Graph Neural Bandits

Yunzhe Qi*
University of Illinois at Urbana-Champaign
yunzheq2@illinois.edu

Yikun Ban*
University of Illinois at Urbana-Champaign
yikunb2@illinois.edu

Jingrui He
University of Illinois at Urbana-Champaign
jingrui@illinois.edu

ABSTRACT

Contextual bandits algorithms aim to choose the optimal arm with the highest reward out of a set of candidates based on the contextual information. Various bandit algorithms have been applied to real-world applications due to their ability of tackling the exploitation-exploration dilemma. Motivated by online recommendation scenarios, in this paper, we propose a framework named **Graph Neural Bandits (GNB)** to leverage the collaborative nature among users empowered by graph neural networks (GNNs). Instead of estimating rigid user clusters as in existing works, we model the "fine-grained" collaborative effects through estimated user graphs in terms of exploitation and exploration respectively. Then, to refine the recommendation strategy, we utilize separate GNN-based models on estimated user graphs for exploitation and adaptive exploration. Theoretical analysis and experimental results on multiple real data sets in comparison with state-of-the-art baselines are provided to demonstrate the effectiveness of our proposed framework.

CCS CONCEPTS

• **Theory of computation** → **Online learning algorithms**; • **Information systems** → **Personalization**.

KEYWORDS

Contextual Bandits; User Modeling; Graph Neural Networks

ACM Reference Format:
Yunzhe Qi, Yikun Ban, and Jingrui He. 2023. Graph Neural Bandits. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599371

1 INTRODUCTION

Contextual bandits are a specific type of multi-armed bandit problem where the additional contextual information (contexts) related to arms are available in each round, and the learner intends to refine its selection strategy based on the received arm contexts and rewards. Exemplary applications include online content recommendation, advertising [21, 34], and clinical trials [13, 30]. Meanwhile, collaborative effects among users provide researchers the opportunity to design better recommendation strategies, since the target user's preference can be inferred based on other similar users. Such effects have been studied by many bandit works [4, 15, 16, 22, 23]. Different from the conventional collaborative filtering methods [18, 31], bandit-based approaches focus on more dynamic environments under the online learning settings without pre-training [35], especially when the user interactions are insufficient during the early stage of recommendation (such as dealing with new items or users under the news, short-video recommendation settings), which is also referred to as the "cold-start" problem [21]. In such cases, the exploitation-exploration dilemma [3] inherently exists in the decisions of recommendation.

Existing works for clustering of bandits [4, 6, 15, 16, 22, 23] model the user correlations (collaborative effects) by clustering users into **rigid user groups** and then assigning each user group an estimator to learn the underlying reward functions, combined with an Upper Confidence Bound (UCB) strategy for exploration. However, these works only consider the "coarse-grained" user correlations. To be specific, they assume that users from the same group would share identical preferences, i.e., the users from the same group are compelled to make equal contributions to the final decision (arm selection) with regard to the target user. Such formulation of user correlations ("coarse-grained") fails to comply with many real-world application scenarios, because users within the same group can have similar but subtly different preferences. For instance, under the settings of movie recommendation, although we can allocate two users that "favor" the same movie into a user group, their "favoring levels" can differ significantly: one user could be a die hard fan of this movie while the other user just considers this movie to be average-to-good. In this case, it would not be the best strategy to vaguely consider they share the same preferences and treat them identically. Note that similar concerns also exist even when we switch the binary ranking system to a categorical one (e.g., the rating system out of 5 stars or 10 points), because as long as we model with user categories (i.e., rigid user groups), there will likely be divergence among the users within the same group. Meanwhile, with more fine-sorted user groups, there will be less historical user interactions allocated to each single group because of the decreasing number of associated users. This can lead to the bottlenecks for the group-specific estimators due to the insufficient user interactions. Therefore, given a target user, it is more practical to assume that the rest of the users would impose different levels of collaborative effects on this user.

Motivated by the limitations of existing works, in this paper, we propose a novel framework, named **Graph Neural Bandits (GNB)**, to formulate "fine-grained" user collaborative effects, where the correlation of user pairs is preserved by user graphs. Given a target user, other users are allowed to make different contributions to the

*Both authors contributed equally.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '23, August 6–10, 2023, Long Beach, CA, USA
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599371

final decision based on the strength of their correlation with the target user. However, the user correlations are usually unknown, and the learner is required to estimate them on the fly. Here, the learner aims to approximate the correlation between two users by exploiting their past interactions; on the other hand, the learner can benefit from exploring the potential correlations between users who do not have sufficient interactions, or the correlations that might have changed. In this case, we formulate this problem as the exploitation-exploration dilemma in terms of the user correlations. To solve this new challenge, GNB separately constructs two kinds of user graphs, named "user exploitation graphs" and "user exploration graphs". Then, we apply two individual graph neural networks (GNNs) on the user graphs, to incorporate the collaborative effects in terms of both exploitation and exploration in the decision-making process. Our main contributions are:

- **[Problem Settings]** Different from existing works formulating the "coarse-grained" user correlations by neglecting the divergence within user groups, we introduce a new problem setting to model the "fine-grained" user collaborative effects via user graphs. Here, pair-wise user correlations are preserved to contribute differently to the decision-making process. (**Section** 3)
- **[Proposed Framework]** We propose a framework named GNB, which has the novel ways to build two kinds of user graphs in terms of exploitation and exploration respectively. Then, GNB utilizes GNN-based models for a refined arm selection strategy by leveraging the user correlations encoded in these two kinds of user graphs for the arm selection. (**Section** 4)
- **[Theoretical Analysis]** With standard assumptions, we provide the theoretical analysis showing that GNB can achieve the regret upper bound of complexity $O(\sqrt{T\log(Tn)})$, where T is the number of rounds and n is the number of users. This bound is sharper than the existing related works. (**Section** 5)
- **[Experiments]** Extensive experiments comparing GNB with nine state-of-the-art algorithms are conducted on various data sets with different specifications, which demonstrate the effectiveness of our proposed GNB framework. (**Section** 6)

Due to the page limit, interested readers can refer to the arXiv version of this paper for supplementary contents.

2 RELATED WORKS

Assuming the reward mapping function to be linear, the linear upper confidence bound (UCB) algorithms [1, 3, 11, 21] were first proposed to tackle the exploitation-exploration dilemma. After kernel-based methods [12, 29] were used to tackle the kernel-based reward mapping function under the non-linear settings, neural algorithms [5, 40, 42] have been proposed to utilize neural networks to estimate the reward function and confidence bound. Meanwhile, AGG-UCB [25] adopts GNN to model the arm group correlations. GCN-UCB [28] manages to apply the GNN model to embed arm contexts for the downstream linear regression, and GNN-PE [20] utilizes the UCB based on information gains to achieve exploration for classification tasks on graphs. Instead of using UCB, EE-Net [7] applies a neural network to estimate prediction uncertainty. Nonetheless, all of these works fail to consider the collaboration effects among users under the real-world application scenarios.

To model user correlations, [9, 34] assume the user social graph is known, and apply an ensemble of linear estimators. Without the prior knowledge of user correlations, CLUB [16] introduces the user clustering problem with the graph-connected components, and SCLUB [22] adopts dynamic user sets and set operations, while DynUCB [24] assigns users to their nearest estimated clusters. Then, CAB [15] studies the arm-specific user clustering, and LOCB [4] estimates soft-margin user groups with local clustering. COFIBA [23] utilizes user and arm co-clustering for collaborative filtering. Meta-Ban [6] applies a neural meta-model to adapt to estimated user groups. However, all these algorithms consider rigid user groups, where users from the same group are treated equally with no internal differentiation. Alternatively, we leverage GNNs [10, 14, 17, 27, 32, 33, 38, 39] to learn from the "fine-grained" user correlations and arm contexts simultaneously.

3 GNB: PROBLEM DEFINITION

Suppose there are a total of n users with the user set $\mathcal{U} = \{1, \cdots, n\}$. At each time step $t \in [T]$, the learner will receive a target user $u_t \in \mathcal{U}$ to serve, along with candidate arms $\mathcal{X}_t = \{\boldsymbol{x}_{i,t}\}_{i\in[a]}$, $|\mathcal{X}_t| = a$. Each arm is described by a d-dimensional context vector $\boldsymbol{x}_{i,t} \in \mathbb{R}^d$ with $\|\boldsymbol{x}_{i,t}\|_2 = 1$, and $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$ is also associated with a reward $r_{i,t}$. As the user correlation is one important factor in determining the reward, we define the following reward function:

$$r_{i,t} = h(\boldsymbol{x}_{i,t}, u_t, \mathcal{G}_{i,t}^{(1),*}) + \epsilon_{i,t} \tag{1}$$

where $h(\cdot)$ is the unknown reward mapping function, and $\epsilon_{i,t}$ stands for some zero-mean noise such that $\mathbb{E}[r_{i,t}] = h(\boldsymbol{x}_{i,t}, u_t, \mathcal{G}_{i,t}^{(1),*})$. Here, we have $\mathcal{G}_{i,t}^{(1),*} = (\mathcal{U}, E, W_{i,t}^{(1),*})$ being the **unknown** user graph induced by arm $\boldsymbol{x}_{i,t}$, which encodes the "fine-grained" user correlations in terms of the **expected rewards**. In graph $\mathcal{G}_{i,t}^{(1),*}$, each user $u \in \mathcal{U}$ will correspond to a node; meanwhile, $E = \{e(u,u')\}_{u,u'\in\mathcal{U}}$ refers to the set of edges, and the set $W_{i,t}^{(1),*} = \{w_{i,t}^{(1),*}(u,u')\}_{u,u'\in\mathcal{U}}$ stores the weights for each edge from E. Note that under real-world application scenarios, users sharing the same preference for certain arms (e.g., sports news) may have distinct tastes over other arms (e.g., political news). Thus, we allow each arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$ to induce different user collaborations $\mathcal{G}_{i,t}^{(1),*}$.

Then, motivated by various real applications (e.g., online recommendation with normalized ratings), we consider $r_{i,t}$ to be bounded $r_{i,t} \in [0,1]$, which is standard in existing works (e.g., [4, 6, 15, 16]). Note that as long as $r_{i,t} \in [0,1]$, we do not impose the distribution assumption (e.g., sub-Gaussian distribution) on noise term $\epsilon_{i,t}$.

[Reward Constraint] To bridge user collaborative effects with user preferences (i.e., rewards), we consider the following constraint for reward function in **Eq.** 1. The intuition is that for any two users with comparable user correlations, they will incline to share similar tastes for items. For arm $\boldsymbol{x}_{i,t}$, we consider the difference of expected rewards between any two users $u, u' \in \mathcal{U}$ to be governed by

$$\left|\mathbb{E}[r_{i,t}|u, \boldsymbol{x}_{i,t}] - \mathbb{E}[r_{i,t}|u', \boldsymbol{x}_{i,t}]\right| \leq \Psi\big(\mathcal{G}_{i,t}^{(1),*}[u:], \mathcal{G}_{i,t}^{(1),*}[u':]\big) \tag{2}$$

where $\mathcal{G}_{i,t}^{(1),*}[u:]$ represents the normalized adjacency matrix row of $\mathcal{G}_{i,t}^{(1),*}$ that corresponds to user (node) u, and $\Psi : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ denotes an unknown mapping function. The reward function definition (**Eq.** 1) and the constraint (**Eq.** 2) motivate us to design the GNB framework, to be introduced in Section 4.

Then, we proceed to give the formulation of $\mathcal{G}_{i,t}^{(1),*} = (\mathcal{U}, E, W_{i,t}^{(1),*})$ below. Given arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$, users with strong correlations tend to have similar expected rewards, which will be reflected by $W_{i,t}^{(1),*}$.

Definition 1 (User Correlation for Exploitation). *In round t, for any two users $u, u' \in \mathcal{U}$, their exploitation correlation score $w_{i,t}^{(1),*}(u,u')$ w.r.t. a candidate arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$ is defined as*

$$w_{i,t}^{(1),*}(u,u') = \Psi^{(1)}\big(\mathbb{E}[r_{i,t}|u,\ \boldsymbol{x}_{i,t}],\ \mathbb{E}[r_{i,t}|u',\ \boldsymbol{x}_{i,t}]\big)$$

where $\mathbb{E}[r_{i,t}|u,\ \boldsymbol{x}_{i,t}], i \in [a]$ is the expected reward in terms of the user-arm pair $(u, \boldsymbol{x}_{i,t})$. Given two users $u, u' \in \mathcal{U}$, the function $\Psi^{(1)} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ maps from their expected rewards $\mathbb{E}[r_{i,t}|u,\ \boldsymbol{x}_{i,t}]$ to their user exploitation score $w_{i,t}^{(1),*}(u,u')$.

The edge weight $w_{i,t}^{(1),*}(u,u')$ measures the correlation between the two users' preferences. When $w_{i,t}^{(1),*}(u,u')$ is large, u and u' tend to have the same taste; Otherwise, these two users' preferences will be different in expectation. In this paper, we consider the mapping functions $\Psi^{(1)}$ as the prior knowledge. For example, $\Psi^{(1)}$ can be the radial basis function (RBF) kernel or normalized absolute difference.

[Modeling with User Exploration Graph $\mathcal{G}_{i,t}^{(2),*}$] Unfortunately, $\mathcal{G}_{i,t}^{(1),*}$ is the **unknown** prior knowledge in our problem setting. Thus, the learner has to estimate $\mathcal{G}_{i,t}^{(1),*}$ by exploiting the current knowledge, denoted by $\mathcal{G}_{i,t}^{(1)} = (\mathcal{U}, E, W_{i,t}^{(1)})$, where $W_{i,t}^{(1)} = \{w_{i,t}^{(1)}(u,u')\}_{u,u' \in \mathcal{U}}$ is the estimation of $W_{i,t}^{(1),*}$ based on the function class $\mathcal{F} = \{f_u^{(1)}\}_{u \in \mathcal{U}}$ where $f_u^{(1)}$ is the hypothesis specified to user u. However, greedily exploiting $\mathcal{G}_{i,t}^{(1)}$ may lead to the sub-optimal solution, or overlook some correlations that may only be revealed in the future rounds. Thus, we propose to construct another **user exploration graph** $\mathcal{G}_{i,t}^{(2),*}$ for principled exploration to measure the estimation gap $\mathcal{G}_{i,t}^{(1),*} - \mathcal{G}_{i,t}^{(1)}$, which refers to the uncertainty of the estimation of graph $\mathcal{G}_{i,t}^{(1)}$.

For each arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$, we formulate the user exploration graph $\mathcal{G}_{i,t}^{(2),*} = (\mathcal{U}, E, W_{i,t}^{(2),*})$, with the set of edge weights $W_{i,t}^{(2),*} = \{w_{i,t}^{(2),*}(u,u')\}_{\forall u,u' \in \mathcal{U}}$. Here, $\mathcal{G}_{i,t}^{(2),*}$ models the uncertainty of estimation $\mathcal{G}_{i,t}^{(1)}$ in terms of the true exploitation graph $\mathcal{G}_{i,t}^{(1),*}$, and $\mathcal{G}_{i,t}^{(2),*}$ can be thought as the oracle exploration graph, i.e., "perfect exploration". Then, with the aforementioned hypothesis $f_u^{(1)}(\boldsymbol{x}_{i,t})$ for estimating the expected reward of arm $\boldsymbol{x}_{i,t}$ given u, we introduce the formulation of $\mathcal{G}_{i,t}^{(2),*}$ as the user exploration correlation.

Definition 2 (User Correlation for Exploration). *In round t, given two users $u, u' \in \mathcal{U}$ and an arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$, their underlying exploration correlation score is defined as*

$$w_{i,t}^{(2),*}(u,u') = \Psi^{(2)}\Big(\mathbb{E}[r_{i,t}|u,\boldsymbol{x}_{i,t}] - f_u^{(1)}(\boldsymbol{x}_{i,t}), \mathbb{E}[r_{i,t}|u',\boldsymbol{x}_{i,t}] - f_{u'}(\boldsymbol{x}_{i,t})\Big)$$

with $\mathbb{E}[r_{i,t}|u,\ \boldsymbol{x}_{i,t}] - f_u^{(1)}(\boldsymbol{x}_{i,t}), i \in [a]$ being the potential gain of the estimation $f_u^{(1)}(\boldsymbol{x}_{i,t})$ for the user-arm pair $(u, \boldsymbol{x}_{i,t})$. Here, $f_u^{(1)}(\cdot)$ is the reward estimation function specified to user u, and $\Psi^{(2)} : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$ is the mapping from user potential gains $\mathbb{E}[r_{i,t}|u,\ \boldsymbol{x}_{i,t}] - f_u^{(1)}(\boldsymbol{x}_{i,t})$ to their exploration correlation score.

Here, $w_{i,t}^{(2),*}(u,u')$ is defined based on the potential gain of $f_u^{(1)}(\cdot)$, i.e., $\mathbb{E}[r_{i,t}|u,\ \boldsymbol{x}_{i,t}] - f_u^{(1)}(\boldsymbol{x}_{i,t})$, to measure the estimation uncertainty. Note that our formulation is distinct from the formulation in [7], where they only focus on the single-bandit setting with no user collaborations, and all the users will be treated identically.

As we have discussed, $w_{i,t}^{(2),*}(u,u')$ measures the uncertainty of estimation $w_{i,t}^{(1)}(u,u')$. When $w_{i,t}^{(2),*}(u,u')$ is large, the uncertainty of estimated exploitation correlation, i.e., $w_{i,t}^{(1)}(u,u')$, will also be large, and we should explore them more. Otherwise, we have enough confidence towards $w_{i,t}^{(1)}(u,u')$, and we can exploit $w_{i,t}^{(1)}(u,u')$ in a secure way. Analogous to $\Psi^{(1)}$ in **Def.** 1, we consider the mapping function $\Psi^{(2)}$ as the known prior knowledge.

[Learning Objective] With the received user u_t in each round $t \in [T]$, the learner is expected to recommend an arm $x_t \in \mathcal{X}_t$ (with reward r_t) in order to minimize the cumulative pseudo-regret

$$R(T) = \mathbb{E}[\sum_{t=1}^{T}(r_t^* - r_t)] \tag{3}$$

where we have r_t^* being the reward for the optimal arm, such that $\mathbb{E}[r_t^*|u_t, \mathcal{X}_t] = \max_{\boldsymbol{x}_{i,t} \in \mathcal{X}_t} h(\boldsymbol{x}_{i,t}, u_t, \mathcal{G}_{i,t}^{(1),*})$.

[Comparing with Existing Problem Definitions] The problem definition of existing user clustering works (e.g., [4, 6, 15, 16, 22]) only formulates "coarse-grained" user correlations. In their settings, for a user group $\mathcal{N} \subseteq \mathcal{U}$ with the mapping function $h_{\mathcal{N}}$, all the users in $\mathcal{N}$ are forced to share the same reward mapping given an arm $\boldsymbol{x}_{i,t}$, i.e., $\mathbb{E}[r_{i,t} \mid u, \boldsymbol{x}_{i,t}] = h_{\mathcal{N}}(\boldsymbol{x}_{i,t}), \forall u \in \mathcal{N}$. In contrast, our definition of the reward function enables us to model the pairwise "fine-grained" user correlations by introducing another two important factors u and $\mathcal{G}_{i,t}^{(1),*}$. With our formulation, each user here is allowed to produce different rewards facing the same arm, i.e., $\mathbb{E}[r_{i,t} \mid u, \boldsymbol{x}_{i,t}] = h(\boldsymbol{x}_{i,t}, u, \mathcal{G}_{i,t}^{(1),*}), \forall u \in \mathcal{N}$. Here, with different users u, the corresponding expected reward $h(\boldsymbol{x}_{i,t}, u, \mathcal{G}_{i,t}^{(1),*})$ can be different. Therefore, our definition of the reward function is more generic, and it can also readily generalize to existing user clustering algorithms (with "coarse-grained" user correlations) by allowing each single user group to form an isolated sub-graph in $\mathcal{G}_{i,t}^{(1),*}$ with no connections across different sub-graphs (i.e., user groups).

[Notation] Up to round t, denoting $\mathcal{T}_{u,t} \subseteq [t]$ as the collection of time steps at which user $u \in \mathcal{U}$ has been served, we use $\mathcal{P}_{u,t} = \{(\boldsymbol{x}_\tau, r_\tau)\}_{\tau \in \mathcal{T}_{u,t}}$ to represent the collection of received arm-reward pairs associated with user u, and $T_{u,t} = |\mathcal{T}_{u,t}|$ refers to the corresponding number of rounds. Here, $\boldsymbol{x}_\tau \in \mathcal{X}_\tau, r_\tau \in [0,1]$ separately refer to the chosen arm and actual received reward in round $\tau \in [t]$. Similarly, we use $\mathcal{P}_t = \{(\boldsymbol{x}_\tau, r_\tau)\}_{\tau \in [t]}$ to denote all the past records (i.e., arm-reward pairs), up to round t. For a graph $\mathcal{G}$, we let $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be its adjacency matrix (with self-loops), and $\boldsymbol{D} \in \mathbb{R}^{n \times n}$ refers to the corresponding degree matrix.

4 GNB: PROPOSED FRAMEWORK

The workflow of our proposed GNB framework (Figure 1) consists of four major components: **First**, we derive the estimation for user

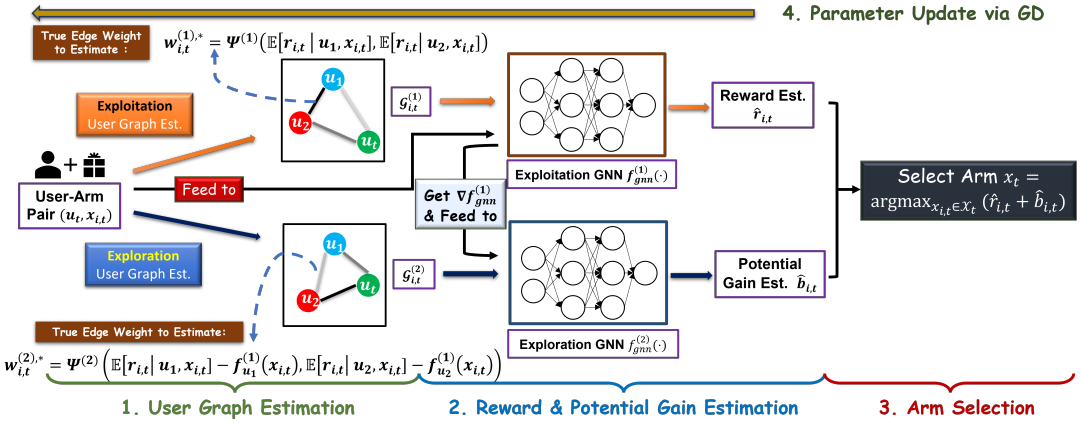

Figure 1: Workflow of the proposed Graph Neural Bandits (GNB) framework.

exploitation graphs $\mathcal{G}_{i,t}^{(1),*}, i \in [a]$ (denoted by $\mathcal{G}_{i,t}^{(1)}$), and the exploration graphs $\mathcal{G}_{i,t}^{(2),*}, i \in [a]$ (denoted by $\mathcal{G}_{i,t}^{(2)}$), to model user correlations in terms of exploitation and exploration respectively; **Second**, to estimate the reward and the potential gain by leveraging the "fine-grained" correlations, we propose the GNN-based models $f_{gnn}^{(1)}(\cdot), f_{gnn}^{(2)}(\cdot)$ to aggregate the correlations of the target user-arm pair on estimated graphs $\mathcal{G}_{i,t}^{(1)}$ and $\mathcal{G}_{i,t}^{(2)}$, respectively; **Third**, we select the arm $\boldsymbol{x}_t$, based on the estimated arm reward and potential gain calculated by our GNN-based models; **Finally**, we train the parameters of GNB using gradient descent (GD) on past records.

4.1 User Graph Estimation with User Networks

Based on the definition of unknown true user graphs $\mathcal{G}_{i,t}^{(1),*}, \mathcal{G}_{i,t}^{(2),*}$ w.r.t. arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$ (**Definition** 1 and 2), we proceed to derive their estimations $\mathcal{G}_{i,t}^{(1)}, \mathcal{G}_{i,t}^{(2)}, i \in [a]$ with individual user networks $f_u^{(1)}$, $f_u^{(2)}, u \in \mathcal{U}$. Afterwards, with these two kinds of estimated user graphs $\mathcal{G}_{i,t}^{(1)}$ and $\mathcal{G}_{i,t}^{(2)}$, we will be able to apply our GNN-based models to leverage the user correlations under the exploitation and the exploration settings. The pseudo-code is presented in **Alg.** 1.

[User Exploitation Network $f_u^{(1)}$] For each user $u \in \mathcal{U}$, we use a neural network $f_u^{(1)}(\cdot) = f_u^{(1)}(\cdot; \Theta_u^{(1)})$ to learn user u's preference for arm $\boldsymbol{x}_{i,t}$, i.e., $\mathbb{E}[r_{i,t}|u, \boldsymbol{x}_{i,t}]$. Following the **Def.** 1, we construct the exploitation graph $\mathcal{G}_{i,t}^{(1)}$ by estimating the user exploitation correlation based on user preferences. Thus, in $\mathcal{G}_{i,t}^{(1)}$, we consider the edge weight of two user nodes u, u' as

$$w_{i,t}^{(1)}(u, u') = \Psi^{(1)}\left(f_u^{(1)}(\boldsymbol{x}_{i,t}), f_{u'}^{(1)}(\boldsymbol{x}_{i,t})\right) \tag{4}$$

where $\Psi^{(1)}(\cdot,\cdot)$ is the mapping function applied in **Def.** 1 (line 16, **Alg.** 1). Here, $f_u^{(1)}(\cdot)$ will be trained by GD with chosen arms $\{\boldsymbol{x}_\tau\}_{\tau \in \mathcal{T}_{u,t}}$ as samples, and received reward $\{r_\tau\}_{\tau \in \mathcal{T}_{u,t}}$ as the labels, where $\mathcal{L}_u^{(1)}(\Theta_u^{(1)}) = \sum_{\tau \in \mathcal{T}_{u,t}} \left|f_u^{(1)}(\boldsymbol{x}_\tau; \Theta_u^{(1)}) - r_\tau\right|^2$ will be the corresponding quadratic loss. Recall that $\boldsymbol{x}_\tau$ and r_τ stand for the chosen arm and the received reward respectively in round τ.

[User Exploration Network $f_u^{(2)}$] Given user $u \in \mathcal{U}$, to estimate the potential gain (i.e., the uncertainty for the reward estimation) $\mathbb{E}[r|u, \boldsymbol{x}_{i,t}] - f_u^{(1)}(\boldsymbol{x}_{i,t})$ for arm $\boldsymbol{x}_{i,t}$, we adopt the user exploration network $f_u^{(2)}(\cdot) = f_u^{(2)}(\cdot; \Theta_u^{(2)})$ inspired by [7]. As it has proved that the confidence interval (uncertainty) of reward estimation can be expressed as a function of network gradients [25, 42], we apply $f_u^{(2)}(\cdot)$ to directly learn the uncertainty with the gradient of $f_u^{(1)}(\cdot)$. Thus, the input of $f_u^{(2)}(\cdot)$ will be the network gradient of $f_u^{(1)}(\cdot)$ given arm $\boldsymbol{x}_{i,t}$, denoted as $\nabla f_u^{(1)}(\boldsymbol{x}_{i,t}) = \nabla_\Theta f_u^{(1)}(\boldsymbol{x}_{i,t}; [\Theta_u^{(1)}]_{t-1})$, where $[\Theta_u^{(1)}]_{t-1}$ refer to the parameters of $f_u^{(1)}$ in round t (before training [line 11, **Alg.** 1]). Analogously, given the estimated user exploration graph $\mathcal{G}_{i,t}^{(2)}$ and two user nodes u, u', we let the edge weight be

$$w_{i,t}^{(2)}(u, u') = \Psi^{(2)}\left(f_u^{(2)}\left(\nabla f_u^{(1)}(\boldsymbol{x}_{i,t})\right), f_{u'}^{(2)}\left(\nabla f_{u'}^{(1)}(\boldsymbol{x}_{i,t})\right)\right) \tag{5}$$

as in line 17, **Alg.** 1, and $\Psi^{(2)}(\cdot,\cdot)$ is the mapping function that has been applied in **Def.** 2. With GD, $f_u^{(2)}(\cdot)$ will be trained with the past gradients of $f_u^{(1)}$, i.e., $\{\nabla f_u^{(1)}(\boldsymbol{x}_\tau)\}_{\tau \in \mathcal{T}_{u,t}}$ as samples; and the potential gain (uncertainty) $\{r_\tau - f_u^{(1)}(\boldsymbol{x}_\tau; [\Theta_u^{(1)}]_{\tau-1})\}_{\tau \in \mathcal{T}_{u,t}}$ as labels. The quadratic loss is defined as $\mathcal{L}_u^{(2)}(\Theta_u^{(2)}) = \sum_{\tau \in \mathcal{T}_{u,t}} \left|f_u^{(2)}(\nabla f_u^{(1)}(\boldsymbol{x}_\tau); \Theta_u^{(2)}) - (r_\tau - f_u^{(1)}(\boldsymbol{x}_\tau; [\Theta_u^{(1)}]_{\tau-1}))\right|^2$.

[Network Architecture] Here, we can apply various architectures for $f_u^{(1)}(\cdot), f_u^{(2)}(\cdot)$ to deal with different application scenarios (e.g., Convolutional Neural Networks [CNNs] for visual content recommendation tasks). For the theoretical analysis and experiments, with user $u \in \mathcal{U}$, we apply separate L-layer ($L \geq 2$) fully-connected (FC) networks as the user exploitation and exploration network

$$f_u(\boldsymbol{\chi}; \Theta_u) = \Theta_L \sigma(\Theta_{L-1} \sigma(\Theta_{L-2} \dots \sigma(\Theta_1 \boldsymbol{\chi}))), \ \sigma := \mathrm{ReLU}(\cdot) \tag{6}$$

with $\Theta_u = [\mathrm{vec}(\Theta_1)^\intercal, \dots, \mathrm{vec}(\Theta_L)^\intercal]^\intercal$ being the vector of trainable parameters. Here, since $f_u^{(1)}(\cdot), f_u^{(2)}(\cdot)$ are both the L-layer FC network (**Eq.** 6), the input $\boldsymbol{\chi}$ can be substituted with either the arm context $\boldsymbol{x}_{i,t}$ or the network gradient $\nabla f_u^{(1)}(\boldsymbol{x}_{i,t})$ accordingly.

[Parameter Initialization] The weight matrices of the first layer are slightly different for two kinds of user networks, as $\Theta_1^{(1)} \in \mathbb{R}^{m \times d}$, $\Theta_1^{(2)} \in \mathbb{R}^{m \times p_u^{(1)}}$ where $p_u^{(1)}$ is the dimensionality of $\Theta_u^{(1)}$. The weight matrix shape for the rest of the $L-1$ layers will be the same for these two kinds of user networks, which are $\Theta_l \in \mathbb{R}^{m \times m}, l \in [2, \cdots, L-1]$, and $\Theta_L \in \mathbb{R}^{1 \times m}$. To initialize $f_u^{(1)}, f_u^{(2)}$, the weight matrix entries for their first $L-1$ layers $\{\Theta_1, \dots \Theta_{L-1}\}$ are drawn from the Gaussian distribution $N(0, 2/m)$, and the entries of the last layer weight matrix Θ_L are sampled from $N(0, 1/m)$.

ALGORITHM 1: Graph Neural Bandits (GNB)

1 **Input:** Number of rounds T, network width m, information propagation hops k. Functions for edge weight estimation $\Psi^{(1)}(\cdot,\cdot), \Psi^{(2)}(\cdot,\cdot) : \mathbb{R}\times\mathbb{R} \mapsto \mathbb{R}$.
2 **Output:** Arm recommendation $\boldsymbol{x}_t$ for each time step t.
3 **Initialization:** Initialize trainable parameters for all models.
4 **for** $t \in \{1, 2, \ldots, T\}$ **do**
5 Receive a user u_t and a set of arm contexts $\mathcal{X}_t = \{\boldsymbol{x}_{i,t}\}_{i\in[a]}$.
6 **for** *each candidate arm* $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$ **do**
7 Construct two kinds of user graphs $\mathcal{G}_{i,t}^{(1)}, \mathcal{G}_{i,t}^{(2)}$ = **Procedure** *Estimating Arm-Specific User Graphs* $(\boldsymbol{x}_{i,t})$. [line 13-20]
8 Compute reward estimation [**Eq.** 10] $\hat{r}_{i,t} = f_{gnn}^{(1)}(\boldsymbol{x}_{i,t},\ \mathcal{G}_{i,t}^{(1)}; [\Theta_{gnn}^{(1)}]_{t-1})$, and the potential gain [**Eq.** 11] $\hat{b}_{i,t} = f_{gnn}^{(2)}(\nabla[f_{gnn}^{(1)}]_{i,t},\ \mathcal{G}_{i,t}^{(2)}; [\Theta_{gnn}^{(2)}]_{t-1})$.
9 **end**
10 Play arm $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}_{i,t}\in\mathcal{X}_t} \left(\hat{r}_{i,t} + \hat{b}_{i,t}\right)$, and observe its true reward r_t.
11 Train the user networks $f_{u_t}^{(1)}(\cdot;\Theta_{u_t}^{(1)})$, $f_{u_t}^{(2)}(\cdot;\Theta_{u_t}^{(2)})$ and GNN models $f_{gnn}^{(1)}(\cdot;\Theta_{gnn}^{(1)})$, $f_{gnn}^{(2)}(\cdot;\Theta_{gnn}^{(2)})$ with GD.
12 **end**
13 **Procedure** *Estimating Arm-Specific User Graphs* $(\boldsymbol{x}_{i,t})$
14 Initialize arm graphs $\mathcal{G}_{i,t}^{(1)}, \mathcal{G}_{i,t}^{(2)}$.
15 **for** *each user pair* $(u, u') \in \mathcal{U}\times\mathcal{U}$ **do**
16 For edge weight $w_{i,t}^{(1)}(u,u') \in W_{i,t}^{(1)}$, update $w_{i,t}^{(1)}(u,u') = \Psi^{(1)}\big(f_u^{(1)}(\boldsymbol{x}_{i,t}), f_{u'}^{(1)}(\boldsymbol{x}_{i,t})\big)$. [**Eq.** 4]
17 For edge weight $w_{i,t}^{(2)}(u,u') \in W_{i,t}^{(2)}$, based on the [**Eq.** 5], update $w_{i,t}^{(2)}(u,u') = \Psi^{(2)}\big(f_u^{(2)}\big(\nabla f_u^{(1)}(\boldsymbol{x}_{i,t})\big), f_{u'}^{(2)}\big(\nabla f_{u'}^{(1)}(\boldsymbol{x}_{i,t})\big)\big)$.
18 **end**
19 Return user graphs $\mathcal{G}_{i,t}^{(1)}, \mathcal{G}_{i,t}^{(2)}$.
20 **end**

4.2 Achieving Exploitation and Exploration with GNN Models on Estimated User Graphs

With derived user exploitation graphs $\mathcal{G}_{i,t}^{(1)}$, and exploitation graphs $\mathcal{G}_{i,t}^{(2)}$, $i \in [a]$, we apply two GNN models to separately estimate the arm reward and potential gain for a refined arm selection strategy, by utilizing the past interaction records with all the users.

4.2.1 The Exploitation GNN $f_{gnn}^{(1)}(\cdot)$. In round t, with the estimated user exploitation graph $\mathcal{G}_{i,t}^{(1)}$ for arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$, we apply the exploitation GNN model $f_{gnn}^{(1)}(\boldsymbol{x}_{i,t}, \mathcal{G}_{i,t}^{(1)}; \Theta_{gnn}^{(1)})$ to collaboratively estimate the arm reward $\widehat{r}_{i,t}$ for the received user $u_t \in \mathcal{U}$. We start from learning the aggregated representation for k hops, as

$$\boldsymbol{H}_{agg} = \sigma\big((S_{i,t}^{(1)})^k \cdot (\boldsymbol{X}_{i,t}\boldsymbol{\Theta}_{agg}^{(1)})\big) \in \mathbb{R}^{n\times m} \tag{7}$$

where $S_{i,t}^{(1)} = (\boldsymbol{D}_{i,t}^{(1)})^{-\frac{1}{2}}\boldsymbol{A}_{i,t}^{(1)}(\boldsymbol{D}_{i,t}^{(1)})^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix of $\mathcal{G}_{i,t}^{(1)}$, and σ represents the ReLU activation function. With m being the network width, we have $\boldsymbol{\Theta}_{agg}^{(1)} \in \mathbb{R}^{nd\times m}$ as the trainable weight matrix. After propagating the information for k hops over the user graph, each row of $\boldsymbol{H}_{agg}$ corresponds to the aggregated m-dimensional hidden representation for one specific user-arm pair $(u, \boldsymbol{x}_{i,t}), u \in \mathcal{U}$. Here, the propagation of multi-hop information can provide a global perspective over the users, since it also involves the neighborhood information of users' neighbors [33, 41]. To achieve this, we have the embedding matrix $\boldsymbol{X}_{i,t}$ (in **Eq.** 7) for arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t, i \in [a]$ being

$$\boldsymbol{X}_{i,t} = \begin{pmatrix} \boldsymbol{x}_{i,t}^\intercal & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{x}_{i,t}^\intercal & \cdots & \boldsymbol{0} \\ \vdots & & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{x}_{i,t}^\intercal \end{pmatrix} \in \mathbb{R}^{n\times nd} \tag{8}$$

which partitions the weight matrix $\boldsymbol{\Theta}_{gnn}^{(1)}$ for different users. In this way, it is designed to generate individual representations w.r.t. each user-arm pair $(u, \boldsymbol{x}_{i,t}), u \in \mathcal{U}$ before the k-hop propagation (i.e., multiplying with $(S_{i,t}^{(1)})^k$), which correspond to the rows of the matrix multiplication $(\boldsymbol{X}_{i,t}\boldsymbol{\Theta}_{agg}^{(1)}) \in \mathbb{R}^{n\times m}$.

Afterwards, with $\boldsymbol{H}_0 = \boldsymbol{H}_{agg}$, we feed the aggregated representations into the L-layer ($L \geq 2$) FC network, represented by

$$\begin{aligned} &\boldsymbol{H}_l = \sigma(\boldsymbol{H}_{l-1}\cdot\boldsymbol{\Theta}_l^{(1)}) \in \mathbb{R}^{n\times m},\ l \in [L-1], \\ &\widehat{\boldsymbol{r}}_{all}(\boldsymbol{x}_{i,t}) = \boldsymbol{H}_{L-1}\cdot\boldsymbol{\Theta}_L^{(1)} \in \mathbb{R}^n \end{aligned} \tag{9}$$

where $\widehat{\boldsymbol{r}}_{all}(\boldsymbol{x}_{i,t}) \in \mathbb{R}^n$ represents the reward estimation for all the users in $\mathcal{U}$, given the arm $\boldsymbol{x}_{i,t}$. Given the target user u_t in round t, the reward estimation for the user-arm pair $(u_t, \boldsymbol{x}_{i,t})$ would be the corresponding element in $\widehat{\boldsymbol{r}}_{all}$ (line 8, **Alg.** 1), represented by:

$$\widehat{r}_{i,t} = f_{gnn}^{(1)}(\boldsymbol{x}_{i,t},\ \mathcal{G}_{i,t}^{(1)}; [\Theta_{gnn}^{(1)}]_{t-1}) = [\widehat{\boldsymbol{r}}_{all}(\boldsymbol{x}_{i,t})]_{u_t} \tag{10}$$

where $\boldsymbol{\Theta}_{gnn}^{(1)} = [\mathrm{vec}(\boldsymbol{\Theta}_{agg}^{(1)})^\intercal, \mathrm{vec}(\boldsymbol{\Theta}_1^{(1)})^\intercal, \ldots, \mathrm{vec}(\boldsymbol{\Theta}_L^{(1)})^\intercal]^\intercal \in \mathbb{R}^p$ represent the trainable parameters of the exploitation GNN model, and we have $[\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}$ being the parameters $\boldsymbol{\Theta}_{gnn}^{(1)}$ in round t (before training [line 11, **Alg.** 1]). Here, the weight matrix shapes are $\boldsymbol{\Theta}_l^{(1)} \in \mathbb{R}^{m\times m}, l \in [1, \cdots, L-1]$, and the L-th layer $\boldsymbol{\Theta}_L^{(1)} \in \mathbb{R}^m$.

[Training $f_{gnn}^{(1)}$ with GD] The exploitation GNN $f_{gnn}^{(1)}(\cdot)$ will be trained with GD based on the received records $\mathcal{P}_t$. Then, we apply the quadratic loss function based on reward predictions $\{f_{gnn}^{(1)}(\boldsymbol{x}_\tau,\ \mathcal{G}_\tau^{(1)}; \boldsymbol{\Theta}_{gnn}^{(1)})\}_{\tau\in[t]}$ of chosen arms $\{\boldsymbol{x}_\tau\}_{\tau\in[t]}$, the actual received rewards $\{r_\tau\}_{\tau\in[t]}$, and the user exploitation graph $\mathcal{G}_\tau^{(1)}$ for chosen arms $\boldsymbol{x}_\tau, \tau \in [t]$. The corresponding quadratic loss will be $\mathcal{L}_{gnn}^{(1)}(\Theta_{gnn}^{(1)}) = \sum_{\tau\in[t]} \left|f_{gnn}^{(1)}(\boldsymbol{x}_\tau, \mathcal{G}_\tau^{(1)}; \boldsymbol{\Theta}_{gnn}^{(1)}) - r_\tau\right|^2$.

[Connection with the Reward Function Definition (*Eq.* 1) and Constraint (*Eq.* 2)] The existing works (e.g.,[2]) show that the FC network is naturally Lipschitz continuous with respect to the input when width m is sufficiently large. Thus, for GNB, with aggregated hidden representations $\boldsymbol{H}_{agg}$ being the input to the FC network (**Eq.** 9), we will have the difference of reward estimations $\widehat{r}_{i,t}$ bounded by the distance of rows in matrix $\boldsymbol{H}_{agg}$ (i.e., aggregated hidden representations). Here, given $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$ and users $u_1, u_2 \in \mathcal{U}$,

their estimated reward difference $|[\widehat{r}_{all}(\boldsymbol{x}_{i,t})]_{u_1} - [\widehat{r}_{all}(\boldsymbol{x}_{i,t})]_{u_2}|$ can be bounded by the distance of the corresponding rows in $\boldsymbol{S}_{i,t}$ (i.e., $\|\boldsymbol{S}_{i,t}^{(1)}[u_1:] - \boldsymbol{S}_{i,t}^{(1)}[u_2:]\|$) given the exploitation GNN model. This design matches our definition and the constraint in **Eq.** 1-2.

4.2.2 The Exploration GNN $f_{gnn}^{(2)}(\cdot)$. Given a candidate arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$, to achieve adaptive exploration with the user exploration collaborations encoded in $\mathcal{G}_{i,t}^{(2)}$, we apply a second GNN model $f_{gnn}^{(2)}(\cdot)$ to evaluate the potential gain $\widehat{b}_{i,t}$ for the reward estimation $\widehat{r}_{i,t} = f_{gnn}^{(1)}(\boldsymbol{x}_{i,t},\ \mathcal{G}_{i,t}^{(1)}; [\Theta_{gnn}^{(1)}]_{t-1})$ [**Eq.** 10], denoted by

$$\widehat{b}_{i,t} = f_{gnn}^{(2)}(\nabla[f_{gnn}^{(1)}]_{i,t}, \mathcal{G}_{i,t}^{(2)}; [\Theta_{gnn}^{(2)}]_{t-1}) = [\widehat{\boldsymbol{b}}_{all}(\boldsymbol{x}_{i,t})]_{u_t}. \quad (11)$$

The architecture of $f_{gnn}^{(2)}(\cdot)$ can also be represented by **Eq.** 7-10. While $f_{gnn}^{(1)}(\cdot), f_{gnn}^{(2)}(\cdot)$ have the same network width m and number of layers L, the dimensionality of $\boldsymbol{\Theta}_{agg}^{(1)} \in \mathbb{R}^{nd\times m}, \boldsymbol{\Theta}_{agg}^{(2)} \in \mathbb{R}^{np\times m}$ is different. Analogously, $\widehat{\boldsymbol{b}}_{all}(\boldsymbol{x}_{i,t}) \in \mathbb{R}^n$ is the potential gain estimation for all the users in $\mathcal{U}$, w.r.t. arm $\boldsymbol{x}_{i,t}$ and the exploitation GNN $f_{gnn}^{(1)}(\cdot)$. Here, the inputs are user exploration graph $\mathcal{G}_{i,t}^{(2)}$, and the gradient of the exploitation GNN, represented by $\nabla[f_{gnn}^{(1)}]_{i,t} = \nabla_{\Theta_{gnn}^{(1)}} f_{gnn}^{(1)}(\boldsymbol{x}_{i,t}, \mathcal{G}_{i,t}^{(1)}; [\Theta_{gnn}^{(1)}]_{t-1})$. The exploration GNN $f_{gnn}^{(2)}(\cdot)$ leverages the user exploration graph $\mathcal{G}_{i,t}^{(2)}$ and the gradients of $f_{gnn}^{(1)}(\cdot)$ to estimate the uncertainty of reward estimations, which stands for our adaptive exploration strategy (downward or upward exploration). More discussions are in Appendix **Section** B.

[Training $f_{gnn}^{(2)}$ with GD] Similar to $f_{gnn}^{(1)}$, we train $f_{gnn}^{(2)}$ with GD by minimizing the quadratic loss, denoted by $\mathcal{L}_{gnn}^{(2)}(\Theta_{gnn}^{(2)}) = \sum_{\tau\in[t]} \big|f_{gnn}^{(2)}(\nabla[f_{gnn}^{(1)}]_\tau, \mathcal{G}_\tau^{(2)}; \Theta_{gnn}^{(2)}) - \big(r_\tau - f_{gnn}^{(1)}(\boldsymbol{x}_\tau, \mathcal{G}_\tau^{(1)}; [\Theta_{gnn}^{(1)}]_{\tau-1})\big)\big|^2$. This is defined to measure the difference between the estimated potential gains $\{f_{gnn}^{(2)}(\nabla[f_{gnn}^{(1)}]_\tau, \mathcal{G}_\tau^{(2)}; \Theta_{gnn}^{(2)})\}_{\tau\in[t]}$, and the corresponding labels $\{r_\tau - f_{gnn}^{(1)}(\boldsymbol{x}_\tau,\ \mathcal{G}_\tau^{(1)}; [\Theta_{gnn}^{(1)}]_{\tau-1})\}_{\tau\in[t]}$.

Remark 4.1 (Reducing Input Complexity)**.** The input of $f_{gnn}^{(2)}(\cdot)$ is the gradient $\nabla_\Theta f_{gnn}^{(1)}(\boldsymbol{x})$ given the arm $\boldsymbol{x}$, and its dimensionality is naturally $p = (nd \times m) + (L-1) \times m^2 + m$, which can be a large number. Inspired by CNNs, e.g., [26], we apply the **average pooling** to approximate the original gradient vector in practice. In this way, we can save the running time and reduce space complexity simultaneously. Note this approach is also compatible with user networks in Subsection 4.1. To prove its effectiveness, we will apply this approach on GNB for all the experiments in Section 6.

Remark 4.2 (Working with Large Systems)**.** When facing a large number of users, we can apply the "approximated user neighborhood" to reduce the running time in practice. Given user graphs $\mathcal{G}_{i,t}^{(1)}, \mathcal{G}_{i,t}^{(2)}$ in terms of arm $\boldsymbol{x}_{i,t}$, we derive approximated user neighborhoods $\widetilde{\mathcal{N}}^{(1)}(u_t),\ \widetilde{\mathcal{N}}^{(2)}(u_t) \subset \mathcal{U}$ for the target user u_t, with size $|\widetilde{\mathcal{N}}^{(1)}(u_t)| = |\widetilde{\mathcal{N}}^{(2)}(u_t)| = \widetilde{n}$, where $\widetilde{n} << n$. For instance, we can choose $\widetilde{n}$ "representative users" (e.g., users posting high-quality reviews on e-commerce platforms) to form $\widetilde{\mathcal{N}}^{(1)}(u_t), \widetilde{\mathcal{N}}^{(2)}(u_t)$, and apply the corresponding approximated user sub-graphs for downstream GNN models to reduce the computation cost and space cost in practice. Related experiments are provided in Subsection 6.3.

[Parameter Initialization] For the parameters of both GNN models (i.e., $\Theta_{gnn}^{(1)}$ and $\Theta_{gnn}^{(2)}$), the matrix entries of the aggregation weight matrix $\boldsymbol{\Theta}_{agg}$ and the first $L-1$ FC layers $\{\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_{L-1}\}$ are drawn from the Gaussian distribution $N(0, 2/m)$. Then, for the last layer weight matrix $\boldsymbol{\Theta}_L$, we draw its entries from $N(0, 1/m)$.

4.2.3 Arm Selection Mechanism and Model Training. In round t, with the current parameters $[\Theta_{gnn}^{(1)}]_{t-1}, [\Theta_{gnn}^{(2)}]_{t-1}$ for GNN models before model training, the selected arm is chosen as

$$\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}_{i,t}\in\mathcal{X}_t} \Big[f_{gnn}^{(1)}(\boldsymbol{x}_{i,t},\ \mathcal{G}_{i,t}^{(1)}; [\Theta_{gnn}^{(1)}]_{t-1}) + f_{gnn}^{(2)}(\nabla_{\Theta_{gnn}^{(1)}} f_{gnn}^{(1)}(\boldsymbol{x}_{i,t},\ \mathcal{G}_{i,t}^{(1)}; [\Theta_{gnn}^{(1)}]_{t-1}),\ \mathcal{G}_{i,t}^{(2)}; [\Theta_{gnn}^{(2)}]_{t-1}) \Big]$$

based on the estimated reward and potential gain (line 10, **Alg.** 1). After receiving reward r_t, we update user networks $f_{u_t}^{(1)}, f_{u_t}^{(2)}$ of user u_t, and GNN models based on GD (line 11, **Alg.** 1).

5 THEORETICAL ANALYSIS

In this section, we present the theoretical analysis for the proposed GNB. Here, we consider each user $u \in \mathcal{U}$ to be evenly served T/n rounds up to time step T, i.e., $|\mathcal{T}_{u,t}| = T_{u,t} = T/n$, which is standard in closely related works (e.g., [4, 16]). To ensure the neural models are able to efficiently learn the underlying reward mapping, we have the following assumption regarding the arm separateness.

Assumption 5.1 (ρ-Separateness of Arms). *After a total of T rounds, for every pair $\boldsymbol{x}_{i,t}, \boldsymbol{x}_{i',t'}$ with $t, t' \in [T]$ and $i, i' \in [a]$, if $(t,i) \neq (t',i')$, we have $\|\boldsymbol{x}_{i,t} - \boldsymbol{x}_{i',t'}\|_2 \geq \rho$ where $0 < \rho \leq O(\frac{1}{L})$.*

Note that the above assumption is mild, and it has been commonly applied in existing works on neural bandits [7] and over-parameterized neural networks [2]. Since L can be manually set (e.g., $L = 2$), we can easily satisfy the condition $0 < \rho \leq O(\frac{1}{L})$ as long as no two arms are identical. Meanwhile, Assumption 4.2 in [42] and Assumption 3.4 from [40] also imply that no two arms are the same, and they measure the arm separateness in terms of the minimum eigenvalue λ_0 (with $\lambda_0 > 0$) of the Neural Tangent Kernel (NTK) [19] matrix, which is comparable with our Euclidean separateness ρ. Based on **Def.** 1 and **Def.** 2, given an arm $\boldsymbol{x}_{i,t} \in \mathcal{X}_t$, we denote the adjacency matrices as $\boldsymbol{A}_{i,t}^{(1),*}$ and $\boldsymbol{A}_{i,t}^{(2),*}$ for the true arm graphs $\mathcal{G}_{i,t}^{(1),*}$, $\mathcal{G}_{i,t}^{(2),*}$. For the sake of analysis, given any adjacency matrix $\boldsymbol{A}$, we derive the normalized adjacency matrix $\boldsymbol{S}$ by scaling the elements of $\boldsymbol{A}$ with $1/n$. We also set the propagation parameter $k = 1$, and define the mapping functions $\Psi^{(1)}(a,b), \Psi^{(2)}(a,b) := \exp(-\|a-b\|)$ given the inputs a, b. Note that our results can be readily generalized to other mapping functions with the Lipschitz-continuity properties.

Next, we proceed to show the regret bound $R(T)$ after T time steps [**Eq.** 3]. Here, the following **Theorem** 5.2 covers two types of error: (1) the estimation error of user graphs; and (2) the approximation error of neural models. Let η_1, J_1 be the learning rate and GD training iterations for user networks, and η_2, J_2 denote the learning rate and iterations for GNN models. The proof sketch of **Theorem** 5.2 is presented in Appendix **Section** C.

Theorem 5.2 (Regret Bound). *Define $\delta \in (0,1)$, $0 < \xi_1, \xi_2 \leq O(1/T)$ and $0 < \rho \leq O(1/L)$, $c_\xi > 0$, $\xi_L = (c_\xi)^L$. With the user*

networks defined in ***Eq.*** *6 and the GNN models defined in* ***Eq.*** *7–9 with L FC-layers, let* $m \geq \Omega\big(Poly(T, L, a, \frac{1}{\rho}) \cdot \xi_L \log(1/\delta)\big)$, $n \geq \widetilde{\Omega}\big(Poly(L)\big)$. *Set the learning rates and GD iterations*

$$\eta_1 = \Theta\big(\frac{\rho}{m \cdot Poly(T, n, a, L)}\big), \quad \eta_2 = \Theta\big(\frac{\rho}{m \cdot Poly(T, a, L)}\big),$$

$$J_1 = \Theta\big(\frac{Poly(T, n, a, L)}{\rho \cdot \delta^2} \cdot \log(\frac{1}{\xi_1})\big),\ J_2 = \Theta\big(\frac{Poly(T, a, L)}{\rho \cdot \delta^2} \cdot \log(\frac{1}{\xi_2})\big).$$

Then, following ***Algorithm*** *1, with probability at least* $1 - \delta$, *the* T*-round pseudo-regret* $R(T)$ *of GNB can be bounded by*

$$R(T) \leq \sqrt{T} \cdot \big(O(L\xi_L) \cdot \sqrt{2\log(\frac{Tn \cdot a}{\delta})}\big) + \sqrt{T} \cdot O(L) + O(\xi_L) + O(1).$$

Recall that L is generally a small integer (e.g., we set $L = 2$ for experiments in **Section** 6), which makes the condition on number of users reasonable as n is usually a gigantic number in real-world recommender systems. We also have m to be sufficiently large under the over-parameterization regime, which makes the regret bound hold. Here, we have the following remarks.

Remark 5.3 (Dimension terms $d, \tilde{d}$)**.** Existing neural single-bandit (i.e., with no user collaborations) algorithms [25, 40, 42] keep the bound of $O(\tilde{d}\sqrt{T}\log(T))$ based on gradient mappings and ridge regression. $\tilde{d}$ is the effective dimension of the NTK matrix, which can grow along with the number of parameters p and rounds T. The linear user clustering algorithms (e.g., [4, 15, 22]) have the bound $O(d\sqrt{T}\log(T))$ with context dimension d, which can be large with a high-dimensional context space. Alternatively, the regret bound in **Theorem** 5.2 is free of terms d and $\tilde{d}$, as we apply the generalization bounds of over-parameterized networks instead [2, 8], which are unrelated to dimension terms d or $\tilde{d}$.

Remark 5.4 (From $\sqrt{n}$ to $\sqrt{\log(n)}$)**.** With n being the number of users, existing user clustering works (e.g., [4, 6, 16, 22]) involve a $\sqrt{n}$ factor in the regret bound as the cost of leveraging user collaborative effects. Instead of applying separate estimators for each user group, our proposed GNB only ends up with a $\sqrt{\log(n)}$ term to incorporate user collaborations by utilizing dual GNN models for estimating the arm rewards and potential gains correspondingly.

Remark 5.5 (Arm i.i.d. Assumption)**.** Existing clustering of bandits algorithms (e.g., [6, 15, 16, 22]) and the single-bandit algorithm EE-Net [7] typically require the arm i.i.d. assumption for the theoretical analysis, which can be strong since the candidate arm pool $\mathcal{X}_t, t \in [T]$ is usually conditioned on the past records. Here, instead of using the regression-based analysis as in existing works, our proof of **Theorem** 5.2 applies the martingale-based analysis instead to help alleviate this concern.

6 EXPERIMENTS

In this section, we evaluate the proposed GNB framework on multiple real data sets against nine state-of-the-art algorithms, including the linear user clustering algorithms: (1) **CLUB** [16], (2) **SCLUB** [22], (3) **LOCB** [4], (4) **DynUCB** [24], (5) **COFIBA** [23]; the neural single-bandit algorithms: (6) **Neural-Pool** adopts one single Neural-UCB [42] model for all the users with the UCB-type exploration strategy; (7) **Neural-Ind** assigns each user with their own separate Neural-UCB [42] model; (8) **EE-Net** [7]; and, the neural user clustering algorithm: (9) **Meta-Ban** [6]. We leave the implementation details and data set URLs to Appendix **Section** A.

6.1 Real Data Sets

In this section, we compare the proposed GNB with baselines on six data sets with different specifications.

[Recommendation Data Sets] "MovieLens rating dataset" includes reviews from 1.6×10^5 users towards 6×10^4 movies. Here, we select 10 genome-scores with the highest variance across movies to generate the movie features $\boldsymbol{v}_i \in \mathbb{R}^d, d = 10$. The user features $\boldsymbol{v}_u \in \mathbb{R}^d, u \in \mathcal{U}$ are obtained through singular value decomposition (SVD) on the rating matrix. We use K-means to divide users into $n = 50$ groups based on $\boldsymbol{v}_u$, and consider each group as a node in user graphs. In each round t, a user u_t will be drawn from a randomly sampled group. For the candidate pool $\mathcal{X}_t$ with $|\mathcal{X}_t| = a = 10$ arms, we choose one bad movie ($\leq$ two stars, out of five) rated by u_t with reward 1, and randomly pick the other 9 good movies with reward 0. The target here is to help users avoid bad movies. For "Yelp" data set, we build the rating matrix w.r.t. the top 2,000 users and top 10,000 arms with the most reviews. Then, we use SVD to extract the 10-dimensional representation for each user and restaurant. For an arm, if the user's rating $\geq$ three stars (out of five stars), the reward is set to 1; otherwise, the reward is 0. Similarly, we apply K-means to obtain $n = 50$ groups based on user features. In round t, a target u_t, is sampled from a randomly selected group. For $\mathcal{X}_t$, we choose one good restaurant rated by u_t with reward 1, and randomly pick the other 9 bad restaurants with reward 0.

[Classification Data Sets] We also perform experiments on four real classification data sets under the recommendation settings, which are "MNIST" (with the number of classes $C = 10$), "Shuttle" ($C = 7$), the "Letter" ($C = 26$), and the "Pendigits" ($C = 10$) data sets. Each class will correspond to one node in user graphs. Similar to previous works [6, 42], given a sample $\boldsymbol{x} \in \mathbb{R}^d$, we transform it into C different arms, denoted by $\boldsymbol{x}_1 = (\boldsymbol{x}, 0, \ldots, 0), \boldsymbol{x}_2 = (0, \boldsymbol{x}, \ldots, 0), \ldots, \boldsymbol{x}_C = (0, 0, \ldots, \boldsymbol{x}) \in \mathbb{R}^{d+C-1}$ where we add $C - 1$ zero digits as the padding. The received reward $r_t = 1$ if we select the arm of the correct class, otherwise $r_t = 0$.

6.1.1 Experiment Results. Figure 2 illustrates the cumulative regret results on the six data sets, and the red shade represents the standard deviation of GNB. Here, our proposed GNB manages to achieve the best performance against all these strong baselines. Since the MovieLens data set involves real arm features (i.e., genome-scores), the performance of different algorithms on the MovieLens data set tends to have larger divergence. Note that due to the inherent noise within these two recommendation data sets, we can observe the "linear-like" regret curves, which are common as in existing works (e.g., [6]). In this case, to show the model convergence, we will present the convergence results for the recommendation data sets in Appendix **Subsec.** A.3. Among the baselines, the neural algorithms (Neural-Pool, EE-Net, Meta-Ban) generally perform better than linear algorithms due to the representation power of neural networks. However, as Neural-Ind considers no correlations among users, it tends to perform the worst among all baselines on these two data sets. For classification data sets, Meta-Ban performs better than the other baselines by modeling user (class) correlations with the neural network. Since the classification data sets generally involve complex reward mapping functions, it can lead to the poor performances of linear algorithms. Our proposed GNB outperforms the baselines by modeling fine-grained correlations and utilizing

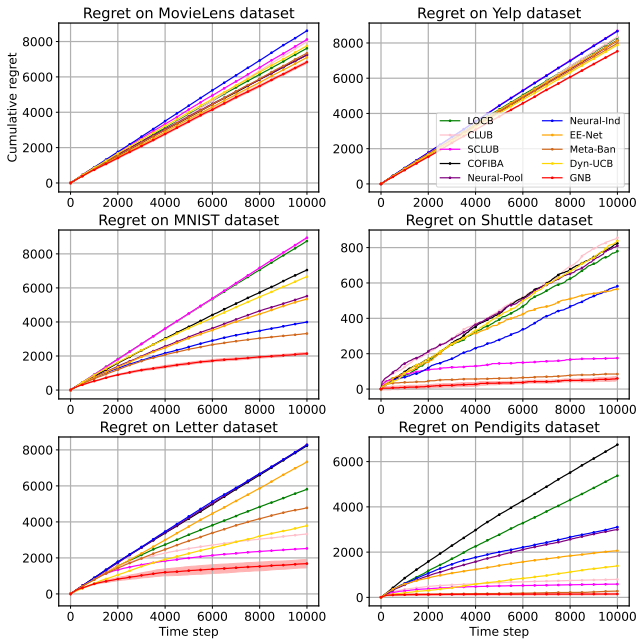

Figure 2: Cumulative regrets on the recommendation and classification data sets.

the adaptive exploration strategy simultaneously. In addition, GNB only takes at most 75% of Meta-Ban's running time for experiments, since Meta-Ban needs to train the framework individually for each arm before making predictions. We will discuss more about the running time in **Subsec.** 6.5.

6.2 Effects of Propagation Hops k

We also include the experiments on the MovieLens data set with 100 users to further investigate the effects of the propagation hyper-parameter k. Recall that given two input vectors w, v, we apply the RBF kernel as the mapping functions $\Psi^{(1)}(w,v) = \Psi^{(2)}(w,v) = \exp(-\gamma \cdot \|w-v\|^2)$ where γ is the kernel bandwidth. The experiment results are shown in the **Table** 1 below, and the value in the brackets "[]" is the element standard deviation of the normalized adjacency matrix of user exploitation graphs.

	Bandwidth γ			
k	0.1	1	2	5
1	7276 $[1.6 \times 10^{-4}]$	7073 $[1.4 \times 10^{-3}]$	7151 $[2.2 \times 10^{-3}]$	7490 $[3.9 \times 10^{-3}]$
2	6968 $[1.0 \times 10^{-4}]$	6966 $[7.7 \times 10^{-4}]$	7074 $[1.3 \times 10^{-3}]$	7087 $[2.5 \times 10^{-3}]$
3	7006 $[7.1 \times 10^{-5}]$	7018 $[7.0 \times 10^{-4}]$	6940 $[1.2 \times 10^{-3}]$	7167 $[1.9 \times 10^{-3}]$

Table 1: Cumulative regrets on MovieLens dataset with 100 users (different k / kernel bandwidth). The value in the brackets "[]" is the element standard deviation of the corresponding normalized adjacency matrix.

Here, increasing the value of parameter k will generally make the normalized adjacency matrix elements "smoother", as we can see from the decreasing standard deviation values. This matches the low-pass nature of graph multi-hop feature propagation [33]. With larger k values, GNB will be able to propagate the information for mores hops. In contrast, with a smaller k value, it is possible that the target user will be "heavily influenced" by only several specific users. However, overly large k values can also lead to the "over-smoothing" problem [36, 37], which can impair the model performance. Therefore, the practitioner may need to choose k value properly under different application scenarios.

6.3 Effects of the Approximated Neighborhood

In this subsection, we conduct experiments to support our claim that applying approximated user neighborhoods is a feasible solution to reduce the computational cost, when facing the increasing number of users (**Remark** 4.2). We consider three scenarios where the number of users $n \in \{200, 300, 500\}$. Meanwhile, we let the size of the approximated user neighborhood $\tilde{\mathcal{N}}^{(1)}(u_t), \tilde{\mathcal{N}}^{(2)}(u_t)$ fix to $\tilde{n} = |\tilde{\mathcal{N}}^{(1)}(u_t)| = |\tilde{\mathcal{N}}^{(2)}(u_t)| = 50$ for all these three experiment settings, and the neighborhood users are sampled from the user pool $\mathcal{U}$ in the experiments.

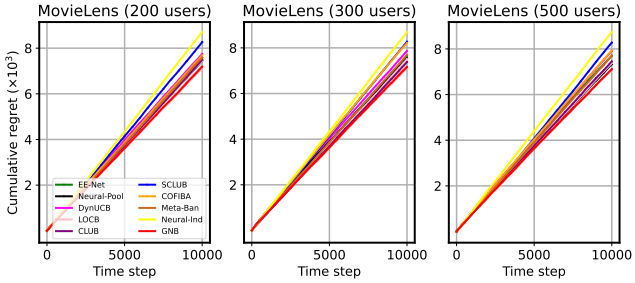

Figure 3: Cumulative regrets for different number of users with approximated user neighborhood (MovieLens data set).

	Avg. regret per round at different t				
Algorithm	2000	4000	6000	8000	10000
CLUB	0.7691	0.7513	0.7464	0.7468	0.7496
Neural-Ind	0.8901	0.8808	0.8790	0.8754	0.8741
Neural-Pool	0.7681	0.7526	0.7405	0.7362	0.7334
EE-Net	0.7886	0.7723	0.7642	0.7618	0.7582
Meta-Ban	0.7811	0.7761	0.7754	0.7729	0.7708
GNB ($\widetilde{n} = 50$)	0.7760	0.7245	0.7190	0.7265	0.7140
GNB ($\widetilde{n} = 100$)	0.7406	0.7178	0.7172	0.7110	0.7104
GNB ($\widetilde{n} = 150$)	0.7291	0.7228	0.7129	0.7105	0.7085

Table 2: Running for 10000 rounds and with the number of users $n = 500$ for the MovieLens data set, the comparison between GNB and baselines on average regret per round.

Here, we see that the proposed GNB still outperforms the baselines with increasing number of users. In particular, given a total of 500 users, the approximated neighborhood is only 10% (50 users) of the overall user pool. These results can show that applying approximated user neighborhoods (Remark 4.2) is a practical way to scale-up GNB in real-world application scenarios. In addition, in **Table** 2, we also include the average regret per round across different time steps. With the number of users $n = 500$ on the MovieLens data set, we include the experiments given different numbers of "representative users" $\widetilde{n} \in \{50, 100, 150\}$ to better show the model performance when applying the approximated neighborhood. Here, increasing the number of "representative users" $\widetilde{n}$ can lead to better performances of GNB, while it also shows that a

small number of "representative users" will be enough for GNB to achieve satisfactory performances.

6.4 Effects of the Adaptive Exploration

To show the necessity of the adaptive exploration strategy, we consider an alternative arm selection mechanism (different from line 10, **Alg.** 1) in round $t \in [T]$, as $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}_{i,t} \in \mathcal{X}_t} \left(\hat{r}_{i,t} + \alpha \cdot \hat{b}_{i,t}\right)$, given the estimated reward and potential gain. Here, we introduce an additional parameter $\alpha \in [0, 1]$ as the exploration coefficient to control the exploration levels (i.e., larger α values will lead to higher levels of exploration). Here, we show the experiment results with $\alpha \in \{0, 0.1, 0.3, 0.7, 1.0\}$ on the "MNIST" and "Yelp" data sets.

	Regret results with different α values				
Dataset	$\alpha = 0$	$\alpha = 0.1$	$\alpha = 0.3$	$\alpha = 0.7$	$\alpha = 1$
Yelp	7612	7444	7546	7509	7457
MNIST	2323	2110	2170	2151	2141

Table 3: Results with different exploration coefficients α.

In Table 3, regarding the results of the "Yelp" data set, although the performances of GNB do not differ significantly with different α values, our adaptive exploration strategy based on user exploration graphs is still helpful to improve GNB's performances, which is validated by the fact that setting $\alpha \in (0, 1]$ will lead to better results compared with the situation where no exploration strategies are involved (setting $\alpha = 0$). On the other hand, as for the results of the "MNIST" data set, different α values will lead to relatively divergent results. One reason can be that with larger context dimension d in the "MNIST" data set, the underlying reward mapping inclines to be more complicated compared with that of the "Yelp" data set. In this case, leveraging the exploration correlations will be more beneficial. Thus, the adaptive exploration strategy is necessary to improve the performance of GNB by estimating the potential gains based on "fine-grained" user (class) correlations.

6.5 Running Time vs. Performance

In Figure 4, we show the results in terms of cumulative regret [y-axis, smaller = better] and running time [x-axis, smaller = better]. Additional results are in Appendix **Subsec.** A.2. Each colored point here refers to one single method. The point labeled as "GNB_Run" refers to the time consumption of GNB on the arm recommendation process only, and the point "GNB" denotes the overall running time of GNB, including the recommendation and model training process.

Although the linear baselines tend to run faster compared with our proposed GNB, their experiment performances (Subsec. 6.1.1) are not comparable with GNB, as their linear assumption can be too strong for many application scenarios. In particular, for the data set with high context dimension d, the mapping from the arm context to the reward will be much more complicated and more difficult to learn. For instance, as shown by the experiments on the MNIST data set ($d = 784$), the neural algorithms manage to achieve a significant improvement over the linear algorithms (and the other baselines) while enjoying the reasonable running time. Meanwhile, we also have the following remarks: (1) We see that for the two recommendation tasks, GNB takes approximately 0.4 second per round to make the arm recommendation with satisfactory performances for the received user; (2) In all the experiments, we train the GNB framework per 100 rounds after $t > 1000$ and still

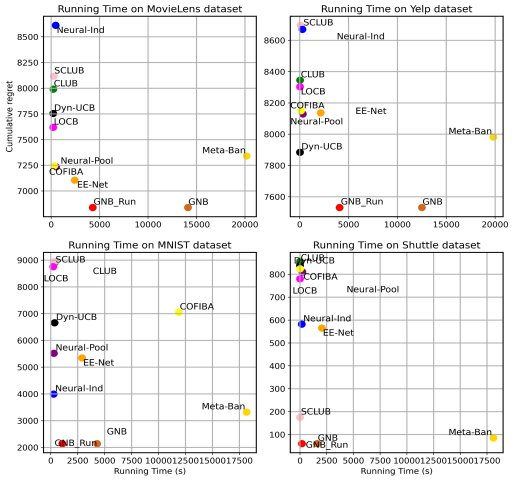

Figure 4: Running time vs. performance with baselines.

manage to achieve the good performance. In this case, the running time of GNB in a long run can be further improved considerably by reducing the training frequency when we already have sufficient user interaction data and a well-trained framework; (3) Moreover, since we are actually predicting the rewards and potential gains for all the nodes within the user graph (or the approximated user graph as in Remark 4.2), GNB is able to serve multiple users in each round simultaneously without running the recommendation procedure for multiple times, which is efficient in real-world cases.

6.6 Supplementary Experiments

Due to page limit, we present supplementary experiments in Appendix Section A, including: (1) [**Subsec.** A.1] experiments showing the potential impact on GNB when there exist underlying user clusters; (2) [**Subsec.** A.2] complementary contents for **Subsec.** 6.5 regarding the "Letter" and "Pendigits" data sets; (3) [**Subsec.** A.3] the convergence results of GNB on recommendation data sets.

7 CONCLUSION

In this paper, we propose a novel framework named GNB to model the fine-grained user collaborative effects. Instead of modeling user correlations through the estimation of rigid user groups, we estimate the user graphs to preserve the pair-wise user correlations for exploitation and exploration respectively, and utilize individual GNN-based models to achieve the adaptive exploration with respect to the arm selection. Under standard assumptions, we also demonstrate the improvement of the regret bound over existing methods from new perspectives of "fine-grained" user collaborative effects and GNNs. Extensive experiments are conducted to show the effectiveness of our proposed framework against strong baselines.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation under Award No. IIS-1947203, IIS-2117902, IIS-2137468, and Agriculture and Food Research Initiative (AFRI) grant no. 2020-67021-32799/project accession no.1024178 from the USDA National Institute of Food and Agriculture. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems* 24 (2011), 2312–2320.

[2] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*. PMLR, 242–252.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.

[4] Yikun Ban and Jingrui He. 2021. Local clustering in contextual multi-armed bandits. In *Proceedings of the Web Conference 2021*. 2335–2346.

[5] Yikun Ban, Jingrui He, and Curtiss B Cook. 2021. Multi-facet contextual bandits: A neural network perspective. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 35–45.

[6] Yikun Ban, Yunzhe Qi, Tianxin Wei, and Jingrui He. 2022. Neural Collaborative Filtering Bandits via Meta Learning. *arXiv preprint arXiv:2201.13395* (2022).

[7] Yikun Ban, Yuchen Yan, Arindam Banerjee, and Jingrui He. 2022. EE-Net: Exploitation-Exploration Neural Networks in Contextual Bandits. In *International Conference on Learning Representations*.

[8] Yuan Cao and Quanquan Gu. 2019. Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Advances in Neural Information Processing Systems* 32 (2019), 10836–10846.

[9] Nicolo Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A gang of bandits. In *NeurIPS*. 737–745.

[10] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247* (2018).

[11] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual bandits with linear payoff functions. In *AISTATS*. 208–214.

[12] Aniket Anand Deshmukh, Urun Dogan, and Clay Scott. 2017. Multi-task learning for contextual bandits. In *NeurIPS*. 4848–4856.

[13] Audrey Durand, Charis Achilleos, Demetris Iacovides, Katerina Strati, Georgios D Mitsis, and Joelle Pineau. 2018. Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis. In *Machine learning for healthcare conference*. PMLR, 67–82.

[14] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International Conference on Learning Representations*.

[15] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. 2017. On context-dependent clustering of bandits. In *ICML*. 1253–1262.

[16] Claudio Gentile, Shuai Li, and Giovanni Zappella. 2014. Online clustering of bandits. In *ICML*. 757–765.

[17] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.

[19] Arthur Jacot, Franck Gabriel, and Clément Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems* 31 (2018).

[20] Parnian Kassraie, Andreas Krause, and Ilija Bogunovic. 2022. Graph Neural Network Bandits. *arXiv preprint arXiv:2207.06456* (2022).

[21] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*. 661–670.

[22] Shuai Li, Wei Chen, Shuai Li, and Kwong-Sak Leung. 2019. Improved Algorithm on Online Clustering of Bandits. In *IJCAI*. 2923–2929.

[23] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative filtering bandits. In *SIGIR*. 539–548.

[24] Trong T Nguyen and Hady W Lauw. 2014. Dynamic clustering of contextual multi-armed bandits. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1959–1962.

[25] Yunzhe Qi, Yikun Ban, and Jingrui He. 2022. Neural Bandit with Arm Group Graph. *arXiv preprint arXiv:2206.03644* (2022).

[26] Filip Radenović, Giorgos Tolias, and Ondřej Chum. 2018. Fine-tuning CNN image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence* 41, 7 (2018), 1655–1668.

[27] Victor Garcia Satorras and Joan Bruna Estrach. 2018. Few-Shot Learning with Graph Neural Networks. In *International Conference on Learning Representations*.

[28] Sohini Upadhyay, Mikhail Yurochkin, Mayank Agarwal, Yasaman Khazaeni, and Djallel Bouneffouf. 2020. Graph Convolutional Network Upper Confident Bound. (2020).

[29] Michal Valko, Nathan Korda, Rémi Munos, Ilias Flaounas, and Nello Cristianini. 2013. Finite-Time Analysis of Kernelised Contextual Bandits. In *Uncertainty in Artificial Intelligence*.

[30] Sofía S Villar, Jack Bowden, and James Wason. 2015. Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges. *Statistical science: a review journal of the Institute of Mathematical Statistics* 30, 2 (2015), 199.

[31] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[32] Max Welling and Thomas N Kipf. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[33] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[34] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. 2016. Contextual bandits in a collaborative environment. In *SIGIR*. 529–538.

[35] Qingyun Wu, Huazheng Wang, Yanen Li, and Hongning Wang. 2019. Dynamic Ensemble of Contextual Bandits to Satisfy Users' Changing Interests. In *WWW*. 2080–2090.

[36] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*. PMLR, 5453–5462.

[37] Keyulu Xu, Mozhi Zhang, Stefanie Jegelka, and Kenji Kawaguchi. 2021. Optimization of graph neural networks: Implicit acceleration by skip connections and more depth. In *International Conference on Machine Learning*. PMLR, 11592–11602.

[38] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.

[39] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *International Conference on Machine Learning*. PMLR, 7134–7143.

[40] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. 2021. Neural Thompson Sampling. In *International Conference on Learning Representations*.

[41] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *NeurIPS*. 321–328.

[42] Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*. PMLR, 11492–11502.

A EXPERIMENTS (CONT.)

Here, for all the UCB-based baselines, we choose their exploration parameter from the range $\{0.01, 0.1, 1\}$ with grid search. We set the $L = 2$ for all the deep learning models including our proposed GNB, and set the network width $m = 100$. The learning rate of all neural algorithms are selected by grid search from the range $\{0.0001, 0.001, 0.01\}$. For EE-Net [7], we follow the default settings in their paper by using a hybrid decision maker, where the estimation is $f_1 + f_2$ for the first 500 time steps, and then we apply an additional neural network for decision making afterwards. For Meta-Ban, we follow the settings in their paper by tuning the clustering parameter γ through the grid search on $\{0.1, 0.2, 0.3, 0.4\}$. For GNB, we choose the k-hop user neighborhood $k \in \{1, 2, 3\}$ with grid search. Reported results are the average of 3 runs. The URLs for the data sets are: MovieLens: https://www.grouplens.org/datasets/movielens/20m/. Yelp: https://www.yelp.com/dataset. MNIST/Shuttle/Letter/Pendigits: https://archive.ics.uci.edu/ml/datasets.

A.1 Experiments with Underlying User Groups

To understand the influence of potential underlying user clusters, we conduct the experiments on the MovieLens and the Yelp data sets, with controlled number of underlying user groups. The underlying user groups are derived by using hierarchical clustering on the user features, with a total of 50 users approximately. Here, we compare GNB with four representative baselines with relatively good performances, including DynUCB [24] [fixed number of user clusters], LOCB [4] [fixed number of user clusters with local clustering], CLUB [16] [distance-based user clustering], Neural-UCB-Pool [42] [neural single-bandit algorithm], and Meta-Ban [6] [neural user clustering bandits]. DynUCB and LOCB are given the **true cluster number** as the prior knowledge to determine the quantity of user clusters or random seeds. Results are shown in Fig. 5.

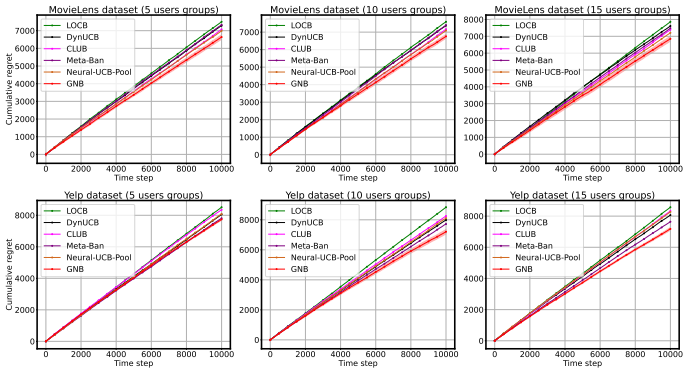

Figure 5: Cumulative regrets for different number of underlying user groups (on MovieLens and Yelp data sets)

As we can see from the results, our proposed GNB outperforms other baselines across different data sets and number of user groups. In particular, with more underlying user groups, the performance improvement of GNB over the baselines will slightly increase, which can be the result of the increasingly complicated user correlations. The modeling of fine-grained user correlations, the adaptive exploration strategy with the user exploration graph, and the representation power of our GNN-based architecture can be the reasons for GNB's good performances.

A.2 Running Time vs. Performance (Cont.)

In Figure 6, we include the comparison with baselines in terms of the running performance and running time on the last two classification data sets ("Letter" and "Pendigits" data sets). Analogously, the results match our conclusions that by applying the GNN models and the "fine-grained" user correlations, GNB can find a good balance between the computational cost and recommendation performance.

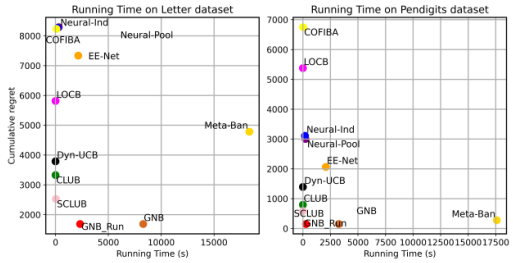

Figure 6: Running time & performance comparison.

A.3 Convergence of GNN Models

Here, for each separate time step interval of 2000 rounds, we show the average cumulative regrets results on MovieLens and Yelp data sets within this interval. We also include the baselines (EE-Net [7], Neural-UCB-Pool [42] and Meta-Ban [6]) for comparison.

	Time step intervals				
Data (Algo.)	0-2k	2k-4k	4k-6k	6k-8k	8k-10k
M (GNB)	0.7050	0.6895	0.6795	0.6770	0.6685
Y (GNB)	0.7705	0.7685	0.7485	0.7450	0.7330
M (EE-Net)	0.7320	0.7120	0.7072	0.6945	0.7070
Y (EE-Net)	0.8480	0.8420	0.7965	0.7950	0.7865
M (Neural-UCB)	0.8090	0.7115	0.7165	0.6945	0.6865
Y (Neural-UCB)	0.8380	0.8115	0.8185	0.7940	0.8025
M (Meta-Ban)	0.7760	0.7245	0.7190	0.7265	0.7140
Y (Meta-Ban)	0.8525	0.8035	0.7930	0.7600	0.7620

Table 4: In different time step intervals, the average regrets per round on MovieLens (M) and Yelp (Y) data sets.

As in Table 4, the average regret per round of GNB is decreasing, along with more time steps. Compared with baselines, GNB manages to achieve the best prediction accuracy across different time step intervals by modeling the fine-grained user correlations and applying the adaptive exploration strategy. Here, one possible reason for the "linear-like" curves of the cumulative regrets is that these two recommendation data sets contain considerable inherent noise, which makes it hard for algorithms to learn the underlying reward mapping function. In this case, achieving experimental improvements on these two data sets is non-trivial.

B INTUITION OF ADAPTIVE EXPLORATION

Recall that GNB adopts a second GNN model ($f_{gnn}^{(2)}$) to adaptively learn the potential gain, which can be either positive or negative. The intuition is that the exploitation model (i.e., $f_{gnn}^{(1)}(\cdot)$) can "provide the excessively high estimation of the reward", and applying the UCB based exploration (e.g., [42]) can amplify the mistake as the UCB is non-negative. For the simplicity of notation, let us denote the expected reward of an arm x as $\mathbb{E}[r] = h(x)$, where h is the unknown reward mapping function. The reward estimation

is denoted as $\hat{r} = f_1(x)$ where $f_1(\cdot)$ is the exploitation model. For GNB, when the estimated reward is lower than the expected reward ($f_1(x) < h(x)$), we will apply the "upward" exploration (i.e., positive exploration score) to increase the chance of arm x being explored. Otherwise, if the estimated reward is higher than the expected reward ($f_1(x) > h(x)$), we will apply the "downward" exploration (i.e., negative exploration score) instead to tackle the excessively high reward estimation. Here, we apply the exploration GNN $f_{gnn}^{(2)}$ to adaptively learn the relationship between the gradients of f_1 and the reward estimation residual $h(x) - f_1(x)$, based on the user exploration graph for a refined exploration strategy. Readers can also refer to [7] for additional insights of neural adaptive exploration.

C PROOF SKETCH OF THEOREM 5.2

For the full proof of the regret bound, please refer to our arXiv version of the paper. Apart from two kinds of estimated user graphs $\{\mathcal{G}_{i,t}^{(1)}\}_{i\in[a]}$, $\{\mathcal{G}_{i,t}^{(2)}\}_{i\in[a]}$ at each time step t, we can also define true user exploitation graph $\{\mathcal{G}_{i,t}^{(1),*}\}_{i\in[a]}$ and true user exploration graph $\{\mathcal{G}_{i,t}^{(2),*}\}_{i\in[a]}$ based on **Def.** 1 and **Def.** 2 respectively. Comparably, the true normalized adjacency matrices of $\mathcal{G}_{i,t}^{(1),*}, i \in [a]$ are represented as $S_{i,t}^{(1),*}$. With r_t, r_t^* separately being the rewards for the selected arm $\boldsymbol{x}_t \in \mathcal{X}_t$ and the optimal arm $\boldsymbol{x}_t^* \in \mathcal{X}_t$, we formulate the pseudo-regret for a single round t, as $R_t = \mathbb{E}[r_t^*|u_t, \mathcal{X}_t] - \mathbb{E}[r_t|u_t, \mathcal{X}_t]$ w.r.t. the candidate arms $\mathcal{X}_t$ and served user u_t. Here, with **Algorithm** 1, we denote $f_{gnn}(\boldsymbol{x}_t) = f_{gnn}^{(1)}(\boldsymbol{x}_t, \mathcal{G}_t^{(1)}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}) + f_{gnn}^{(2)}(\nabla f_t^{(1)}(\boldsymbol{x}_t), \mathcal{G}_t^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1})$, with the arm $\boldsymbol{x}_t$, gradients $\nabla f_t^{(1)}(\boldsymbol{x}_t) = \frac{\nabla_{\boldsymbol{\Theta}_{gnn}^{(1)}} f_{gnn}^{(1)}(\boldsymbol{x}_t, \mathcal{G}_t^{(1)}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1})}{c_g L}$ ($c_g > 0$ is the normalization factor, such that $\|\nabla f_t^{(1)}(\boldsymbol{x}_t)\|_2 \leq 1$), as well as the estimated user graphs $\mathcal{G}_t^{(1)}, \mathcal{G}_t^{(2)}$ related to chosen arm $\boldsymbol{x}_t$. On the other hand, with the true graph $\mathcal{G}_t^{(1),*}$ of arm $\boldsymbol{x}_t$, the corresponding gradients will be $\nabla f_t^{(1),*}(\boldsymbol{x})$. Analogously, we also have the estimated user graphs $\mathcal{G}_{t,*}^{(1)}, \mathcal{G}_{t,*}^{(2)}$ for the optimal arm $\boldsymbol{x}_t^*$. Afterwards, in round $t \in [T]$, the single-round regret R_t will be

$$
\begin{aligned}
R_t &= \mathbb{E}[r_t^*|u_t, \mathcal{X}_t] - \mathbb{E}[r_t|u_t, \mathcal{X}_t] \\
&= \mathbb{E}[r_t^*|u_t, \mathcal{X}_t] - f_{gnn}(\boldsymbol{x}_t) + f_{gnn}(\boldsymbol{x}_t) - \mathbb{E}[r_t|u_t, \mathcal{X}_t] \\
&\underset{\text{(i)}}{\leq} \mathbb{E}[r_t^*|u_t, \mathcal{X}_t] - f_{gnn}(\boldsymbol{x}_t^*) + f_{gnn}(\boldsymbol{x}_t) - \mathbb{E}[r_t|u_t, \mathcal{X}_t] \\
&\leq \mathbb{E}\big[|r_t^* - f_{gnn}(\boldsymbol{x}_t^*)|\big|u_t, \mathcal{X}_t\big] + \mathbb{E}\big[|r_t - f_{gnn}(\boldsymbol{x}_t)|\big|u_t, \mathcal{X}_t\big] \\
&= \mathbb{E}\Big[|f_{gnn}^{(2)}(\nabla f_t^{(1)}(\boldsymbol{x}_t^*),\ \mathcal{G}_{t,*}^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}) \\
&\qquad - (r_t^* - f_{gnn}^{(1)}(\boldsymbol{x}_t^*,\ \mathcal{G}_{t,*}^{(1)}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}))|\Big|u_t, \mathcal{X}_t\Big] + \\
&\quad \mathbb{E}\Big[|f_{gnn}^{(2)}(\nabla f_t^{(1)}(\boldsymbol{x}_t),\ \mathcal{G}_t^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}) \\
&\qquad - (r_t - f_{gnn}^{(1)}(\boldsymbol{x}_t,\ \mathcal{G}_t^{(1)}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}))|\Big|u_t, \mathcal{X}_t\Big] \\
&= \mathrm{CB}_t(\boldsymbol{x}_t) + \mathrm{CB}_t(\boldsymbol{x}_t^*)
\end{aligned}
$$

where inequality (i) is due to the arm pulling mechanism, i.e., $f_{gnn}(\boldsymbol{x}_t) \geq f_{gnn}(\boldsymbol{x}_t^*)$, and $\mathrm{CB}_t(\cdot)$ is the regret bound function in round t, formulated by the last equation. Then, given arm $\boldsymbol{x} \in \mathcal{X}_t$ and its reward r, with the aforementioned notation, we have

$$
\begin{aligned}
\mathrm{CB}_t(\boldsymbol{x}) = \mathbb{E}\Big[&|f_{gnn}^{(2)}(\nabla f_t^{(1)}(\boldsymbol{x}),\ \mathcal{G}^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}) \\
&- (r - f_{gnn}^{(1)}(\boldsymbol{x},\ \mathcal{G}^{(1)}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}))|\Big|u_t, \mathcal{X}_t\Big] \\
\leq\ & \underbrace{\mathbb{E}\Big[|f_{gnn}^{(2)}(\nabla f_t^{(1),*}(\boldsymbol{x}), \mathcal{G}^{(2),*}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}) - (r - f_{gnn}^{(1)}(\boldsymbol{x},\ \mathcal{G}^{(1),*}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}))|\Big|u_t, \mathcal{X}_t\Big]}_{I_1} \\
&+ \underbrace{\mathbb{E}\Big[|f_{gnn}^{(1)}(\boldsymbol{x},\ \mathcal{G}^{(1),*}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}) - f_{gnn}^{(1)}(\boldsymbol{x},\ \mathcal{G}^{(1)}; [\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1})|\Big|u_t, \mathcal{X}_t\Big]}_{I_2} \\
&+ \underbrace{\mathbb{E}\Big[|f_{gnn}^{(2)}(\nabla f_t^{(1),*}(\boldsymbol{x}), \mathcal{G}^{(2),*}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}) - f_{gnn}^{(2)}(\nabla f_t^{(1),*}(\boldsymbol{x}),\ \mathcal{G}^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1})|\Big|u_t, \mathcal{X}_t\Big]}_{I_3} \\
&+ \underbrace{\mathbb{E}\Big[|f_{gnn}^{(2)}(\nabla f_t^{(1),*}(\boldsymbol{x}), \mathcal{G}^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}) - f_{gnn}^{(2)}(\nabla f_t^{(1)}(\boldsymbol{x}),\ \mathcal{G}^{(2)}; [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1})|\Big|u_t, \mathcal{X}_t\Big]}_{I_4}.
\end{aligned}
$$

Here, we have the term I_1 representing the estimation error induced by the GNN model parameters $\{[\boldsymbol{\Theta}_{gnn}^{(1)}]_{t-1}, [\boldsymbol{\Theta}_{gnn}^{(2)}]_{t-1}\}$, the term I_2 denoting the error caused by the estimation of user exploitation graph. Then, error term I_3 is caused by the estimation of user exploration graph, and term I_4 is the output difference given input gradients $\nabla f_t^{(1),*}(\boldsymbol{x})$ and $\nabla f_t^{(1)}(\boldsymbol{x})$, which are individually associated with the true user exploitation graph $\mathcal{G}^{(1),*}$ and the estimation $\mathcal{G}^{(1)}$. These four terms I_1, I_2, I_3, I_4 can be bounded respectively. Afterwards, with the notation from **Theorem** 5.2, we have the pseudo regret after T rounds, i.e., $R(T)$, as

$$
\begin{aligned}
R(T) = \sum_{t\in[T]} R_t \leq\ & 2\cdot\sqrt{T}\Big(\sqrt{2\xi_2} + \frac{3L}{\sqrt{2}} + (1+\gamma_2)\sqrt{2\log(\frac{Tn\cdot a}{\delta})}\Big) + \\
& \sqrt{T}\cdot O(\xi_L)\cdot\Big(\sqrt{2\xi_1} + \frac{3L}{\sqrt{2}} + (1+\gamma_1)\sqrt{2\log(\frac{Tn\cdot a}{\delta})}\Big) + O(1)
\end{aligned}
$$

where the inequality is because we have sufficient large network width $m \geq \Omega(\mathrm{Poly}(T, L, a, 1/\rho)\cdot\log(1/\delta))$ as indicated in **Theorem** 5.2. Meanwhile, with sufficient $m \geq \Omega(\mathrm{Poly}(T, \rho^{-1}))$, the terms γ_1, γ_2 can also be upper bounded by $O(1)$, which leads to

$$
\begin{aligned}
R(T) &\leq \sqrt{T}\cdot\Big(O(\sqrt{\xi_2} + \xi_L\sqrt{\xi_1}) + O(L\xi_L) + O(\xi_L)\cdot\sqrt{2\log(\frac{Tn\cdot a}{\delta})}\Big) + \\
&\qquad \sqrt{T}\cdot O(L) + O(\xi_L) + O(1) \\
&\leq \sqrt{T}\cdot\Big(O(L\xi_L) + O(\xi_L)\cdot\sqrt{2\log(\frac{Tn\cdot a}{\delta})}\Big) + \sqrt{T}O(L) + O(\xi_L) + O(1)
\end{aligned}
$$

since we have $\xi_1, \xi_2 \leq O(\frac{1}{T})$. This will complete the proof sketch.