(0)})\\ \vdots\\ \mathbf{g}(\mathbf{x}_{TK}; \mathbf{w}^{(0)})\end{bmatrix}, \quad \boldsymbol{\Phi} = \begin{bmatrix}\boldsymbol{\phi}(\mathbf{x}_{1,1}; \mathbf{w}_0)\\ \vdots\\ \boldsymbol{\phi}(\mathbf{x}_{t,k}; \mathbf{w}_{t-1})\\ \vdots\\ \boldsymbol{\phi}(\mathbf{x}_{T,K}; \mathbf{w}_{T-1})\end{bmatrix}. \tag{B.3}$$

and $\boldsymbol{\Theta}, \boldsymbol{\Theta}_0 \in \mathbb{R}^{TK\times TKd}$ as follows

$$\boldsymbol{\Theta}^* = \begin{bmatrix}\boldsymbol{\theta}^{*\top} & & & & \\ & \ddots & & & \\ & & \boldsymbol{\theta}^{*\top} & & \\ & & & \ddots & \\ & & & & \boldsymbol{\theta}^{*\top}\end{bmatrix}, \quad \boldsymbol{\Theta}_0 = \begin{bmatrix}\boldsymbol{\theta}_0^\top & & & & \\ & \ddots & & & \\ & & \boldsymbol{\theta}_0^\top & & \\ & & & \ddots & \\ & & & & \boldsymbol{\theta}_0^\top\end{bmatrix}, \tag{B.4}$$

It can be verified that $\boldsymbol{\Psi} = \mathbf{P}\mathbf{D}[\mathbf{Q}_1^\top, \mathbf{Q}_2^\top]$ and $\mathbf{P}\mathbf{D}\mathbf{Q}_2^\top = \boldsymbol{\Theta}_0\mathbf{G}$. Note that we have $\mathbf{Q}_2^\top\mathbf{Q}_2$ is positive definite by Assumption 4.3, which corresponds to the neural tangent kernel matrix defined on the

first L layers. Then we can define $\mathbf{w}^*$ as follows

$$\mathbf{w}^* = \mathbf{w}^{(0)} + 1/\sqrt{m}\mathbf{Q}_2(\mathbf{Q}_2^\top\mathbf{Q}_2)^{-1}\mathbf{D}^{-1}\mathbf{P}^\top(\mathbf{r} - \boldsymbol{\Theta}^*\boldsymbol{\Phi}). \tag{B.5}$$

We can verify that

$$\boldsymbol{\Theta}^*\boldsymbol{\Phi} + \sqrt{m}\mathbf{P}\mathbf{D}\mathbf{Q}_2^\top(\mathbf{w}^* - \mathbf{w}^{(0)}) = \mathbf{r}.$$

On the other hand, we have

$$\begin{aligned}\|\mathbf{w}^* - \mathbf{w}^{(0)}\|_2^2 &\leq 1/m(\mathbf{r} - \boldsymbol{\Theta}^*\boldsymbol{\Phi})^\top\mathbf{P}\mathbf{D}^{-1}(\mathbf{Q}_2^\top\mathbf{Q}_2)^{-1}\mathbf{D}^{-1}\mathbf{P}^\top(\mathbf{r} - \boldsymbol{\Theta}^*\boldsymbol{\Phi}) \\ &\leq 1/m(\mathbf{r} - \boldsymbol{\Theta}^*\boldsymbol{\Phi})^\top\mathbf{H}^{-1}(\mathbf{r} - \boldsymbol{\Theta}^*\boldsymbol{\Phi}),\end{aligned}$$

which completes the proof. □

B.2 Proof of Lemma A.2

Note that we can view the output of the last hidden layer $\boldsymbol{\phi}(\mathbf{x}; \mathbf{w})$ defined in (2.3) as a vector-output neural network with weight parameter $\mathbf{w}$. The following lemma shows that the output of the neural network $\boldsymbol{\phi}$ is bounded at the initialization.

Lemma B.2 (Lemma 4.4 in Cao and Gu (2019a))**.** Let $\delta \in (0, 1)$, and the width of the neural network satisfy $m \geq C_0 L \log(TKL/\delta)$. Then for all $t \in [T]$, $k \in [K]$ and $j \in [d]$, we have $|\phi_j(\mathbf{x}_{t,k}; \mathbf{w}^{(0)})| \leq C_1\sqrt{\log(TK/\delta)}$ with probability at least $1 - \delta$, where $\mathbf{w}^{(0)}$ is the initialization of the neural network.

In addition, in a smaller neighborhood of the initialization, the gradient of the neural network $\boldsymbol{\phi}$ is uniformly bounded.

Lemma B.3 (Lemma B.3 in Cao and Gu (2019a))**.** Let $\omega \leq C_0 L^{-6}(\log m)^{-3}$ and $\mathbf{w} \in \mathbb{B}(\mathbf{w}_0, \omega)$. Then for all $t \in [T]$, $k \in [K]$ and $j \in [d]$, the gradient of the neural network $\boldsymbol{\phi}$ defined in (2.3) satisfies $\|\nabla_{\mathbf{w}}\phi_j(\mathbf{x}_{t,k}; \mathbf{w})\|_2 \leq C_1\sqrt{Lm}$ with probability at least $1 - TKL^2\exp(-C_2 m\omega^{2/3}L)$.

The next lemma provides an upper bound on the gradient of the squared loss function defined in (3.5). Note that our definition of the loss function is slightly different from that in Allen-Zhu et al. (2019b) due to the output layer $\boldsymbol{\theta}_i$ and thus there is an additional term on the upper bound of $\|\boldsymbol{\theta}_i\|_2$ for all $i \in [T]$.

Lemma B.4 (Theorem 3 in Allen-Zhu et al. (2019b))**.** Let $\omega \leq C_0\delta^{3/2}/(T^{9/2}L^6\log^3 m)$. For all $\mathbf{w} \in \mathbb{B}(\mathbf{w}^{(0)}, \omega)$, with probability at least $1 - \exp(-C_1 m\omega^{2/3}L)$ over the randomness of $\mathbf{w}^{(0)}$, it holds that

$$\|\nabla\mathcal{L}(\mathbf{w})\|_2^2 \leq \frac{C_2 Tm\mathcal{L}(\mathbf{w})\sup_{i=1,\ldots,H}\|\boldsymbol{\theta}_i\|_2^2}{d}.$$

Proof of Lemma A.2. Fix the epoch number q and we omit it in the subscripts in the rest of the proof when no confusion arises. Recall that $\mathbf{w}^{(s)}$ is the s-th iterate in Algorithm 2. Let $\delta > 0$ be

any constant. Let ω be defined as follows.

$$\omega = \delta^{3/2} m^{-1/2} T^{-9/2} L^{-6} \log^{-3}(m). \tag{B.6}$$

We will prove by induction that with probability at least $1-\delta$ the following statement holds for all $s = 0, 1, \ldots, n$

$$\phi_j(\mathbf{x}; \mathbf{w}^{(s)}) \leq C_0 \sum_{h=0}^{s} \frac{\sqrt{\log(TK/\delta)}}{h+1}, \quad \text{for } \forall j \in [d]; \text{ and } \|\mathbf{w}_q^{(s)} - \mathbf{w}^{(0)}\| \leq \omega. \tag{B.7}$$

First note that (B.7) holds trivially when $s = 0$ due to Lemma B.2. Now we assume that (B.7) holds for all $j = 0, \ldots, s$. The loss function in (3.5) can be bounded as follows.

$$\mathcal{L}(\mathbf{w}^{(j)}) = \sum_{i=1}^{qH} (\boldsymbol{\theta}_i^\top \boldsymbol{\phi}(\mathbf{x}_i; \mathbf{w}^{(j)}) - \widehat{r}_i)^2 \leq \sum_{i=1}^{qH} 2(\|\boldsymbol{\theta}_i\|_2^2 \cdot \|\boldsymbol{\phi}(\mathbf{x}_i; \mathbf{w}^{(j)})\|_2^2 + 1).$$

By the update rule of $\boldsymbol{\theta}_t$, we have

$$\|\boldsymbol{\theta}_t\|_2 = \bigg\| \bigg(\lambda \mathbf{I} + \sum_{i=1}^{t} \boldsymbol{\phi}(\mathbf{x}_i; \mathbf{w}_{i-1}) \boldsymbol{\phi}(\mathbf{x}_i; \mathbf{w}_{i-1})^\top \bigg)^{-1} \sum_{i=1}^{t} \boldsymbol{\phi}(\mathbf{x}_i; \mathbf{w}_{i-1}) \widehat{\mathbf{r}} \bigg\|_2 \leq 2d, \tag{B.8}$$

where the inequality is due to Lemma A.5, which combined with (B.7) immediately implies

$$\mathcal{L}(\mathbf{w}^{(j)}) \leq C_1 T d^3 \log(TK/\delta) \bigg(\sum_{h=0}^{j} \frac{1}{h+1} \bigg)^2 \leq C_1 T d^3 \log(TK/\delta) \log^2 n. \tag{B.9}$$

Substituting (B.8) and (B.9) into the inequality in Lemma B.4, we also have

$$\big\| \nabla \mathcal{L}\big(\mathbf{w}^{(j)}\big) \big\|_2 \leq C_2 \sqrt{dTm \mathcal{L}(\mathbf{w}^{(j)})} \leq C_3 d^2 T \log(n) \sqrt{m \log(TK/\delta)}. \tag{B.10}$$

Now we consider $\mathbf{w}^{(s+1)}$. By triangle inequality we have

$$\begin{aligned}
\big\| \mathbf{w}^{(s+1)} - \mathbf{w}^{(0)} \big\|_2 &\leq \sum_{j=0}^{s} \big\| \mathbf{w}^{(j+1)} - \mathbf{w}^{(j)} \big\|_2 \\
&= \sum_{j=0}^{s} \eta \big\| \nabla \mathcal{L}\big(\mathbf{w}^{(j)}\big) \big\|_2 \\
&\leq \sum_{j=0}^{s} \eta d^2 T \log(n) \sqrt{m \log(TK/\delta)},
\end{aligned} \tag{B.11}$$

where the last inequality is due to (B.10). If we choose the step size η_q in the q-th epoch such that

$$\eta \leq \frac{\omega}{d^2 T n \log(n) \sqrt{m \log(TK/\delta)}}, \tag{B.12}$$

then we have $\|\mathbf{w}_q^{(s+1)}-\mathbf{w}^{(0)}\|_2 \leq \omega$. Note that the choice of m, ω satisfies the condition in Lemma A.3. Thus we know $\phi_j(\mathbf{x};\mathbf{w})$ is almost linear in $\mathbf{w}$, which leads to

$$
\begin{aligned}
|\phi_j(\mathbf{x};\mathbf{w}^{(s+1)})| &\leq |\phi_j(\mathbf{x};\mathbf{w}^{(s)}) + \langle \nabla\phi_j(\mathbf{x};\mathbf{w}^{(s)}), \mathbf{w}^{(s+1)}-\mathbf{w}^{(s)}\rangle| + C_5\omega^{4/3}L^3 d^{-1/2}\sqrt{m\log m} \\
&\leq \sum_{h=0}^{s} \frac{C\sqrt{\log(TK/\delta)}}{h+1} + \eta\sqrt{dm}\|\nabla\mathcal{L}(\mathbf{w}^{(s)})\|_2 + 2C_5\omega^{4/3}L^3 d^{-1/2}\sqrt{m\log m} \\
&\leq \sum_{h=0}^{s} \frac{C_0\sqrt{\log(TK/\delta)}}{h+1} + C_3\eta\sqrt{dm}\sqrt{CT^2d^4m\log(TK/\delta)}\log n \\
&\quad + 2C_5\omega^{4/3}L^3 d^{-1/2}\sqrt{m\log m} \\
&= \sum_{h=0}^{s} \frac{C_0\sqrt{\log(TK/\delta)}}{h+1} + \frac{\omega\sqrt{dm}}{n} + 2C_5\omega^{4/3}L^3 d^{-1/2}\sqrt{m\log m}, \qquad \text{(B.13)}
\end{aligned}
$$

where in the second inequality we used the induction hypothesis (B.7), Cauchy-Schwarz inequality and Lemma B.3, and the third inequality is due to (B.10). Note that the definition of ω in (B.6) ensures that $\omega\sqrt{dm} < 1/2$ and $\omega^{4/3}L^3d^{-1/2}\sqrt{m\log m} \leq m^{-1/6}T^{-6}L^{-5}d^{-1/2}\sqrt{\log m} \leq 1/n$ as long as $m \geq n^6$. Plugging these two upper bounds back into (B.13) finishes the proof of (B.7).

Note that for any $t \in [T]$, we have $\mathbf{w}_t = \mathbf{w}_q^{(n)}$ for some $q = 1, 2, \ldots$. Since we have $\mathbf{w}_t \in \mathbb{B}(\mathbf{w}, \omega)$, the gradient $\mathbf{g}(\mathbf{x};\mathbf{w}^{(0)})$ can be directly bounded by Lemma B.3, which implies $\|\mathbf{g}(\mathbf{x};\mathbf{w}^{(0)})\|_F \leq C_6\sqrt{dLm}$. Applying (B.7) with $s = n$, we have the following bound of the neural network function $\boldsymbol{\phi}(\mathbf{x};\mathbf{w}_q^{(n)}) = \boldsymbol{\phi}(\mathbf{x};\mathbf{w}_t)$ for all t in the q-th epoch

$$
\|\boldsymbol{\phi}(\mathbf{x};\mathbf{w}_t)\|_2 \leq C_0\sqrt{d\log(n)\log(TK/\delta)},
$$

which completes the proof. In this proof, $\{C_j > 0\}_{j=0,\ldots,6}$ are constants independent of problem parameters. □

B.3 Proof of Lemma A.4

The following lemma characterizes the concentration property of self-normalized martingales.

Lemma B.5 (Theorem 1 in Abbasi-Yadkori et al. (2011))**.** Let $\{\xi\}_{t=1}^{\infty}$ be a real-valued stochastic process and $\{\mathbf{x}_t\}_{t=1}^{\infty}$ be a stochastic process in $\mathbb{R}^d$. Let $\mathcal{F}_t = \sigma(\mathbf{x}_1, \ldots, \mathbf{x}_{t+1}, \xi-1, \ldots, \xi_t)$ be a σ-algebra such that $\mathbf{x}_t$ and ξ_t are $\mathcal{F}_{t-1}$-measurable. Let $\mathbf{A}_t = \lambda\mathbf{I} + \sum_{s=1}^{t}\mathbf{x}_s\mathbf{x}_s^\top$ for some constant $\lambda > 0$ and $S_t = \sum_{s=1}^{t}\xi_s\mathbf{x}_i$. If we assume ξ_t is ν-subGaussian conditional on $\mathcal{F}_{t-1}$, then for any $\eta \in (0,1)$, with probability at least $1-\delta$, we have

$$
\|S_t\|_{\mathbf{A}_t^{-1}}^2 \leq 2\nu^2\log\bigg(\frac{\det(\mathbf{A}_t)^{1/2}\det(\lambda\mathbf{I})^{-1/2}}{\delta}\bigg).
$$

Proof of Lemma A.4. Let $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\mathbf{x}_{1,a_1};\mathbf{w}_0), \ldots, \boldsymbol{\phi}(\mathbf{x}_{t,a_t};\mathbf{w}_{t-1})] \in \mathbb{R}^{d\times t}$ be the collection of feature vectors of the chosen arms up to time t and $\widehat{\mathbf{r}}_t = (\widehat{r}_1, \ldots, \widehat{r}_t)^\top$ be the concatenation of all received rewards. According to Algorithm 1, we have $\mathbf{A}_t = \lambda\mathbf{I} + \boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top$ and thus

$$
\boldsymbol{\theta}_t = \mathbf{A}_t^{-1}\mathbf{b}_t = (\lambda\mathbf{I} + \boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top)^{-1}\boldsymbol{\Phi}_t\widehat{\mathbf{r}}_t.
$$

By Lemma A.1, the underlying reward generating function $r_t = r(\mathbf{x}_{t,a_t}) = \mathbb{E}[\widehat{r}(\mathbf{x}_{t,a_t})|\mathbf{x}_{t,a_t}]$ can be rewritten as

$$r_t = \langle \boldsymbol{\theta}^*, \boldsymbol{\phi}(\mathbf{x}_{t,a_t}; \mathbf{w}_{t-1}) \rangle + \boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{t,a_t}; \mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)}).$$

By the definition of the reward in (3.3) we have $\widehat{r}_t = r_t + \xi_t$. Therefore, it holds that

$$\begin{aligned}
\boldsymbol{\theta}_t &= \mathbf{A}_t^{-1}\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top\boldsymbol{\theta}^* + \mathbf{A}_t^{-1}\sum_{s=1}^{t}\boldsymbol{\phi}(\mathbf{x}_{s,a_s}; \mathbf{w}_{s-1})(\boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{s,a_s}; \mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)}) + \xi_s) \\
&= \boldsymbol{\theta}^* - \lambda\mathbf{A}_t^{-1}\boldsymbol{\theta}^* + \mathbf{A}_t^{-1}\sum_{s=1}^{t}\boldsymbol{\phi}(\mathbf{x}_{s,a_s}; \mathbf{w}_{s-1})(\boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{s,a_s}; \mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)}) + \xi_s).
\end{aligned}$$

Note that $\mathbf{A}_t$ is positive definite as long as $\lambda > 0$. Therefore $\|\cdot\|_{\mathbf{A}_t}$ and $\|\cdot\|_{\mathbf{A}_t}$ are well defined norms. Then for any $\delta \in (0, 1)$ by triangle inequality we have

$$\begin{aligned}
\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^* - \mathbf{A}_t^{-1}\boldsymbol{\Phi}_t\boldsymbol{\Theta}_t\mathbf{G}_t(\mathbf{w}^* - \mathbf{w}^{(0)})\|_{\mathbf{A}_t} &\leq \lambda\|\boldsymbol{\theta}^*\|_{\mathbf{A}_t^{-1}} + \|\boldsymbol{\Phi}_t\boldsymbol{\xi}_t\|_{\mathbf{A}_t^{-1}} \\
&\leq \nu\sqrt{2\log\bigg(\frac{\det(\mathbf{A}_t)^{1/2}\det(\lambda\mathbf{I})^{-1/2}}{\delta}\bigg)} + \lambda^{1/2}M
\end{aligned}$$

holds with probability at least $1 - \delta$, where in the last inequality we used Lemma B.5 and the fact that $\|\boldsymbol{\theta}^*\|_{\mathbf{A}_t^{-1}} \leq \lambda^{-1/2}\|\boldsymbol{\theta}^*\|_2 \leq \lambda^{-1/2}M$ by Lemma A.1. Plugging the definition of $\boldsymbol{\Phi}_t, \boldsymbol{\Theta}_t$ and $\mathbf{G}_t$ and apply Lemma A.6, we further have

$$\begin{aligned}
&\bigg\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^* - \mathbf{A}_t^{-1}\sum_{s=1}^{t}\boldsymbol{\phi}(\mathbf{x}_{s,a_s}; \mathbf{w}_{s-1})\boldsymbol{\theta}_0^\top \mathbf{g}(\mathbf{x}_{s,a_s}; \mathbf{w}^{(0)})(\mathbf{w}^* - \mathbf{w}^{(0)})\bigg\|_{\mathbf{A}_t} \\
&\leq \nu\sqrt{2\big(d\log(1 + t(\log HK)/\lambda) + \log 1/\delta\big)} + \lambda^{1/2}M,
\end{aligned}$$

where we used the fact that $\|\phi(\mathbf{x}; \mathbf{w})\|_2 \leq C\sqrt{d\log HK}$ by Lemma A.2. □

B.4 Proof of Lemma A.5

We now prove the technical lemma that upper bounds $\|\mathbf{A}_t^{-1}\sum_{s=1}^{t}\boldsymbol{\phi}_s\zeta_s\|_2$.

Proof of Lemma A.5. We first construct auxiliary vectors $\widetilde{\boldsymbol{\phi}}_t \in \mathbb{R}^{d+1}$ and matrices $\mathbf{B}_t \in \mathbb{R}^{(d+1)\times(d+1)}$ for all $t = 1, \ldots$ in the following way:

$$\widetilde{\boldsymbol{\phi}}_t = \begin{bmatrix} G^{-1}\boldsymbol{\phi}_t \\ \sqrt{1 - G^{-2}\|\boldsymbol{\phi}_t\|_2^2} \end{bmatrix}, \quad \mathbf{B}_t = \begin{bmatrix} \mathbf{A}_t^{-1} & \mathbf{0}_d \\ \mathbf{0}_d^\top & 0 \end{bmatrix}, \tag{B.14}$$

where $\mathbf{0}_d \in \mathbb{R}^d$ is an all-zero vector. Then by definition we immediately have

$$\bigg\|\mathbf{A}_t^{-1}\sum_{s=1}^{t}\boldsymbol{\phi}_s\zeta_s\bigg\|_2 = \bigg\|\mathbf{B}_t\sum_{s=1}^{t}\widetilde{\boldsymbol{\phi}}_s\zeta_s\bigg\|_2. \tag{B.15}$$

For all $s = 1, 2 \ldots$, let $\{\beta_{s,j}\}_{j=1}^{d+1}$ be the coefficients of the decomposition of $U^{-1}\zeta_s\widetilde{\boldsymbol{\phi}}_s$ on the natural basis. Specifically, let $\{\mathbf{e}_1, \ldots, \mathbf{e}_{d+1}\}$ be the natural basis of $\mathbb{R}^{d+1}$ such that the entries of $\mathbf{e}_j$ are all zero except the j-th entry which equals 1. Then we have

$$U^{-1}\zeta_s\widetilde{\boldsymbol{\phi}}_s = \sum_{j=1}^{d} \beta_{s,j}\mathbf{e}_j, \quad \forall s = 1, 2, \ldots \tag{B.16}$$

We can conclude that $|\beta_{s,j}| \leq 1$ since $|\zeta_s| \leq U$ and $\|\widetilde{\boldsymbol{\phi}}_s\|_2 \leq 1$. Moreover, it is easy to verify that $\|\widetilde{\boldsymbol{\phi}}_t\|_2 = 1$ for all $t \geq 1$. Therefore, we have

$$\begin{aligned}
\left\|\mathbf{B}_t \sum_{s=1}^{t} \widetilde{\boldsymbol{\phi}}_s\zeta_s\right\|_2 &= \left\|\mathbf{B}_t \sum_{s=1}^{t} \widetilde{\boldsymbol{\phi}}_s\widetilde{\boldsymbol{\phi}}_s^\top\widetilde{\boldsymbol{\phi}}_s\zeta_s\right\|_2 \\
&= \left\|\mathbf{B}_t \sum_{s=1}^{t} \widetilde{\boldsymbol{\phi}}_s\widetilde{\boldsymbol{\phi}}_s^\top U\sum_{j=1}^{d}\beta_{s,j}\mathbf{e}_j\right\|_2 \\
&= U\left\|\sum_{j=1}^{d}\mathbf{B}_t \sum_{s=1}^{t} \widetilde{\boldsymbol{\phi}}_s\widetilde{\boldsymbol{\phi}}_s^\top\beta_{s,j}\mathbf{e}_j\right\|_2 \\
&\leq U\sum_{j=1}^{d}\left\|\mathbf{B}_t \sum_{s=1}^{t} \widetilde{\boldsymbol{\phi}}_s\widetilde{\boldsymbol{\phi}}_s^\top\beta_{s,j}\right\|_2 \\
&= U\sum_{j=1}^{d}\left\|\mathbf{A}_t^{-1} \sum_{s=1}^{t} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top\beta_{s,j}\right\|_2,
\end{aligned} \tag{B.17}$$

where the inequality is due to triangle inequality and the last equation is due to the definition of $\widetilde{\boldsymbol{\phi}}_t$ and $\mathbf{B}_t$ in (B.14). For each $j = 1, \ldots, d+1$, we have

$$\begin{aligned}
\left\|\mathbf{A}_t^{-1} \sum_{s=1}^{t} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top\beta_{s,j}\right\|_2 &= \left\|\mathbf{A}_t^{-1} \sum_{s\in[t]:\beta_{s,j}\geq 0} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top\beta_{s,j} + \mathbf{A}_t^{-1} \sum_{s\in[t]:\beta_{s,j}< 0} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top\beta_{s,j}\right\|_2 \\
&\leq \left\|\mathbf{A}_t^{-1} \sum_{s\in[t]:\beta_{s,j}\geq 0} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top\beta_{s,j}\right\|_2 + \left\|\mathbf{A}_t^{-1} \sum_{s\in[t]:\beta_{s,j}< 0} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top(-\beta_{s,j})\right\|_2.
\end{aligned} \tag{B.18}$$

Since we have $|\beta_{s,j}| \leq 1$, it immediately implies

$$\begin{aligned}
\mathbf{A}_t = \lambda\mathbf{I} + \sum_{s=1}^{t} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top &\succ \sum_{s\in[t]:\beta_{s,j}\geq 0} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top\beta_{s,j}, \\
\mathbf{A}_t = \lambda\mathbf{I} + \sum_{s=1}^{t} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top &\succ \sum_{s\in[t]:\beta_{s,j}< 0} \boldsymbol{\phi}_s\boldsymbol{\phi}_s^\top(-\beta_{s,j}).
\end{aligned}$$

Further by the fact that $\|\mathbf{A}^{-1}\mathbf{B}\|_2 \leq 1$ for any $\mathbf{A} \succ \mathbf{B} \succeq 0$, combining the above results with (B.18)

yields

$$\left\| \mathbf{A}_t^{-1} \sum_{s=1}^{t} \boldsymbol{\phi}_s \boldsymbol{\phi}_s^\top \beta_{s,j} \right\|_2 \leq 2.$$

Finally, substituting the above results into (B.17) and (B.15) we have

$$\left\| \mathbf{A}_t^{-1} \sum_{s=1}^{t} \boldsymbol{\phi}_s \zeta_s \right\|_2 \leq 2Ud,$$

which completes the proof. □

References

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

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L. and Schapire, R. (2014). Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*.

Allen-Zhu, Z., Li, Y. and Liang, Y. (2019a). Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*.

Allen-Zhu, Z., Li, Y. and Song, Z. (2019b). A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*.

Arora, S., Du, S., Hu, W., Li, Z. and Wang, R. (2019a). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*.

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

Audibert, J.-Y., Munos, R. and Szepesvári, C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science* **410** 1876–1902.

Auer, P., Cesa-Bianchi, N. and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning* **47** 235–256.

Balakrishnan, S., Wainwright, M. J., Yu, B. et al. (2017). Statistical guarantees for the em algorithm: From population to sample-based analysis. *The Annals of Statistics* **45** 77–120.

Cai, Q., Yang, Z., Lee, J. D. and Wang, Z. (2019). Neural temporal-difference learning converges to global optima. In *Advances in Neural Information Processing Systems*.

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

Cao, Y. and Gu, Q. (2019b). A generalization theory of gradient descent for learning over-parameterized deep relu networks. *arXiv preprint arXiv:1902.01384* .

Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*.

Chowdhury, S. R. and Gopalan, A. (2017). On kernelized multi-armed bandits. In *International Conference on Machine Learning*.

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

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* **2** 303–314.

Dani, V., Hayes, T. P. and Kakade, S. M. (2008). Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory*.

Deshmukh, A. A., Kumar, A., Boyles, L., Charles, D., Manavoglu, E. and Dogan, U. (2020). Self-supervised contextual bandits in computer vision. *arXiv preprint arXiv:2003.08485* .

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

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

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

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

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep learning*. MIT press.

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

Kveton, B., Zaheer, M., Szepesvari, C., Li, L., Ghavamzadeh, M. and Boutilier, C. (2020). Randomized exploration in generalized linear bandits. In *International Conference on Artificial Intelligence and Statistics*.

Langford, J. and Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

LeCun, Y., Bengio, Y. and Hinton, G. (2015). Deep learning. *nature* **521** 436–444.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J. and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*.

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

Li, L., Lu, Y. and Zhou, D. (2017). Provably optimal algorithms for generalized linear contextual bandits. In *International Conference on Machine Learning*.

Liu, B., Cai, Q., Yang, Z. and Wang, Z. (2019). Neural trust region/proximal policy optimization attains globally optimal policy. In *Advances in Neural Information Processing Systems*.

Oymak, S. and Soltanolkotabi, M. (2020). Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *IEEE Journal on Selected Areas in Information Theory* .

Riquelme, C., Tucker, G. and Snoek, J. (2018). Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*.
URL https://openreview.net/forum?id=SyYe6k-CW

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

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M. and Adams, R. (2015). Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* **25** 285–294.

Valko, M., Korda, N., Munos, R., Flaounas, I. and Cristianini, N. (2013). Finite-time analysis of kernelised contextual bandits. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*.

Wang, L., Cai, Q., Yang, Z. and Wang, Z. (2020). Neural policy gradient methods: Global optimality and rates of convergence. In *International Conference on Learning Representations*.
URL https://openreview.net/forum?id=BJgQfkSYDS

Wang, Z., Liu, H. and Zhang, T. (2014). Optimal computational and statistical rates of convergence for sparse nonconvex learning problems. *Annals of statistics* **42** 2164.

Xu, P. and Gu, Q. (2020). A finite-time analysis of q-learning with neural network function approximation. In *International Conference on Machine Learning*.

Xu, P., Ma, J. and Gu, Q. (2017). Speeding up latent variable gaussian graphical model estimation via nonconvex optimization. In *Advances in Neural Information Processing Systems*.

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

Zhang, W., Zhou, D., Li, L. and Gu, Q. (2020). Neural thompson sampling. *arXiv preprint arXiv:2010.00827* .

Zhou, D., Li, L. and Gu, Q. (2020). Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*.

Zou, D., Cao, Y., Zhou, D. and Gu, Q. (2018). Stochastic gradient descent optimizes over-parameterized deep relu networks. *arXiv preprint arXiv:1811.08888* .

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