Neural Combinatorial Clustered Bandits for Recommendation Systems

Baran Atalar, Carlee Joe-Wong

Carnegie Mellon University
batalar@andrew.cmu.edu, cjoewong@andrew.cmu.edu

Abstract

We consider the contextual combinatorial bandit setting where in each round, the learning agent, e.g., a recommender system, selects a subset of "arms," e.g., products, and observes rewards for both the individual base arms, which are a function of known features (called "context"), and the super arm (the subset of arms), which is a function of the base arm rewards. The agent's goal is to simultaneously learn the unknown reward functions and choose the highest-reward arms. For example, the "reward" may represent a user's probability of clicking on one of the recommended products. Conventional bandit models, however, employ restrictive reward function models in order to obtain performance guarantees. We make use of deep neural networks to estimate and learn the unknown reward functions and propose Neural UCB Clustering (NeUClust), which adopts a clustering approach to select the super arm in every round by exploiting underlying structure in the context space. Unlike prior neural bandit works, NeUClust uses a neural network to estimate the super arm reward and select the super arm, thus eliminating the need for a known optimization oracle. We nontrivially extend prior neural combinatorial bandit works to prove that NeUClust achieves sublinear regret in the number of rounds. Experiments on real recommendation datasets show that NeUClust achieves better regret and reward than other contextual combinatorial and neural bandit algorithms.

Extended version — https://arxiv.org/abs/2410.14586
Code — https://github.com/BaranAtalar/NeUClust

Introduction

The contextual combinatorial bandit (CC-MAB) setting has been applied to content recommendation such as movies (Qin, Chen, and Zhu 2014), where we recommend multiple movies to a user or select multiple users to whom a movie will be recommended. In this application, an agent is confronted with a set of choices, or "arms," e.g., a set of movies, each of which will yield an unknown reward that depends on a known context, e.g., movie genre. Over the course of several rounds, the agent must learn to select the "best" arms. We consider the usual semi-bandit feedback setting where in each round, the user receives the individual (base arm) rewards of the arms selected, which are functions of their contexts, as well as a total (super arm) reward, which is a function of the base arm rewards. In recommendation settings, where each base arm might represent a user to which a product is being recommended, the base arm reward would be the user's rating for that product and the super arm reward might indicate whether a threshold number of users rated the product highly. However, both the base arm and the super arm reward functions are stochastic and unknown to the user, and we aim to learn these reward functions in an online manner in order to select the arms which would yield the highest rewards for a given context. The main challenge in the CC-MAB and bandits setting in general is to *balance between exploiting arms that we already know to be good and exploring new arms* that might be better. A relatively new approach to the exploration-exploitation tradeoff has been to use neural networks to learn the base arm reward functions.

Drawbacks of existing formulations. Modeling reward functions with neural networks introduces greater expressibility into the rewards learned, but existing neural and non-neural combinatorial bandit works generally require the existence of an exact or approximation oracle (Hwang, Chai, and Oh 2023; Elahi et al. 2023; Chen, Wang, and Yuan 2013; Chen, Xu, and Lu 2018), which selects the super arm after being given as input the estimated reward/quality of each base arm. This oracle, however, may be difficult to derive a priori, before the super arm reward function is known. Indeed, many combinatorial bandit problems are NP-hard and may not even have known approximation algorithms (Kong et al. 2021), e.g., variants of knapsack problems. We **eliminate the need for an oracle** by leveraging the structure of recommendation problems. In particular, we recognize that *the context space of many recommendation problems has a clustered structure*. Thus, we can cluster arms by clustering their contexts and use the resulting groupings, aided by an additional neural network-based estimation of super arm rewards, to guide our search for the optimal super arm.

Challenges of our approach. To estimate the performance of the clusters, we need an expression for the super arm reward for the base arms in each cluster. We do so by again using a neural network to model the super arm reward and taking advantage of the recommendation problem structure: super arm reward functions in recommendation settings are often *monotonic*, since users are more likely to prefer a

group of recommended products if they like each individual product. Thus, we can utilize a monotonic neural network to estimate the super arm reward. Incorporating this structure into a theoretical regret bond, however, introduces new challenges: first, we must account for the effect of the error in clustering the context space on the regret. Second, by using a second neural network for the super arm selection instead of a solution oracle, we cannot use existing proof approaches that use an oracle (Hwang, Chai, and Oh 2023).

Related Work

Some works in the bandit literature adaptively cluster users (arms) such as (Gentile, Li, and Zappella 2014; Bui, Johari, and Mannor 2012) but their problem setting is not of a combinatorial nature and thus cannot be directly applied to our setting. Most commonly, prior works make certain **parametric assumptions** about the reward model to be learned such as linearity with respect to the context (Wen, Kveton, and Ashkan 2015; Zong et al. 2016; Chu et al. 2011). Towards more general reward models, some **discretization based approaches** use fixed or adaptive discretization (Chen, Xu, and Lu 2018; Nika, Elahi, and Tekin 2020) of the context space to exploit similarity of context features. However, such methods could scale poorly to large context spaces, and the difficulty of the problem highly depends on the complexity of the super arm reward model.

More recently, there has been a growing literature of using **neural networks** to facilitate learning in the bandit setting. These advancements exploit recent results on the generalization of deep neural networks (Jacot, Gabriel, and Hongler 2021; Arora et al. 2019; Cao and Gu 2019) and require no parametric assumptions on the reward as a function of the context other than an upper bound (Zhou, Li, and Gu 2020; Zhang et al. 2021). Yet while the field of neural contextual bandits is relatively well explored, to our knowledge, there is only one neural contextual combinatorial bandit study in the literature with regret bounds (Hwang, Chai, and Oh 2023). However, unlike our work they only use one neural network to learn the base arm reward function and rely solely on an exact oracle to do super arm selections. Ban, He, and Cook (2021) choose multiple arms in each round, but their setting is quite different since they assume K bandits and select an arm from each bandit in every round.

Our Contributions

In this work, we propose a **provably efficient contextual combinatorial neural bandit algorithm** while making mild assumptions about the reward functions. Our proposed algorithm makes use of two neural networks to learn the base arm and super arm reward functions, as shown in Figure 1. We formulate our problem in the Problem Formulation Section and then make the following **contributions:**

- We propose a **neural contextual combinatorial clustered bandit algorithm** (NeUClust) in the Proposed NeUClust Algorithm Section. To the best of our knowledge, it is the first to use neural networks to learn the base arm and super arm reward functions while making use of clustering to exploit the underlying structure of the context space to guide super arm selections. In doing so, we avoid the need for an oracle to choose the super arm.
- We **prove** (in Regret Analysis and Proofs) that NeUClust achieves $\widetilde{O}\left(\widetilde{d}\sqrt{T}\right)$ regret over T rounds, where $\widetilde{d}$ is the effective dimension of a neural tangent kernel matrix.
- We show through **experimental results** on real world recommendation datasets (MovieLens, Yelp) that our algorithm outperforms state-of-the-art neural contextual/-combinatorial and CC-MAB algorithms (Experiments).

We conclude by discussing future work.

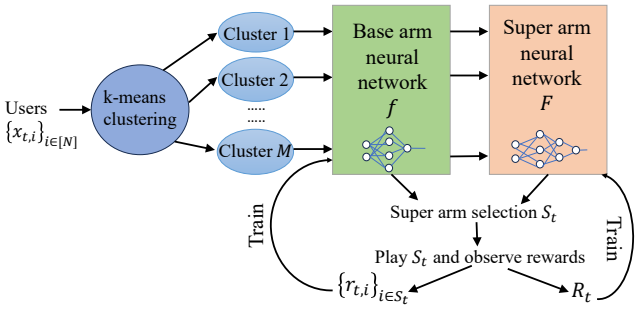

Figure 1: Our neural contextual combinatorial bandit formulation, with the online feedback loop for arm selection.

Problem Formulation

We first introduce some useful mathematical notation before outlining our CC-MAB model, the neural network models we employ for the reward functions, and our clustering-based method for selecting arms in each round. For a vector $\mathbf{x} \in \mathbb{R}^d$, we represent its ℓ_2 norm by $\|\mathbf{x}\|_2$, ℓ_1 norm by $\|\mathbf{x}\|_1$, ℓ_0 norm by $\|\mathbf{x}\|_0$, and transpose by $\mathbf{x}^T$. The ℓ_2 norm by a positive definite matrix $\mathbf{A}$ is defined by $\|\mathbf{x}\|_\mathbf{A} := \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}$. We denote by $[N]$, $N \in \mathbb{Z}^+$, the set $\{1, 2, \ldots, N\}$. We let $\mathbf{1}^K = (1, 1, ..., 1) \in \mathbb{R}^K$ denote the ones vector.

Contextual Combinatorial Bandit Setting

In this paper, we consider a contextual combinatorial bandit with N being the total number of **arms** and T the total number of **rounds**. At the start of every round t, the agent observes the **context** vectors of all arms, denoted by $\{\mathbf{x}_{t,i} \in \mathbb{R}^d \mid i \in [N]\}$, and chooses a subset of the available arms, namely a **super arm** $S_t \subset [N]$ with a fixed size, or budget, K, i.e., $|S_t| = K$. We denote by $\mathcal{S}$ the set of all feasible super arms with cardinality K, i.e., $\mathcal{S} = \{S \subset [N] \mid |S| = K\}$. In the application of movie recommendation, for example, the base arms could correspond to users where there is an incoming movie at every round which we would like to recommend to a subset of users. The context would correspond to past information about the ratings the users gave to movies and the genres of the movies the users rated.

When a super arm $S_t \in \mathcal{S}$ is selected in round t, the agent observes the **base arm rewards** of the chosen super arm, namely $\{r_{t,i}\}_{i\in S_t}$ and receives a **total (super arm) reward**

of $R(S_t, \mathbf{r}_t)$ where $\mathbf{r}_t = [r_{t,i}]_{i \in S_t}$ hence the super arm reward is a function of the individual base arm rewards. This type of bandit feedback at every round is often referred to as **semi-bandit feedback**, which is common in combinatorial bandits (Audibert, Bubeck, and Lugosi 2014). In the movie recommendation example setup, the base arm and super arm rewards would correspond to the rating given by one user and the collection of users we selected, which would be information that would be available to us. We use these observed super arm and base arm rewards in every round to train our neural networks as explained in more detail in the Proposed NeUClust Algorithm Section. We assume that the base arm rewards $r_{t,i}$ are generated as follows:

$$r_{t,i} = h(\mathbf{x}_{t,i}) + \xi_{t,i} \tag{1}$$

$\forall t \in [T]$ and $i \in [N]$ where h is an unknown function satisfying $0 \leq h(\mathbf{x}) \leq 1$ and $\xi_{t,i}$ is ζ-sub-Gaussian noise satisfying $\mathbb{E}[\xi_{t,i} | \mathbf{x}_{1,S_1}, ..., \mathbf{x}_{t-1,S_{t-1}}] = 0$ where $S_t \in \mathcal{S}$ denotes the selected super arm at round t. This is a standard assumption for the stochastic bandit setting. In our movie recommendation setting, (1) models the fact that the rating that a user gives to a movie mainly depends on the user's genre preferences, which has some uncertainty associated with it (encoded in $\xi_{t,i}$) as movie genre is not the sole factor influencing how much a user likes a movie.

Base Arm Reward Function

To learn the base arm reward function h, we use a fully connected neural network with depth $L + 1 \geq 3$ defined as follows (Zhou, Li, and Gu 2020; Zhang et al. 2021):

$$f(\mathbf{x}; \boldsymbol{\theta}) = \sqrt{m} \mathbf{W}_L \sigma(\mathbf{W}_{L-1} \sigma(...\sigma(\mathbf{W}_0 \mathbf{x})) \tag{2}$$

where $\sigma(x) = \max\{x, 0\}$ is the rectified linear unit (ReLU) activation function and $\boldsymbol{\theta} = [\text{vec}(\mathbf{W}_0)^T, ..., \text{vec}(\mathbf{W}_L)^T]^T \in \mathbb{R}^p$ is the weight vector of the neural network where $p = md + m^2(L-1) + m$ where m is the width of the hidden layers and for simplicity of analysis we assume the width is the same for every layer. We assume a depth of $L + 1$ to ensure that we have L hidden layers. This will simplify the notation in our later theoretical analysis. To denote the gradient of the neural network we use $\mathbf{g}(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} f(\mathbf{x}; \boldsymbol{\theta}) \in \mathbb{R}^p$. We make the following assumption about the base arm reward function h, which makes sense in many recommendation applications since the base arm reward function takes as input the average preference vector of a user according to the genres and the genre vector of a movie to output a rating. Hence if a user has a higher preference towards the genres of the movie, then the rating should also increase accordingly.

Assumption 1. *(Lipschitz continuity) $h(\mathbf{x}_{t,i})$ is Lipschitz continuous with respect to the context vector $\mathbf{x}_{t,i}$ of arm i, i.e. $\exists B' > 0$ such that for any $\mathbf{x}_{t,i}, \mathbf{x}_{t,i'}$ it holds that $|h(\mathbf{x}_{t,i}) - h(\mathbf{x}_{t,i'})| \leq B' \left\| \mathbf{x}_{t,i} - \mathbf{x}_{t,i'} \right\|_1$.*

Super Arm Reward Function

The super arm reward function $R(S, \mathbf{r})$ is a function of the selected arms' rewards and gives an understanding about how good the selected arm combination is. Let us denote the base arm rewards vector of a super arm S in round t with context vector $\mathbf{x}_{t,S} = [\mathbf{x}_{t,i}]_{i \in S}$ by $\mathbf{r}_t$, and we denote the expected super arm reward by $\mathbb{E}[R(S_t, \mathbf{r}_t)] = u(S_t, \mathbf{h}(\mathbf{x}_{t,S_t}))$ where $\mathbf{h}(\mathbf{x}_{t,S_t}) = [h(\mathbf{x}_{t,i})]_{i \in S_t}$. In this work the super arm reward function can be any function which satisfies the following mild assumptions, which are common in the combinatorial bandit setting (Li et al. 2016; Qin, Chen, and Zhu 2014; Nika, Elahi, and Tekin 2020).

Assumption 2. *(Lipschitz continuity) $u(S, \mathbf{h})$ is Lipschitz continuous with respect to the expected base arm rewards $\mathbf{h}$, i.e. $\exists B > 0$ such that for any $\mathbf{h}, \mathbf{h}'$ it holds that $|u(S, \mathbf{h}) - u(S, \mathbf{h}')| \leq B \sum_{i \in S} |h_i - h'_i|$.*

Assumption 3. *(Monotonicity) $u(S, \mathbf{h})$ is monotone non-decreasing with respect to the expected base arm reward vector $\mathbf{h}$ so that for any possible S when $h_i \leq h'_i, \forall i \in [N]$ then $u(S, \mathbf{h}) \leq u(S, \mathbf{h}')$.*

It should be noted that *these assumptions do not restrict the generalizability of our work*, since they hold in many real life applications of combinatorial bandits. For example in the previously mentioned movie recommendation setup, the super arm reward function could be the average rating given by the users to the movie that was recommended, which is then monotonic and Lipschitz continuous in each individual rating. Moreover, *we do not require an oracle that knows how to select arms which would maximize the super arm reward function*: as explained in Algorithm 1, we learn this through clustering and using the monotonicity assumption.

To learn the super arm reward function $R(S, \mathbf{r})$, we use a structurally constrained neural network, inspired by the approach in MonoNet (Nguyen et al. 2023) which is composed of monotonically connected layers, ensuring a monotonic relationship between the inputs and the output. Let us denote layer k of the network by $\mathbf{o}^{(k)}$ for $k = 0, 1, ..., L_m$ and hence we can write $\mathbf{o}^{(k+1)} = \sigma\left(\boldsymbol{\Theta}^{(k)} \mathbf{o}^{(k)} + \mathbf{b}^{(k)}\right)$, where $\boldsymbol{\Theta}^{(k)}$ is the weight matrix and $\mathbf{b}^{(k)}$ is the bias. The idea to form a monotonic layer is to transform the weight matrix such that it only has non-negative entries, since this will ensure a monotonic relationship between the input and output of that layer. Hence for a monotonic layer we have:

$$\mathbf{o}^{(k+1)} = \sigma\left(q\left(\boldsymbol{\Theta}^{(k)}\right) \mathbf{o}^{(k)} + \mathbf{b}^{(k)}\right) \tag{3}$$

where q is the weight transform function applied element wise to every entry of the weight matrix and could be any function with nonnegative outputs. We use a neural network with same width for every layer n and with depth $L_m + 1$. By structuring the super arm network so as to guarantee monotonicity, we ensure that the monotonicity assumption holds and *allow our theoretical guarantees to hold* as well.

Regret Formulation

The main objective of the agent in this setting is to minimize the **cumulative expected super arm regret**, which is the standard performance metric for CC-MAB problems and measures the gap in reward between the optimal and selected arms. The regret is formally defined as follows where

S_t^* represents the optimal super arm at round t:

$$\mathcal{R}(T) = \sum_{t=1}^{T} (u(S_t^*, \mathbf{h}(\mathbf{x}_{t,S_t^*}))) - u(S_t, \mathbf{h}(\mathbf{x}_{t,S_t}))) \quad (4)$$

Proposed NeUClust Algorithm

In this section, we present our algorithm Neural UCB Clustering (NeUClust). NeUClust is a neural network based contextual combinatorial upper confidence bound bandit algorithm that makes use of two neural networks to learn the base arm reward function and the super arm reward function. However, since we do not utilize an oracle, *choosing a super arm based on the optimism in the face of uncertainty (OIFU) principle (Lai, Robbins et al. 1985) with the super arm network is difficult*. We overcome this challenge by exploiting our monotonicity assumption and *clustering* the context space. Intuitively, base arms with similar contexts should yield similar rewards, and thus are similarly likely to be part of the optimal super arm. In some settings, we can cluster the context space offline, e.g., we may have sufficient information about users' movie genre preferences to cluster users offline, but in other settings we may need to collect context observations while running NeUClust. We thus propose online and offline variants of our NeUClust algorithm; we present the online version in this section.

NeUClust description. Algorithm 1 shows NeUClust's pseudocode. We first initialize the parameters of the first neural network (line 2), which aims to learn the base arm reward function, by randomly generating $\boldsymbol{\theta}_0 = [\text{vec}(\mathbf{W}_0)^T, ..., \text{vec}(\mathbf{W}_L)^T]^T$, where for each $\ell \in [L]$, $\mathbf{W}_\ell = (\mathbf{W}, 0; 0, \mathbf{W})$ where each entry of $\mathbf{W}$ is generated independently from $\mathcal{N}(0, 4/m)$ and $\mathbf{W}_L = (\mathbf{w}^T, -\mathbf{w}^T)$ with each entry of $\mathbf{w}$ independently sampled from $\mathcal{N}(0, 2/m)$. We initialize the parameters of the second neural network, which tries to learn the super arm reward function, by independently sampling each entry from $\mathcal{N}(1/n, 1)$ where n is the width of the network.

At the start of each round $t \in [T]$, the agent observes the contexts of all arms $\{\mathbf{x}_{t,i}\}_{i\in[N]}$ and clusters the arms based on their contexts (lines 4-5). NeUClust then uses the context information and the output of the base arm neural network and its gradient to construct an upper confidence bound (UCB) v_{t,c_j} of the expected reward for every arm c_j in every cluster c (lines 7-9). We then select the top K arms of each cluster based on the arm UCBs (line 11). The outputs of the base arm network f (i.e., estimates of base arm reward) are fed through a ReLU function (omitted in line 12 for simplicity), to ensure they are nonnegative, and then taken as input to the second network F (monotonic super arm network) to estimate the super arm reward of a cluster. Then the upper confidence bound vector $\mathbf{V}_t$ is used to select the cluster that has the highest average UCB of its corresponding arms (lines 12-14). Then the top K arms of the chosen cluster are played as S_t and the base arm rewards $\{r_{t,i}\}_{i\in S_t}$ and super arm reward R_t are observed (lines 15-16).

Once the rewards $\{r_{t,i}\}_{i\in S_t}$ and R_t are observed, the context vector is updated according to the observed base arm rewards of the played arms. The neural network parameters $\boldsymbol{\theta}_t$ and $\boldsymbol{\Theta}_t$ are updated by using gradient descent with step size η_1 and η_2 for J iterations to minimize the loss functions:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{t'=1}^{t} \left\| \mathbf{f}(\mathbf{x}_{t',S_{t'}}; \boldsymbol{\theta}) - \mathbf{r}_{t'}) \right\|_2^2 + \frac{m\lambda_1}{2} \left\| \boldsymbol{\theta} - \boldsymbol{\theta}_0 \right\|_2^2$$

$$\mathcal{L}(\boldsymbol{\Theta}) = \sum_{t'=1}^{t} \frac{(F(\mathbf{f}(\mathbf{x}_{t',S_{t'}}); \boldsymbol{\Theta}) - R_{t'})^2 + n\lambda_2 \left\| \boldsymbol{\Theta} - \boldsymbol{\Theta}_0 \right\|_2^2}{2}$$

where $\mathbf{f}(\mathbf{x}_{t',S_{t'}}; \boldsymbol{\theta}) = [f(\mathbf{x}_{t,i})]_{i\in S_{t'}}$, $\mathcal{L}(\boldsymbol{\theta})$ shows the loss minimized using l_2-regularization for the base arm neural network and $\mathcal{L}(\boldsymbol{\Theta})$ is the loss minimized for the super arm neural network. $R_{t'}$ is the observed super arm reward of the selected super arm $S_{t'}$ and $\mathbf{r}_{t'}$ represents the vector of observed base arm rewards of the selected super arm. The hyperparameters λ_1, λ_2 adjust for the level of regularization, which centers at the randomly initialized weight vectors. We define the positive exploration scaling factor γ_t similar to (Hwang, Chai, and Oh 2023; Zhou, Li, and Gu 2020) as

Algorithm 1: NeUClust (Online)

1: **Input:** Number of rounds T, regularization parameters λ_1 and λ_2, step sizes η_1 and η_2, number of gradient descent steps J, network widths m and n, network depths L and L_m, number of clusters M, maximum number of iterations of clustering i_c, size of super arm K, norm parameter S.
2: Randomly initialize $\boldsymbol{\theta}_0$ and $\boldsymbol{\Theta}_0$ as described previously and $\mathbf{Z}_0 = \lambda_1 \mathbf{I}$
3: **for** $t = 1, \ldots, T$ **do**
4: Observe $\{\mathbf{x}_{t,i}\}_{i\in[N]}$
5: Run k-means clustering on the observed contexts $\{\mathbf{x}_{t,i}\}_{i\in[N]}$ with $k = M$ and at most i_c iterations
6: Initialize $\mathbf{V}_t = (0, 0, ..., 0) \in \mathbb{R}^M$
7: **for** $c = 1, \ldots, M$ **do**
8: **for** $j \in c$ **do**
9: $v_{t,c_j} = f(\mathbf{x}_{t,c_j}; \boldsymbol{\theta}_{t-1}) + \gamma_t ||\mathbf{g}(\mathbf{x}_{t,c_j}; \boldsymbol{\theta}_{t-1})/\sqrt{m}||_{\mathbf{Z}_{t-1}^{-1}}$
10: **end for**
11: Within cluster c, find top K cluster elements with the highest v value, $\mathbf{b}_c \in \mathbb{R}^K$ vector includes the indices of these elements
12: $V_{t,c} = \sum_{a\in\mathbf{b}_c} v_{t,c_a} + F(\mathbf{f}(\mathbf{x}_{t,\mathbf{b}_c}; \boldsymbol{\theta}_{t-1}))$
13: **end for**
14: $c'' = \arg\max_c(\mathbf{V}_t)$
15: $S_t = \mathbf{b}_{c''}$
16: Play super arm S_t and observe base arm rewards $\{r_{t,i}\}_{i\in S_t}$ and super arm reward R_t
17: Update context vector $\{\mathbf{x}_{t,i}\}_{i\in S_t}$ of the arms played by using the observed base arm rewards
18: Update $\mathbf{Z}_t = \mathbf{Z}_{t-1} + \sum_{i\in S_t} \mathbf{g}(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1}) \mathbf{g}(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})^T / m$
19: Update weights $\boldsymbol{\theta}_t$ by minimizing the loss $\mathcal{L}(\boldsymbol{\theta})$ using gradient descent with step size η_1 for J iterations.
20: Update weights $\boldsymbol{\Theta}_t$ by minimizing the loss $\mathcal{L}(\boldsymbol{\Theta})$ using gradient descent with step size η_2 for J iterations.
21: Update γ_t as explained in Eq. 5
22: **end for**

$$\gamma_t = \Gamma_{1,t}\left(\zeta\sqrt{\log\frac{\det \mathbf{Z}_t}{\det\lambda_1\mathbf{I}} + \Gamma_{2,t} - 2\log\Delta} + \sqrt{\lambda_1}S\right) + (\lambda_1 + C_1tKL)\left((1-\eta_1 m\lambda_1)^{\frac{J}{2}}\sqrt{\frac{tK}{\lambda_1}} + \Gamma_{3,t}\right) \tag{5}$$

for some $\Gamma_{1,t}, \Gamma_{2,t}, \Gamma_{3,t} > 0$ (defined in the extended version of our paper).

Super arm selection without an oracle. As discussed above, our super arm selection first finds the optimal cluster (in terms of the super arm reward) and then selects the K arms with largest base arm rewards from that cluster. By monotonicity of the super arm reward, we thus maximize the super arm reward subject to the constraint that all base arms belong to the chosen cluster. For example, if a user generally likes movies of a particular genre, or more generally prefers a certain kind of product, our clustering should group movies of similar genres (products of similar type) together and then recommend them to the user. While restricting the selected base arms to a specific cluster may lead to a suboptimal reward, our experimental results show that NeUClust outperforms neural combinatorial bandit algorithms that do not have this clustering restriction: such algorithms generally impose other restrictions on the super arm reward, e.g., the presence of an oracle. Moreover, we experimentally verify that real-world recommendation datasets naturally have a clustered structure in the extended version.

For the **offline variant** of the algorithm, we do clustering only once at round $t = 1$ and do not update the context vector after observing the base arm rewards of the arms that were played. This algorithm would work well when the contexts have been formed with an abundance of prior offline data. For example for movie recommendation, this would mean that the context vector, which represents the average rating the user gave to each genre, is formed with a sufficient number of already available ratings from that user.

Regret Analysis

In this section we analyze and derive the regret of the presented algorithm NeUClust in Algorithm 1. We denote by $\{\mathbf{x}^i\}_{i=1}^{TN}$ the collection of all the contexts $\{\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{T,N}\}$.

Definition 1. *(Jacot, Gabriel, and Hongler 2021; Cao and Gu 2019) For a given set of contexts $\{\mathbf{x}^i\}_{i=1}^{TN}$ define*

$$\widetilde{\mathbf{H}}_{i,j}^{(1)} = \mathbf{\Sigma}_{i,j}^{(1)} = \langle \mathbf{x}^i, \mathbf{x}^j\rangle, \mathbf{A}_{i,j}^{(\ell)} = \begin{pmatrix} \mathbf{\Sigma}_{i,i}^{(\ell)} & \mathbf{\Sigma}_{i,j}^{(\ell)} \\ \mathbf{\Sigma}_{i,j}^{(\ell)} & \mathbf{\Sigma}_{j,j}^{(\ell)} \end{pmatrix}$$

$$\mathbf{\Sigma}_{i,j}^{(\ell+1)} = 2\mathbb{E}_{(u,y)\sim\mathcal{N}(\mathbf{0},\mathbf{A}_{i,j}^{(\ell)})}[\sigma(u)\sigma(y)]$$

$$\widetilde{\mathbf{H}}_{i,j}^{(\ell+1)} = 2\widetilde{\mathbf{H}}_{i,j}^{(\ell)}\mathbb{E}_{(u,y)\sim\mathcal{N}(\mathbf{0},\mathbf{A}_{i,j}^{(\ell)})}[\sigma'(u)\sigma'(y)] + \mathbf{\Sigma}_{i,j}^{(\ell+1)}.$$

*Then $\mathbf{H} = (\widetilde{\mathbf{H}}^{(L)} + \mathbf{\Sigma}^{(L)})/2$ is called the **neural tangent kernel** (NTK) matrix on the context set $\{\mathbf{x}^i\}_{i=1}^{TN}$.*

The NTK matrix $\mathbf{H}$ is defined iteratively from the input to the output layer of the L layer neural network (Zhou, Li, and Gu 2020; Zhang et al. 2021; Hwang, Chai, and Oh 2023).

Assumption 4. *(Bounded contexts) For any $i \in [TN]$, $\|\mathbf{x}^i\|_2 = 1$ and $[\mathbf{x}^i]_j = [\mathbf{x}^i]_{j+d/2}$ for $1 \le j \le d/2$ and for some $\lambda_0 \ge 0$, $\mathbf{H} \succeq \lambda_0\mathbf{I}$.*

This is a fairly relaxed assumption which is commonly used in the neural bandit literature (Hwang, Chai, and Oh 2023; Zhang et al. 2021; Zhou, Li, and Gu 2020). The condition on the norm is for ease of analysis, and we can always form a new context $\mathbf{x}' = [\mathbf{x}^T, \mathbf{x}^T]^T/\sqrt{2}$ to satisfy the assumption on the entries of $\mathbf{x}$. An assumption is also made on the positive semi-definiteness of $\mathbf{H}$, which is also standard.

Definition 2. *The **effective dimension** $\widetilde{d}$ of the NTK matrix $\mathbf{H}$ with regularization parameter λ_1 is defined as*

$$\widetilde{d} = \frac{\log\det(\mathbf{I} + \mathbf{H}/\lambda_1)}{\log(1 + TN/\lambda_1)}$$

The effective dimension roughly quantifies the number of dimensions of contexts in the Reproducable Kernel Hilbert Space which the NTK matrix spans. It was first introduced by (Valko et al. 2013) for kernelised contextual bandits and we adapt the definition proposed by (Yang and Wang 2019).

Assumption 5. *(Clustering) Let $\mathcal{A}_t$ be the set of clusters that we find in round t and $\mathcal{C}$ the ground truth clusters of the context space. Without loss of generality, suppose $\mathcal{A}_t$ and $\mathcal{C}$ are ordered so that mapping each cluster $j \in \mathcal{A}_t$ to cluster $j \in \mathcal{C}$ minimizes the average distance between cluster centers in $\mathcal{A}_t$ and $\mathcal{C}$. Then the dot product of the mean contexts of clusters c and c' satisfies $\bar{\mathbf{x}}_{c,t} \cdot \mathbf{x}_{c'} \le \delta_t\ \forall t \in [T]$ and $\forall c' \ne c$ for some $\delta_t > 0$ which is a decreasing function of t. Further, $\bar{\mathbf{x}}_{c,t} \cdot \bar{\mathbf{x}}_{c_t^*} \ge P_t$ for some $P_t > 0$ which is a non-decreasing function of t, where c_t^* denotes the optimal cluster for a given round, $\bar{\mathbf{x}}_{c_t^*}$ is its mean context vector, and we play the super arm in cluster c at round t.*

Assumption 6. *(Nonzero mean contexts) $\forall t \in [T], \forall c' \in \mathcal{C}$, $\|\bar{\mathbf{x}}_{c,t}\|_0 = d, \|\mathbf{x}_{c'}\|_0 = d$, i.e., $\bar{\mathbf{x}}_{c,t}$ and $\mathbf{x}_{c'}$ have no 0 entries. $\mathcal{C}$ is the set of ground truth clusters and we play the super arm in cluster c at round t.*

Assumption 5 essentially states that we are able to identify the optimal cluster and separate the clusters more accurately over time, as we select δ_t to be a decreasing and P_t to be a non-decreasing function; we also implicitly assume that the clusters are sufficiently different from one another, which can be assured by shifting and normalizing the contexts. Assumption 6 states that the mean cluster vector of our chosen cluster in any round and the mean cluster vectors of the actual clusters have no zero elements. This is necessary for our proofs but is not very restrictive in practice since the condition is on the mean vectors. Next we give a condition on the width of the base arm network, which is necessary since our theoretical results hold for overparameterized networks.

Condition 1. *The width of the base arm network satisfies*

$$m \ge C\max\big\{T^6N^6L^6\lambda_0^{-1}\log(T^2N^2L/\Delta)\max(\lambda_0^{-4}, 1),$$
$$L^{-\frac{3}{2}}K^{-\frac{1}{2}}\lambda_1^{\frac{1}{2}}(\log(TNL^2/\Delta))^{\frac{3}{2}}\big\}$$
$$m(\log m)^{-3} \ge \max\big\{C\lambda_1^{-1}L^{24}K^{10}T^{10},$$
$$CT^7K^7L^{14}\lambda_1^{-7}(\lambda_1 + TKL)^6(1 + \sqrt{TK/\lambda_1})^6\big\}.$$

for some constant $C > 0$ and $\Delta \in (0, 1)$ which represents a probability $(1 - \Delta)$ with which Lemmas 2 and 3, presented next, hold. While some of these conditions on m resemble those on the width of the network in (Hwang, Chai, and Oh 2023; Zhou, Li, and Gu 2020), we include extra terms due to the neural network used to estimate the super arm reward.

While much of our regret bound proof follows the outline of Hwang, Chai, and Oh (2023)'s proof of a regret bound for neural contextual combinatorial bandits, Hwang, Chai, and Oh (2023) crucially *do not use a neural network to estimate the super arm*, instead employing a known oracle. We next outline the **new theoretical results** that we need to extend the proof to our setting, which chiefly involves bounding the output of the super arm neural network and incorporating the error from clustering.

Lemma 1. *(Bound on base arm network output) With probability* $(1 - O(L) \cdot e^{-\Omega(m\varepsilon^2/L)})(1 - e^{-\Omega(m\omega^{\frac{2}{3}}L)})(1 - me^{(-m\rho^2/4)})$ *the output of the first neural network*

$$f(\mathbf{x}^i) \leq 4\sqrt{1/\lambda_1}(9T^{-\beta\alpha+0.5})$$

for some $\beta > 0$, $\alpha > 0$ *and* $\omega \leq O(L^{-\frac{9}{2}}(\log m)^{-3})$.

Here ε represents the largest error between the predicted output of the first neural network and the actual base arm reward of the arm selected over T rounds as in (Allen-Zhu, Li, and Song 2019), ρ represents an upper bound on the base arm network weights which holds with high probability by the Gaussian tail bound. The above lemma bounds the output of the base arm neural network for all time steps, which we use to bound the output of the super arm network as shown in the next lemma. To prove Lemma 1, we make use of Lemmas 7 and 8, which require a perturbation to the weights; the ℓ_2 norm of that perturbation is upper bounded by ω. We show that this perturbation is within the values the actual weights take when we do gradient descent. The *main challenge* is showing this bound holds for all time steps and that the weights of the network stay within the desired region.

Lemma 2. *(Bound on super arm network output and expected base arm reward vector) With probability* $(1-O(L)\cdot e^{-\Omega(m\varepsilon^2/L)})(1-e^{-\Omega(m\omega^{\frac{2}{3}}L)})(1-me^{(-m\rho^2/4)})(1-\Delta)$ *the output of the super arm network and the base arm reward vector for the chosen super arm* S_t *are bounded by*

$$F(\mathbf{f}(\mathbf{x}_{t,S_t})) \leq 4\sqrt{1/\lambda_1}(9T^{-\beta\alpha+0.5})Kn^{L_m}$$
$$\mathbf{h}(\mathbf{x}_{t,S_t}) \leq \mathbf{v}_{t,S_t} + \mathbf{1}^K \cdot F(\mathbf{f}(\mathbf{x}_{t,S_t})).$$

The second inequality in this lemma involves an entry-wise comparison between two vectors. This lemma makes use of the fact that the super arm network takes as input the base arm network outputs, and this bound is also used to upper bound the expected base arm reward. This lemma helps bound the expected super arm regret of a single round, which will be done in the following lemma.

Lemma 3. *(Bound on expected regret for a single round) For any* $\Delta \in (0, 1)$*, suppose the width of the neural network* m *satisfies Condition 1. If* $\eta_1 \leq C_1(TKmL + m\lambda_1)^{-1}$*, and* $\lambda_1 \geq C_2LK$*, for some positive constant* C_1, C_2 *with* $C_2 \geq \sqrt{\max_{t,i} \|\mathbf{g}(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})/\sqrt{m}\|_2^2 / L}$ *and let* γ_t *be as defined in Equation 5. Then the expected super arm regret of a single round is bounded with probability at least* $1 - \Delta$ *as*

$$u(\mathbf{h}(\mathbf{x}_{t,S_t^*})) - u(\mathbf{h}(\mathbf{x}_{t,S_t})) \leq$$
$$D\sqrt{\sum_{i \in S_t} \|\mathbf{g}(\mathbf{x}_{t,i}; \boldsymbol{\theta}_{t-1})/\sqrt{m}\|^2_{\mathbf{Z}_{t-1}^{-1}}}$$
$$+ BKe_t + BK^2 4\sqrt{1/\lambda_1}(9T^{-\beta\alpha+0.5})n^{L_m} +$$
$$B'BK\left(\frac{C''Q\delta_t\sqrt{d}N}{P_tY} + \frac{NQ\delta_t\sqrt{d}}{Y}\right)$$

where $e_t := C_3\gamma_{t-1}t^{\frac{1}{6}}K^{\frac{1}{6}}L^{\frac{7}{2}}m^{-\frac{1}{6}}\lambda_1^{-\frac{2}{3}}\sqrt{\log m} + C_4t^{\frac{2}{3}}K^{\frac{2}{3}}m^{-\frac{1}{6}}\lambda_1^{-\frac{2}{3}}\sqrt{\log m}$*,* $D = 2B\gamma_T\sqrt{K}$ *for some absolute constants* $C_3, C_4 > 0$*,* $Y > 0$ *is a lower bound on the* ℓ_2 *norm of all the actual clusters* $c' \in \mathcal{C}$ *and* $Q > 0$ *is a constant related to the Kantorovich inequality.*

The per round regret in Lemma 3 can be separated into different terms coming from clustering, super arm network and base arm network. We give a more detailed explanation after Theorem 1.

Theorem 1. *(Regret bound) Supposing that Assumptions 1-6 hold, let* $\mathbf{h} = [h(\mathbf{x}^i)]_{i=1}^{TN}$*, and select the hyperparameters of the algorithm as follows*

$$\eta_1 = C_1(TKmL + m\lambda_1)^{-1}$$
$$\frac{16m \cdot 81T^{(-2\beta\alpha+1)}K^2n^{2L_m}}{e_T^2} \geq \lambda_1 \geq C_2LK$$
$$J = 2\log\left(\sqrt{\lambda_1/(TK)}/(\lambda_1 + C_3TKL)\right)TKL/(C_1\lambda_1)$$
$$S \geq \sqrt{2\mathbf{h}^T\mathbf{H}^{-1}\mathbf{h}}$$

for constants C_1, C_2, C_3 *defined as in Lemma 3. Then if* m *is chosen such that it satisfies Condition 1, the cumulative expected regret of NeUClust over* T *rounds is* $\mathcal{R}(T) = \widetilde{O}\left(\widetilde{d}\sqrt{T}\right)$ *and is bounded by*

$$\mathcal{R}(T) < \underbrace{2B\sqrt{K}\sqrt{T(2\widetilde{d}\log(1 + TN/\lambda_1) + 3)}\cdot}$$
$$\underbrace{\left(2\zeta\sqrt{\widetilde{d}\log(1 + TN/\lambda_1) + 3 - 2\log\Delta} + 2\sqrt{\lambda_1}S + 2\right)}_{Y_T} +$$
$$\underbrace{BK\left(2\zeta\sqrt{\widetilde{d}\log(1 + \frac{TN}{\lambda_1}) + 3 - 2\log\Delta} + 2\sqrt{\lambda_1}S + 3\right)}_{U_T}$$
$$+ \underbrace{BK^2 4\sqrt{1/\lambda_1}(9\sqrt{T})n^{L_m}}_{P_T} +$$
$$\underbrace{B'BKC_5\left(\frac{NQ\sqrt{d}\left(2\sqrt{T} - 1\right)(C'' + C_6)}{C_6Y}\right)}_{E_T}$$

We thus see that our total regret grows as $\sqrt{T}$, which matches the dependence in prior combinatorial neural bandit work (Hwang, Chai, and Oh 2023). As can be seen from the decomposition, the regret bound is separated into different terms where there is a term coming from clustering (E_T), a term coming from the super arm network (P_T), and two terms coming from the base arm network (Y_T, U_T).

Dependence on K and δ_t. From our regret bound it seems that we are super linear in the super arm size K due to the P_T term, but we can select λ_1 such that we get $\sqrt{K}$ dependence. Similarly the clustering term δ_t could be chosen such that we get sublinear regret in K for E_T. The only unavoidable linear K term is the one coming from U_T. In contrast, the bound of Hwang, Chai, and Oh (2023) is sublinear in terms of K, but it should be noted that this is due to the use of the oracle in their algorithm. The main reason we get linear dependence on K for our bound is because we do not use such an oracle in selecting the super arm.

From this bound, we see that if δ_t is a constant or non-decreasing with t or P_t is decreasing with t, we could incur linear regret. This is intuitive with the formation of our NeUClust algorithm: if the context space does not have an underlying clustered structure or we cannot cluster users with similar preferences correctly, then since super arm selections are guided by clusters we would not make optimal or near optimal arm selections, incurring linear regret.

Experiments

In this section, we measure and evaluate the performance of NeUClust on two recommendation datasets: MovieLens 25M Dataset [1] and the Yelp Open Dataset[2]. We compare our algorithm with the following state of the art combinatorial and neural bandit algorithms: (1) **CN-UCB** (Hwang, Chai, and Oh 2023) which uses a single neural network to approximate the base arm rewards of the arms and uses an exact oracle to select super arms, (2) **Neural-MAB** (Lin et al. 2022) which uses two networks to learn the base arm and super arm reward functions and selects the super arm by going over arms individually and estimating their contribution to the super arm reward, (3) **CC-MAB** (Chen, Xu, and Lu 2018) which partitions the context space into hypercubes for exploration and exploitation, and (4) **K-LinUCB** (Chu et al. 2011) which is a combinatorial version of the LinUCB algorithm and assumes linear realizability. We use greedy oracles for algorithms that require an oracle.

Experiment Setup

We first detail the movie recommendation setup for MovieLens then detail the restaurant recommendation setting for Yelp. Due to a lack of space, we give details about these datasets in the extended version.

We assume that every round we need to recommend an incoming movie to K users, hence each base arm corresponds to a user. The context of a user is represented as its user preference vector, which is a vector showing the average rating the user gave to movies they rated for the $d = 20$ genres. For example, if a user has only watched and rated crime, thriller and horror films, the context of the user will be composed of zeros for the remaining 17 genres. We use this context information to cluster users and identify user types, i.e. users which prefer similar genres. The base arm reward is a binary indicator (0 or 1) of whether the user liked the movie that was recommended (i.e., gave a rating ≥ 4.0). The super arm reward is a binary indicator of whether at least %80 of the K selected users liked the movie (i.e., $\geq$ %80 of users gave a rating $\geq$ 4.0); this reward is highly nonlinear and captures the overall success of our recommendations. As a movie comes in round t, we use the following function to generate the true mean/reward of base arm (user) i for that movie z_t

$$\mu_i = 2/\left(1 + \exp\left(-\frac{\langle \mathbf{a}_{z_t}, \mathbf{x}_{t,i}\rangle}{2\|\mathbf{a}_{z_t} \odot \mathbf{x}_{t,i}\|_0}\right)\right) - 1 \qquad (6)$$

where here $\langle\cdot\rangle$ and $\odot$ represent the inner product and element-wise product respectively, $\mathbf{a}_{z_t}$ represents the genre vector of the movie and $\mathbf{x}_{t,i}$ represents the context of the base arm. We define the mean of a base arm in this way since as the genre vector and context of a user are more similar, it is more likely that the user will like the movie. We let the base arm reward of arm i be defined by the function $r(\mathbf{x}_{t,i}) \sim Bern(\mu_i)$ to capture uncertainty in a user's ratings for movies of a given genre. The super arm reward of a super arm S_t is defined as follows $R(S_t, \mathbf{r}_t) = 1$ if $\sum_{k=1}^{K} r_{t,k} \geq 0.8K$ else $R(S_t, \mathbf{r}_t) = 0$.

For the restaurant recommendation setting for Yelp, in each round, there is an incoming user, and we need to select K restaurants to recommend to the user from a total of approximately 50,000 restaurants. We use term frequency-inverse document frequency (tf-idf) to construct the contexts for the restaurants and cluster the restaurants using this category information. We filter some categories which do not appear often for a final context dimension of $d = 171$. We also construct the user preference vector using the ratings the user gave for the restaurants having these categories. We construct the mean of a base arm, base arm rewards and super arm rewards the same way as explained above for the movie recommendation setup, now using the restaurant context and user preference vector of the incoming user at round t.

Experimental Results

We run our experiments for the offline variant of NeUClust (Algorithm 2, pseudocode in extended version). Due to a lack of space, we give the explanation of the hyperparameter tuning and experimental details for all algorithms in the extended version. The given results for the MovieLens dataset are the mean over 10 independent runs for each of the algorithms, similarly for the Yelp dataset the mean is taken over 10 independent runs. The error bars in the figures (placed at every 100 points for clarity) indicate $\pm$ std. In the extended version, we give further experimental results on NeUClust's effectiveness and the validity of our clustering assumption.

Figures 2 and 3 show the regret achieved by all algorithms on the MovieLens and Yelp datasets, respectively. We can see from both figures that *our proposed NeUClust algorithm*

[1] https://grouplens.org/datasets/movielens/

[2] https://www.yelp.com/dataset

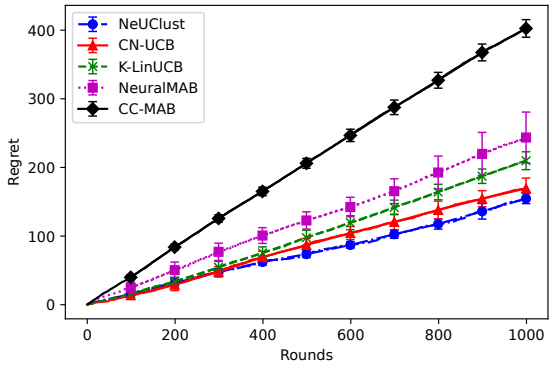

Figure 2: Super arm regret plot for the MovieLens dataset. Our NeUClust algorithm has lower regret than all baselines.

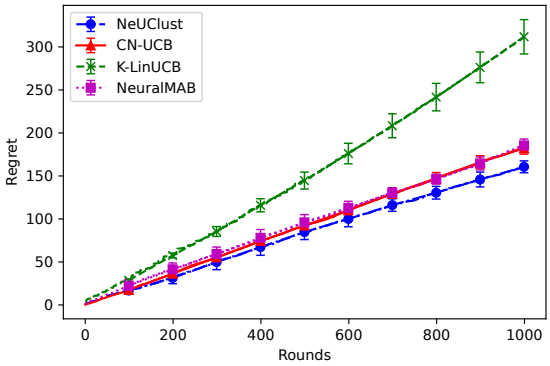

Figure 3: Super arm regret plot for the Yelp dataset. Our NeUClust algorithm has lower regret than all baselines.

outperforms the other baseline algorithms in terms of super arm regret for both datasets. The regret gap even grows over time. The results indicate that unlike some algorithms, NeUClust's regret does not suffer from the increase in the context dimension in going from MovieLens (20-dimensional context) to Yelp (171-dimensional context); indeed, NeUClust exhibits a larger reward gap with the baselines for Yelp compared to MovieLens. As expected, CN-UCB performs most similarly to our proposed algorithm due to its similar algorithmic structure to NeUClust, even though the two algorithms have major differences related to their super arm selection and NeUClust's usage of a second neural network for super arm reward estimation.

We have excluded the results of the Yelp experiments for the CCMAB algorithm since the performance was considerably worse both in terms of runtime and regret, which might have been due to the larger context space that could have impacted the discretization approach used by CCMAB.

To compare the performance of the baselines and our proposed algorithm, we include the following table which displays the final cumulative regret and the runtime in seconds of the algorithms. These results illustrate that despite the use of a second neural network, NeUClust performs comparably in terms of runtime while achieving lower regret.

	Regret (ML)	RT (ML)	Regret (Y)	RT (Y)
NeUClust	**152**	486	**159**	1988
CN-UCB	168	332	183	1553
CCMAB	403	**141**	-	-
K-LinUCB	207	1352	310	81385
NeuralMAB	227	187	185	**1317**

Table 1: Cumulative regret and runtime (RT) (rounded in seconds) comparison of the algorithms for the MovieLens (ML) and Yelp (Y) datasets. NeUClust's runtime is significantly below K-LinUCB's but slightly higher than CN-UCB and NeuralMAB due to using a second neural network to estimate super-arm rewards.

Conclusion

In this paper, we focus on the contextual combinatorial bandit problem and its application to recommender systems. We propose a novel solution algorithm NeUClust, which uses two neural networks to estimate the reward functions and exploits the underlying structure in the context space by doing clustering to guide super arm selections, bypassing the need for an oracle. We believe that having an algorithm which does not require the existence of an oracle is crucial, as some combinatorial optimization problems do not have ready-to-use oracles, such as the maximum coverage problem often considered in recommendation contexts (Kaminskas and Bridge 2016), or only admit approximation oracles (Kong et al. 2021). Using theoretical results of previous works for overparameterized networks and proposing new lemmas to handle clustering and neural network contributions, we prove that our algorithm achieves $\widetilde{O}\left(\widetilde{d}\sqrt{T}\right)$ cumulative regret. We also run experiments on recommendation datasets which illustrate that our algorithm competes well with other algorithms.

Future work includes further investigation of the use of different clustering methods within NeUClust, especially in other applications. In addition, relaxing the constraint that the chosen super arm has all arms corresponding to a single cluster could be explored. With this change, the best arms from different clusters could be selected in the super arm, given that the clusters perform similarly in terms of reward. This modification would thus allow the algorithm to select more diverse super arms, which could make the algorithm more adaptable to different applications beyond recommendation systems. Another direction to explore is to develop an algorithm to adaptively learn the number of clusters M instead of assuming it is known. Our experiments varying the number of clusters M (available in the extended version) show that the value of M used can significantly change the regret, and that a larger number of clusters does not always lead to better performance.

Acknowledgements

This work was supported by the Office of Naval Research under grant N000142412073 and by the National Science Foundation under grant CNS-2103024.

References

Allen-Zhu, Z.; Li, Y.; and Song, Z. 2019. A Convergence Theory for Deep Learning via Over-Parameterization. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 242–252. PMLR.

Arora, S.; Du, S. S.; Hu, W.; Li, Z.; Salakhutdinov, R.; and Wang, R. 2019. *On Exact Computation with an Infinitely Wide Neural Net*, 1–36. Red Hook, NY, USA: Curran Associates Inc.

Audibert, J.-Y.; Bubeck, S.; and Lugosi, G. 2014. Regret in Online Combinatorial Optimization. *Math. Oper. Res.*, 39(1): 31–45.

Ban, Y.; He, J.; and Cook, C. B. 2021. Multi-Facet Contextual Bandits: A Neural Network Perspective. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, 35–45. New York, NY, USA: Association for Computing Machinery. ISBN 9781450383325.

Bui, L.; Johari, R.; and Mannor, S. 2012. Clustered Bandits. arXiv:1206.4169.

Cao, Y.; and Gu, Q. 2019. *Generalization Bounds of Stochastic Gradient Descent for Wide and Deep Neural Networks*, 1–11. Red Hook, NY, USA: Curran Associates Inc.

Chen, L.; Xu, J.; and Lu, Z. 2018. Contextual Combinatorial Multi-Armed Bandits with Volatile Arms and Submodular Reward. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, 3251–3260. Red Hook, NY, USA: Curran Associates Inc.

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial Multi-Armed Bandit: General Framework and Applications. In Dasgupta, S.; and McAllester, D., eds., *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 151–159. Atlanta, Georgia, USA: PMLR.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. 2011. Contextual Bandits with Linear Payoff Functions. In Gordon, G.; Dunson, D.; and Dudík, M., eds., *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, 208–214. Fort Lauderdale, FL, USA: PMLR.

Elahi, S.; Atalar, B.; Öğüt, S.; and Tekin, C. 2023. Contextual Combinatorial Multi-output GP Bandits with Group Constraints. *Transactions on Machine Learning Research*.

Gentile, C.; Li, S.; and Zappella, G. 2014. Online Clustering of Bandits. In Xing, E. P.; and Jebara, T., eds., *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, 757–765. Bejing, China: PMLR.

Hwang, T.; Chai, K.; and Oh, M.-H. 2023. Combinatorial Neural Bandits. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 14203–14236. PMLR.

Jacot, A.; Gabriel, F.; and Hongler, C. 2021. Neural Tangent Kernel: Convergence and Generalization in Neural Networks (Invited Paper). In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, 6. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380539.

Kaminskas, M.; and Bridge, D. 2016. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 7(1): 1–42.

Kong, F.; Yang, Y.; Chen, W.; and Li, S. 2021. The Hardness Analysis of Thompson Sampling for Combinatorial Semi-bandits with Greedy Oracle. *Advances in Neural Information Processing Systems*, 34: 26701–26713.

Lai, T. L.; Robbins, H.; et al. 1985. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1): 4–22.

Li, S.; Wang, B.; Zhang, S.; and Chen, W. 2016. Contextual Combinatorial Cascading Bandits. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 1245–1253. New York, New York, USA: PMLR.

Lin, S.; Yao, Y.; Zhang, P.; Noh, H. Y.; and Joe-Wong, C. 2022. A neural-based bandit approach to mobile crowdsourcing. In *Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications*, HotMobile '22, 15–21. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392181.

Nguyen, A.-P.; Moreno, D. L.; Le-Bel, N.; and Rodríguez Martínez, M. 2023. MonoNet: Enhancing interpretability in neural networks via Monotonic Features. *Bioinformatics Advances*, 3(1): vbad016.

Nika, A.; Elahi, S.; and Tekin, C. 2020. Contextual Combinatorial Volatile Multi-armed Bandit with Adaptive Discretization. In Chiappa, S.; and Calandra, R., eds., *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, 1486–1496. PMLR.

Qin, L.; Chen, S.; and Zhu, X. 2014. Contextual Combinatorial Bandit and its Application on Diversified Online Recommendation. In *SDM*.

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*, UAI'13, 654–663. Arlington, Virginia, USA: AUAI Press.

Wen, Z.; Kveton, B.; and Ashkan, A. 2015. Efficient Learning in Large-Scale Combinatorial Semi-Bandits. In *Proceedings of the 32nd International Conference on Inter-*

national Conference on Machine Learning - Volume 37, ICML'15, 1113–1122. JMLR.org.

Yang, L. F.; and Wang, M. 2019. Reinforcement Learning in Feature Space: Matrix Bandit, Kernels, and Regret Bound. arXiv:1905.10389.

Zhang, W.; Zhou, D.; Li, L.; and Gu, Q. 2021. Neural Thompson Sampling. In *International Conference on Learning Representations*.

Zhou, D.; Li, L.; and Gu, Q. 2020. Neural Contextual Bandits with UCB-based Exploration. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 11492–11502. PMLR.

Zong, S.; Ni, H.; Sung, K.; Ke, N. R.; Wen, Z.; and Kveton, B. 2016. Cascading Bandits for Large-Scale Recommendation Problems. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, 835–844. Arlington, Virginia, USA: AUAI Press. ISBN 9780996643115.