Report on Graph Neural Network Bandit

Xu Bowen
xubw1@shanghaitech.edu.cn

Zhang Tianyi
zhangty12022@shanghaitech.edu.cn

Zhong Wenfu
zhongwf2022@shanghaitech.edu.cn

Abstract

The graph neural network bandit first applies the graph neural networks to approximate the reward. With the help of graph neural networks, the input domain extends from Euclidean space to the Reproducing Kernel Hilbert Space. After reproducing the code of the graph neural network bandit, this report extends its setting. Considering the dependency among arms, we use a social graph to model it and derive UCB-type algorithms. Besides, we try to introduce the Constraint setting in the GNN Bandits scenario to ensure that the accumulative costs of interacting with the environment are within a certain budget. By using Graph Neural Network to estimate the reward and cost, and applying Primal-Dual algorithm to control their balance, we propose a new algorithm: GNN-PD. This Constrained GNN Bandits algorithm can be widely applied in scenarios with large consumption, such as experiments on chemicals and drugs.

1 Introduction

Contextual bandits are a specific type of multi-armed bandit problem where the additional contextual information (contexts) related to arms are available at each round, and the learner intends to refine its selection strategy based on the received arm contexts and rewards. Various contextual bandit algorithms have been applied in real-world recommendation tasks, such as online content recommendation and advertising (Li et al. [2010], Wu et al. [2016]), and clinical trials (Durand et al. [2018], Villar et al. [2015]). The main idea is to exploit the correlation between the rewards and the contexts. On the one hand, linear models or kernelized models are used to model the relationship between the reward and context(Srinivas et al. [2012], Chowdhury and Gopalan [2017]), and on the other hand, they use neural networks to approximate the reward function(Zhou et al. [2020]).

Learning graph structure data, such as molecular or biological map representations, requires the design of sequential methods that effectively utilize graph structures. There are prediction tasks in graph domain such as designing novel materials (Guo and Buehler [2020]), drug discovery (Jiang et al. [2021]), discovering social relationship (Wu et al. [2020]). Graph neural networks (GNNs) are deep learning based methods that operate on graph domain. Due to its convincing performance, GNN has become a widely applied graph analysis method recently. With the power of GNN to estimate a graph reward, a bandit optimizing algorithm has been proposed by Kassraie et al. [2022]. This algorithm employs GNN with one convolutional layer to estimate the unknown reward and achieves sublinear cumulative regret bounds.

In the report, it extends the setting of the graph neural network bandit (Kassraie et al. [2022]), combines the graph feedback with it, and applies constraints on the bandit.

Related Work. Our work is based on the neural bandit, in which a single hidden layer convolutional network (Zhou et al. [2020]) is used to estimate the reward function. It presents key structural assumptions about graph reward functions and establishes a new connection between additive permutation-invariant kernels and GNTKs. It constructs an effective confidence set for graph neural networks and uses it to construct a GNN bandit algorithm that enables sublinear regret.

The bandit problem with graph feedback, proposed in Caron et al. [2012], is modeled by a directed graph, where nodes are the bandit's arms, and once an arm is triggered, all its incident arms are observed. A naturally challenging question is how the graph structure affects the min-max regret. This problem has been widely studied (Mannor and Shamir [2011], Alon et al. [2013], Bian et al. [2022]).

To solve the Constrained Bandits problem, the paper Shangshang et al. [2023] uses multi-layer perceptrons to estimate the reward and cost functions. Also, apply primal-dual algorithm (Neural-PD) to choose the optimal arms. The Neural-PD algorithm achieves an extremely excellent performance of sublinear regret and zero constraint violation. However, this algorithm cannot be used to solve the Graph Structured problem.

2 Formulation

2.1 GNTK

To clarify the use of GNTK, we firstly propose the graph classification problem. We denote the graph as $G(V, E)$, where V is the set of vertices and E is the edges set. Denote $|V|$ as the number of vertices and $|E|$ as the number of edges. The graph classification problem can be stated as: given a specific graph, classify it into its corresponding class. For example, we have two classes of molecular: high energy and low energy.

To achieve the goal of classification, we need to extract the feature of a graph, which facilitates us to use a special kinds of neural network: GNN(Graph Neural Network). Therefore we introduce the definition of GNN and its relationship with the traditional NNs.

We have our architecture of GNN as below,

$$\begin{aligned} f^{(1)}\left(\bar{\mathbf{h}}_{G,j}\right) &= \mathbf{W}^{(1)}\bar{\mathbf{h}}_{G,j} \\ f^{(l)}\left(\bar{\mathbf{h}}_{G,j}\right) &= \sqrt{\frac{2}{m}}\mathbf{W}^{(l)}\sigma_{relu}\left(f^{(l-1)}\left(\bar{\mathbf{h}}_{G,j}\right)\right) \in \mathbb{R}^m, 1 < l \le L \\ f_{GNN}\left(G;\theta\right) &= \frac{1}{N}\sum_{j=1}^{N}\sqrt{2}\mathbf{W}^{(L+1)}\sigma_{relu}\left(f^{(L)}\left(\bar{\mathbf{h}}_{G,j}\right)\right) \end{aligned} \tag{1}$$

where $\bar{\mathbf{h}}_{G,j}$ is the feature of node j in the graph G, which takes the graph as input. This is actually an instance of the graph convolutional network(GCN) (Kipf and Welling [2016]).

Similarly we could have the NN as,

$$\begin{aligned} f^{(1)}\left(\mathbf{x}\right) &= \mathbf{W}^{(1)}\mathbf{x} \\ f^{(l)}\left(\mathbf{x}\right) &= \sqrt{\frac{2}{m}}\mathbf{W}^{(l)}\sigma_{relu}\left(f^{(l-1)}\left(\mathbf{x}\right)\right) \in \mathbb{R}^m, 1 < l \le L \\ f_{NN}\left(\mathbf{x};\theta\right) &= \sqrt{2}\mathbf{W}^{(L+1)}\sigma_{relu}\left(f^{(L)}\left(\mathbf{x}\right)\right) \in \mathbb{R} \end{aligned} \tag{2}$$

It is easy to find the common structure between them. Here we give out some basic observations and conclusions, and then introduce the permutation invariant kernel of the Reproducing Kernel Hilbert Space (RKHS), with NTK (Jacot et al. [2018]) and GNTK.

Theorem 1 *There exists equivalence between GNN and NN as,*

$$f_{GNN}\left(G;\theta\right) = \frac{1}{N}\sum_{i=1}^{N} f_{NN}\left(\bar{\mathbf{h}}_{G,i};\theta\right)$$

Theorem 1 is obvious and trivial and could be proven in a simple way. Then we introduce the property of permutation invariant, which is a crucial character of the graph data. Consider a permutation operator c, which will change the ordering of nodes (e.g. node number of: 1, 2, 3, 4 will be converted into: 3, 4, 1, 2). According to this, we could have the permutation invariant of the GNN.

Theorem 2 *For any permutation operator c we have,*

$$f_{GNN}(G;\theta) = f_{GNN}(c \cdot G;\theta)$$

Theorem 2 is apparent because we added up all the nodes in the GNN architecture formulation(1), and no matter how the nodes are ordered there's no change in the result.

Furthermore, we want the corresponding kernel also to be permutation invariant, thus we could define the kernel of RKHS for two graphs as,

$$\bar{k}(G,G') = \frac{1}{|P_N|^2} \sum_{c,c' \in P_N} \frac{1}{N} \sum_{j=1}^{N} k_{NN}\left(\bar{\mathbf{h}}_{G,c(j)}, \bar{\mathbf{h}}_{G',c'(j)}\right)$$

where the k_{NN} is the corresponding NTK of the neural network.

We here introduce the Neural Tangent Kernel(NTK) theory. Now consider the first order Taylor approximation of $f_{NN}(\mathbf{x};\theta)$ at the initial point θ_0 as,

$$\tilde{f}_{NN}(\mathbf{x};\theta) = \mathbf{g}_{NN}(\mathbf{x};\theta_0)^T(\theta - \theta_0).$$

Without loss of generality, we here assume $f_{NN}(\mathbf{x};\theta_0) = 0$. It is worth noting that its corresponding kernel is $\tilde{k}(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{g}_{NN}(\mathbf{x}_i;\theta_0)^T \mathbf{g}_{NN}(\mathbf{x}_j;\theta_0)$. According to (Jacot et al. [2018]), with some specific initialization of parameters we could have,

$$\lim_{m\to\infty} \frac{\tilde{k}(\mathbf{x}_i,\mathbf{x}_j)}{m} = k_{NN}(\mathbf{x}_i,\mathbf{x}_j)$$

which $k_{NN}(\mathbf{x}_i,\mathbf{x}_j)$ is what we called Neural Tangent Kernel.

Similarly, we could find the kernel $\tilde{k}(G_i,G_j)$ on the first order approximation of GNN and define the corresponding Graph Neural Tangent Kernel(GNTK) as,

$$k_{GNN}(G_i,G_j) = \lim_{m\to\infty} \frac{\tilde{k}(G_i,G_j)}{m}$$

Theorem 3 *There exists a relationship between NTK and GNTK as,*

$$k_{GNN}(G,G') = \frac{1}{N^2} \sum_{j,j'}^{N} k_{NN}\left(\bar{\mathbf{h}}_{G,j}, \bar{\mathbf{h}}_{G,j'}\right)$$

Theorem 3 directly follows from theorem 1.

2.2 GNN Bandits

Traditional bandit algorithm mainly concentrates on the estimation of reward expectation, however, it does not utilize the contextual information well. LinUCB considers maintaining a linear model to predict the reward, however, the ability of linear function is limited. Neural UCB (Zhou et al. [2020]) replaces the linear model with a neural network that is more expressive but not permutation invariant. To utilize the structure information of the graph, we can design a scheme that is embedded with GNN. Considering the completeness, we first introduce two components in kernelized bandit: information gain and confidence sets.

2.2.1 Information Gain

The learner in the bandit problem looks for the optimal decision while observing new rewards, and the maximum information gain indicates the speed of approaching the optimal reward function f^*. Formally, we have the information gain defined as,

$$I(G_1,\ldots,G_T;k_{GNN}) = \frac{1}{2}logdet(\mathbf{I} + \frac{\mathbf{K}_{GNN,T}}{\lambda})$$

where $\mathbf{K}_{GNN,T} = [k_{GNN}(G_i,G_j)]_{i,j\leq T}$ being the kernel matrix. Moreover, we assume here that during the process, we observed contextual graphs: $G_1,\ldots,G_2$ and the corresponding rewards sequence with sub-Gaussian noise centered at zero with the proxy variance being λ.

Theorem 4 *Suppose the observation noise is i.i.d., and drawn from a zero-mean sub-Gaussian distribution, and the input domain is $\mathcal{G}$. Then the maximum information gain associated with k_{GNN} is bounded by,*

$$\gamma_{GNN,T} = max_{\{G_1,\ldots,G_T\}} I\left(G_1,\ldots,G_T; k_{GNN}\right) = \mathcal{O}\left(T^{\frac{d-1}{d}}\left(logT\right)^{\frac{1}{d}}\right)$$

$\gamma_{GNN,T}$ can be expressed in the regret bound.

2.2.2 Confidence Sets

Another useful tool for the analysis of regret bound should be the confidence sets, which gives us the extent of optimal reward function with high probability. Formally, we have,

$$\mathbb{P}\left(\forall G \in \mathcal{G} : f^*\left(G\right) \in \mathcal{C}_{t-1}\left(G,\delta\right)\right) \geq 1-\delta$$

Moreover we express the confidence sets as $\mathcal{C}_{t-1}\left(G,\delta\right) = \left[\hat{\mu}_{t-1} \pm \beta_t \hat{\theta}_{t-1}\right]$, and β_t depends on the confidence level δ. We can estimate these parameters as,

$$\hat{\mu}_{t-1}\left(G\right) := f_{GNN}\left(G; \theta_{t-1}^{(J)}\right)$$

$$\hat{\sigma}_{t-1}\left(G\right) := \frac{\mathbf{g}_{GNN}\left(G;\theta^0\right)^T}{\sqrt{m}}\left(\lambda\mathbf{I} + \frac{1}{t}\sum_{i=1}^{t-1}\frac{\mathbf{g}_{GNN}\left(G_i;\theta^0\right)\mathbf{g}_{GNN}\left(G_i;\theta^0\right)^T}{m}\right)^{-1}\frac{\mathbf{g}_{GNN}\left(G;\theta^0\right)}{\sqrt{m}}$$

We denote λ_0 is the minimum eigenvalue of the kernel matrix calculated for the entire domain, i.e. $\mathbf{K}_{GNN} = \left[k_{GNN}\left(G,G'\right)\right]_{G,G'\in\mathcal{G}}$. And we could note that we replace the gradient of GNN with $\mathbf{g}_{GNN}\left(G;\theta^0\right)$, which comes from the assumption of NTK.

Theorem 5 *Set $\delta \in (0,1)$. Suppose $f^* \in \mathcal{H}_{k_{GNN}}$ with a bounded norm $||f^*||_{k_{GNN}} \leq B$. Assume that random sequences $\left(C_i\right)_{i<t}$ and $\left(\epsilon_i\right)_{i<t}$ are statistically independent. Let the width $m = poly\left(t, L, B, |\mathcal{G}|, \lambda, \lambda_0^{-1}, \log\left(N/\delta\right)\right)$, learning rate $\eta = C\left(Lm + m\lambda\right)^{-1}$ with some universal constant C, and $J \geq 1$. Then for all $G \in \mathcal{G}$, with probability of at least $1-\delta$,*

$$|f^*\left(G\right) - \hat{\mu}_{t-1}\left(G\right)| \lesssim \beta_t \hat{\sigma}_{t-1}\left(G\right)$$

where $\beta_t \approx \sqrt{2}B + \frac{\sigma}{\sqrt{\lambda}}\sqrt{2\log 2|\mathcal{G}|/\delta}$.

Theorem 4 and Theorem 5 are two useful tools for our analysis of regret bound.

3 Algorithms

In this section, we will introduce two main results from (Kassraie et al. [2022]), and discuss the regret bound and effect.

3.1 Comparison

Algorithm 1 and Algorithm 2 respectively show a normal variant of Neural UCB and a phased elimination implementation. The GNN-PE(i.e. the latter) considers finding a reasonable candidate set and explore on it.

Based on the setting in the paper, we further conducted a comparison of them as shown in Figure 1.

3.2 Convergence Analysis

We now get deeper into the convergence analysis of the GNN Phased Elimination algorithm. Recall our useful tools for analysis: 1)Information Gain; 2)Confidence Sets. Firstly, we show that there exists a correlation between Information Gain and the estimated variance.

Algorithm 1 GNN-UCB

```
Require: $m, J, \eta, \lambda, \beta_t, T$
  Initialize network parameters to a random $\theta^0$, and $\hat{\mathbf{K}}_0 = \sigma^2 \mathbf{I}$
  for $t = 1 \dots T$ do
      for $G \in \mathcal{G}$ do
          $\hat{\sigma}_{t-1}^2 \leftarrow \mathbf{g}_{GNN}^T \left(G; \theta^0\right) \hat{\mathbf{K}}_{t-1}^{-1} \mathbf{g}_{GNN} \left(G; \theta^0\right) / m$
          $U_{G,t} \leftarrow f_{GNN} \left(G; \theta_{t-1}^{(J)}\right) + \beta_t \hat{\sigma}_{t-1} \left(G\right)$
      end for
      $G_t = argmax_{G \in \mathcal{G}} U_{G,t}$
      Select $G_t$ and append the rewards vector $\mathbf{y}_t$ by the observed reward.
      Set $\hat{\mathbf{K}}_t \leftarrow \lambda \mathbf{I} + \sum_{i \leq t} \mathbf{g}_{GNN} \left(G_i; \theta^0\right) \mathbf{g}_{GNN} \left(G_i; \theta^0\right) / mt$
      Calculate $\theta_t^{(J)}$ = TrainGNN$\Big(m, J, \eta, \lambda, \theta^0, \left(G_i, y_i\right)_{i \leq t}\Big)$
  end for
```

Algorithm 2 GNN PHASED ELIMINATION

```
Require: $m, J, \eta, \lambda, T$
  Set episode index $e = 1$, episode length $T_e = 1$, and set of potentially optimal graphs $\mathcal{G}_e = \mathcal{G}$
  Initialize network parameters to a random $\theta^0$.
  for $t = 1 \dots T_e$ do
      for $G \in \mathcal{G}$ do
          $\hat{\sigma}_{t-1}^2 \left(G\right) \leftarrow \mathbf{g}_{GNN}^T \left(G; \theta^0\right) \hat{\mathbf{K}}_{t-1}^{-1} \mathbf{g}_{GNN} \left(G; \theta^0\right) / m$
      end for
      Select $G_t = argmax_{G \in \mathcal{G}_e} \hat{\sigma}_{t-1}^2 \left(G\right)$
  end for
  Calculate $\theta_e^{(J)}$ =TrainGNN$\Big(m, J, \eta, \lambda, \theta^0, \left(G_i, y_i\right)_{t=1}^{T_e}\Big)$
  $\mathcal{G}_{e+1} =\leftarrow \Big\{G \in \mathcal{G}_e : f_{GNN} \left(G; \theta_e^{(J)}\right) + \beta \hat{\sigma}_{T_e} \left(G\right) + 2\epsilon \geq \max_{G \in \mathcal{G}_e} \left(f_{GNN} \left(G; \theta_e^{(J)}\right) - \beta \hat{\sigma}_{T_e} \left(G\right)\right)\Big\}$
  Set $T_{e+1} \leftarrow 2T_e,\ e \leftarrow e + 1$ and return to the initialization step.
```

Theorem 6 *The Information Gain is considered within the upper bound of the estimated variance,*

$$\hat{\sigma}_{T_e}^2 \left(G\right) \leq \frac{2\hat{\gamma}_{T_e}}{T_e log \left(1 + \lambda^{-1}\right)}$$

proof. Firstly, the estimated variance decreases during the process, and hence we have,

$$\hat{\sigma}_{T_e}^2 \left(G\right) \leq \frac{1}{T_e} \sum_{t=1}^{T_e} \hat{\sigma}_{t-1}^2 \left(G_t\right)$$

For any $s \in [0, 1/\lambda]$, we have $s^2 \leq \frac{1}{\lambda \log(1+1/\lambda)} \log\left(1 + s^2\right)$. Then we have,

$$\hat{\sigma}_{t-1}^2 \left(G_t\right) \leq \frac{1}{\log\left(1 + 1/\lambda\right)} \log\left(1 + \lambda^{-2} \hat{\sigma}_{t-1}^2 \left(G_t\right)\right)$$

Also we have,

$$\sum_{t=1}^{T_e} \log\left(1 + \lambda^{-2} \hat{\sigma}_{t-1}^2 \left(G_t\right)\right) = 2I\left(G_1, \dots, G_t; \hat{k}_{GNN}\right) \leq 2\hat{\gamma}_{T_e}$$

Combining them we could get the proof.

Next, we introduce two bound theorem to construct the good events.

Theorem 7 *Fixed an episode e, let $\tilde{\delta} = \delta/(3E)$, then for any $G \in \mathcal{G}$ with probability at least $1 - \tilde{\delta}$ we have,*

$$|f_{GNN}\left(G; \theta_e^{(J)}\right) - f^*\left(G\right)| \leq \beta \hat{\sigma}_{T_e} \left(G\right) + \epsilon$$

Theorem 8 *For $\delta \in (0,1)$ and $m = poly\left(t, L, |\mathcal{G}|, \lambda, B, \lambda_0^{-1}, \log\left(N/\delta\right)\right)$, with probability at least $1-\delta$ we have,*

$$\hat{\gamma}_T \leq \gamma_T + \epsilon\left(m\right)$$

where γ_T is the maximum information gain of GNTK over $\mathcal{G}$ and $\epsilon\left(m\right) = o\left(m^{-1/4}\right)$.

With the help of Theorem 7 and Theorem 8, we can construct the good event. Specifically, we define events:

$$A_i = \left\{|f_{GNN}\left(G; \theta_i^{(J)}\right) - f^*\left(G\right)| \leq \beta\hat{\sigma}_{T_i}\left(G\right) + \epsilon\right\}$$
$$B = \left\{\hat{\gamma}_T \leq \gamma_T + \epsilon\left(m\right)|\tilde{\delta} = \frac{\delta}{3}\right\}$$

Then we define the good event as: $C = \left(\bigcap_{i=1}^{E} A_i\right) \cap B$. According to the union bound we have,

$$\mathbb{P}\left(\bigcap_{i=1}^{E} A_i\right) = 1 - \mathbb{P}\left(\bigcup_{i=1}^{E} \bar{A}_i\right) \geq 1 - \frac{\delta}{3}$$
$$\mathbb{P}\left(C\right) = 1 - \mathbb{P}\left(\bar{C}\right) \geq 1 - \frac{2\delta}{3}$$

In the next paragraphs, we will condition on the good event.

Denote G^* as the best choice, $G_t^{(e)}$ as the t-th choice in the e-th epoch and f^* as the actual reward function.

Theorem 9 *Under the good event C defined before, for any e and t we have,*

$$f^*\left(G^*\right) - f^*\left(G_t^{(e)}\right) \leq 4\beta \max_{G \in \mathcal{G}_{e-1}} \left(G\right) + 4\epsilon$$

proof. Firstly, we should announce that the optimal solution will not be eliminated during the elimination process.

$$f^*\left(G\right) \leq f^*\left(G^*\right), \forall G \in \mathcal{G}$$
$$f^*\left(G\right) \geq f_{GNN}\left(G; \theta_e^{(J)}\right) - \beta\hat{\sigma}_{T_e}\left(G\right) - \epsilon$$
$$f^*\left(G^*\right) \leq f_{GNN}\left(G^*; \theta_e^{(J)}\right) + \beta\hat{\sigma}_{T_e}\left(G^*\right) + \epsilon$$

The above inequalities come from Theorem 7, and combine them we can finish the claim. Then we come back to our objective,

$$\begin{aligned}
f^*\left(G^*\right) - f^*\left(G_t^{(e)}\right) \leq & f_{\text{GNN}}\left(G^*; \boldsymbol{\theta}_{e-1}^{(J)}\right) + \beta\hat{\sigma}_{T_{e-1}}\left(G^*\right) + 2\epsilon \\
& - \left(f_{\text{GNN}}\left(G_t^{(e)}; \boldsymbol{\theta}_{e-1}^{(J)}\right) - \beta\hat{\sigma}_{T_{e-1}}\left(G_t^{(e)}\right)\right) \\
& = \left(f_{\text{GNN}}\left(G^*; \theta_{e-1}^{(J)}\right) - \beta\hat{\sigma}_{T_{e-1}}\left(G^*\right) - \epsilon\right) \\
& + 2\beta\hat{\sigma}_{T_{e-1}}\left(G^*\right) + 4\epsilon \\
& - \left(f_{\text{GNN}}\left(G_t^{(e)}; \theta_{e-1}^{(J)}\right) + \beta\hat{\sigma}_{T_{e-1}}\left(G_t^{(e)}\right) + \epsilon\right) \\
& + 2\beta\hat{\sigma}_{T_{e-1}}\left(G_t^{(e)}\right) \\
& \leq 4\beta \max_{G \in \mathcal{G}_{e-1}} \hat{\sigma}_{T_{e-1}}(G) + 4\epsilon
\end{aligned}$$

The last inequality comes from,

$$f_{\text{GNN}}\left(G_t^{(e)}; \theta_{e-1}^{(J)}\right) + \beta\hat{\sigma}_{T_{e-1}}\left(G_t^{(e)}\right) + \epsilon \geq f_{\text{GNN}}\left(G^*; \theta_{e-1}^{(J)}\right) - \beta\hat{\sigma}_{T_{e-1}}\left(G^*\right) - \epsilon$$

which can be referred from the elimination step.

Theorem 10 *The regret of GNN-PE is,*

$$R_T = \mathcal{O}\left(\sqrt{T\gamma_T}\left(B + \frac{\sigma}{\sqrt{\lambda}}\sqrt{\log\frac{|\mathcal{G}|\log T}{\delta}}\right)\right)$$

proof. By referencing to Theorem 8 and Theorem 9, we could have,

$$\begin{aligned}
R_T &\leq m_1 B + \sum_{e=2}^{E}\sum_{t=1}^{T_e} 4\beta \max_{G\in\mathcal{G}_{e-1}} \hat{\sigma}_{T_{e-1}}(G) + 4\epsilon \\
&\leq m_1 B + \sum_{e=2}^{E} 4T_e\beta\sqrt{\frac{2\hat{\gamma}_{T_{e-1}}}{T_{e-1}\log\left(1+\lambda^{-1}\right)}} + 4T_e\epsilon \\
&= m_1 B + \sum_{e=2}^{E} 8\beta\sqrt{\frac{2T_{e-1}\hat{\gamma}_{T_{e-1}}}{\log\left(1+\lambda^{-1}\right)}} + 4T_e\epsilon \\
&\leq m_1 B + \sum_{e=2}^{E} 8\beta\sqrt{\frac{2T\hat{\gamma}_T}{\log\left(1+\lambda^{-1}\right)}} + 4T_e\epsilon \\
&\leq m_1 B + 8(\log(T)+1)\left(\beta\sqrt{\frac{2T\hat{\gamma}_T}{\log\left(1+\lambda^{-1}\right)}} + 4T\epsilon\right) \\
&\leq m_1 B + 8(\log(T)+1)\left(\beta\sqrt{\frac{2T\left(\gamma_T+\epsilon(m)\right)}{\log\left(1+\lambda^{-1}\right)}} + 4T\epsilon\right)
\end{aligned}$$

4 Graph bandit with graph feedback

In the traditional bandit problem, we do not consider the relationship between arms, however, in many practical scenarios, there exists dependence between arms. In this section, we introduce the arms social network and try to utilize such connectivity information.

Consider there exists a network $\mathcal{N}$ between arms with the adjacent matrix $\mathcal{A}$, which indicates the share of information between arms. For example, pulling an arm will give feedback not only consisting of its own reward but also of its neighbor's. How to use the extra information in the exploration and exploitation behavior is really worth studying. Inspired by the work of (Bian et al. [2022]), we proposed two variants based on GNN: GGRIND and EGGRIND, which have shown an efficient utilization of the correlation.

4.1 Graph Feedback

Formally, a K-armed bandit problem is defined by K distributions $P_1, P_2, \ldots, P_K$ for each arm of the bandit with respective means $\mu_1, \mu_2, \ldots, \mu_K$. All rewards $\{X_{i,t}, i \in [1,K], t \geq 1\}$ are assumed to be independent. Then the mean estimate after m observations is $\bar{X}_{i,m} := \frac{1}{m}\sum_{s=1}^{m} X_{i,s}$. In the standard bandit problem, the only information available at time t is the sequence $\left(X_{I_s,s}\right)_{s\leq t}$.

Now we consider the setting of the bandit with graph feedback. We use a social graph to denote the dependent relationship between arms. Given an undirected graph $G_f = (V, E)$, V is the nodes set that represents the arms and any edge $e \in E$ represents the relationship between two nodes. When we pull the arm i, we not only get $X_{i,t}$, but also receive the rewards from all connected nodes in graph G_f. Let $\mathrm{N}(i)$ denote the observation set of arm i consisting of i and its neighbors in G_f. Thus we have the number of observations as

$$O_i(n) := \sum_{t=1}^{n} \mathbf{1}\left\{I_t \in N(i)\right\}$$

With the number of observations, we design the UCB-type algorithms.

4.2 Algorithm

The degree of each node (arm) in the feedback graph G_f is important for the design of the exploration item. Specifically, the greater the degree of the node, the more likely it is to be observed. At the same time, more observation of the nodes will lead to more accurate estimates of their average results. Therefore, there is no need for more exploration on larger nodes. We introduce a new policy that

Algorithm 3 GGRIND

```
Require: $m, J, \eta, \lambda, \beta_t, T, G_f$
  Initialize network parameters to a random $\theta^0$, and $\hat{\mathbf{K}}_0 = \sigma^2\mathbf{I}$
  for $t = 1 \dots T$ do
      for $G \in \mathcal{G}$ do
          $\hat{\sigma}^2_{t-1} \leftarrow \mathbf{g}^T_{GNN}\left(G;\theta^0\right)\hat{\mathbf{K}}^{-1}_{t-1}\mathbf{g}_{GNN}\left(G;\theta^0\right)/m * \frac{1}{deg(G)+1}$
          $U_{G,t} \leftarrow f_{GNN}\left(G;\theta^{(J)}_{t-1}\right) + \beta_t\hat{\sigma}_{t-1}\left(G\right)$
      end for
      $G_t = argmax_{G\in\mathcal{G}}U_{G,t}$
      Select $G_t$ append the rewards vector $\mathbf{y}_t$ by the observed rewards.
      Set $\hat{\mathbf{K}}_t \leftarrow \lambda\mathbf{I} + \sum_{i\le t}\mathbf{g}_{GNN}\left(G_i;\theta^0\right)\mathbf{g}_{GNN}\left(G_i;\theta^0\right)/mt$
      Calculate $\theta^{(J)}_t$ = TrainGNN$\Big(m, J, \eta, \lambda, \theta^0, \left(G_i, y_i\right)_{i\le t}\Big)$
  end for
```

makes further use of the underlying reward observations to improve performances. Consider the two extreme scenarios that can make an arm i played at time t: it has the highest UCB. Either its average estimate is high, which means it is empirically the best arm to play, or the exploration term is high, which means we want more information on it. When the exploration term is high, we want to observe a reward for it to reduce the uncertainty.However, we don't have to pull this arm directly to get an observation: we may as well pull any of its neighbors, especially one with higher empirical rewards, and reduce the bias term all the same.

Algorithm 4 EGGRIND

```
Require: $m, J, \eta, \lambda, \beta_t, T, G_f$
  Initialize network parameters to a random $\theta^0$, and $\hat{\mathbf{K}}_0 = \sigma^2\mathbf{I}$
  for $t = 1 \dots T$ do
      for $G \in \mathcal{G}$ do
          $\hat{\sigma}^2_{t-1} \leftarrow \mathbf{g}^T_{GNN}\left(G;\theta^0\right)\hat{\mathbf{K}}^{-1}_{t-1}\mathbf{g}_{GNN}\left(G;\theta^0\right)/m$
          $U_{G,t} \leftarrow f_{GNN}\left(G;\theta^{(J)}_{t-1}\right) + \beta_t\hat{\sigma}_{t-1}\left(G\right)$
      end for
      $G^u_t = argmax_{G\in\mathcal{G}}U_{G,t}$
      $G_t = argmax_{G\in neighbor(G^u_t)}f_{GNN}\left(G;\theta^{(J)}_{t-1}\right)$
      Select $G_t$ append the rewards vector $\mathbf{y}_t$ by the observed rewards.
      Set $\hat{\mathbf{K}}_t \leftarrow \lambda\mathbf{I} + \sum_{i\le t}\mathbf{g}_{GNN}\left(G_i;\theta^0\right)\mathbf{g}_{GNN}\left(G_i;\theta^0\right)/mt$
      Calculate $\theta^{(J)}_t$ = TrainGNN$\Big(m, J, \eta, \lambda, \theta^0, \left(G_i, y_i\right)_{i\le t}\Big)$
  end for
```

4.3 Result

From figure 1, we know that GGRIND and EGGRIND outperform the phased-elimination and normal UCB. The reason is that we use the social graph to help our algorithm to gather more information. Every time we pull an arm, we have more reward information to estimate the reward function. Thus, GGRIND and EGGRIND have better performance.

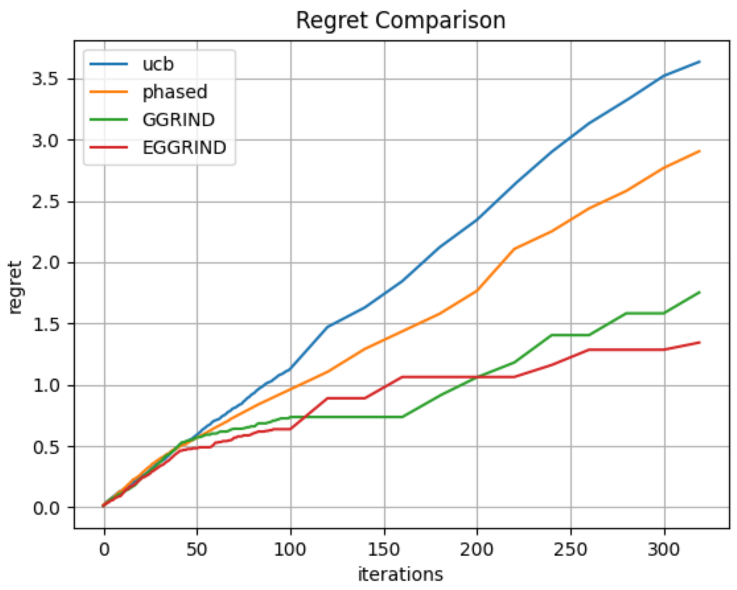

Figure 1: Compare the regrets among different algorithms

5 Constrained Graph Neural Network Bandits

5.1 Introduction to Constrained GNN Bandits

In practical applications, we will inevitably need to consider the energy consumption of the experiments. That is to say, in each round of interaction between the agent and the environment, we will receive a cost to represent the energy consumption of our interaction. In order to limit such an energy consumption can not be too large, we set a budget, indicating a period of time the total energy consumption limit. In practical applications, we not only need the regret of the algorithm to ensure convergence, but we also hope that such an energy consumption can be as small as possible, preferably less than our given budget (that is, zero violation), such a requirement is the design motivation of our Constrained Graph Neural Network(GNN) .

This design has broad application prospects in predicting molecular properties (energy, chemical properties) and predicting drug properties. Because in the above applications, in order to obtain training data by experiments and put these predicted products into production, often require a large amount of consumption, the introduction of the concept of budget constraints is very meaningful for reducing experimental consumption.

5.2 Problem Setting

5.2.1 Stochastic Contextual Bandit

We consider a stochastic contextual bandit model with an agent and a set of N arms, the set of arms is denoted as: $\mathcal{N} \triangleq [N]$. In each round t, within the given horizon T, the agent can observe a context $c(t)$ drawn from the context $\mathcal{C}$ with an unknown distribution. The contexts $c(t)$ are independent and identically distributed ($i.i.d.$) across rounds.

5.2.2 General Nonlinear Rewards & Costs

After arm selection in round t, the agent will receive reward $R(c(t), j)$ and $U(c(t), j)$ through interaction with jth arm. Both the reward and the cost depend on the context $c(j)$ and the time t.

Specifically,

$$R(c(t), j) \triangleq r(c(t)) + \eta(t),$$
$$U(c(t), j) \triangleq u(c(t)) + \xi(t),$$

where $\forall c(t) \in \mathcal{C}, j \in \mathcal{N}$, the mean reward $r(c(t), j) \triangleq E[R(c(t), j)]$, and the mean cost $u(c(t), j) \triangleq E[U(c(t), j)]$; $\eta(t), \xi(t)$ are additive noises following zero-mean Gaussian distributions. It is noteworthy that both the mean reward function $r(\cdot, \cdot)$ and the mean cost function $u(\cdot, \cdot)$ are unknown to the agent and are *general nonlinear* functions of (context, arm-index) pair.

5.2.3 Problem Formulation

Similar to the bandit problem, we aim to maximize the cumulative rewards over the horizon. During each round, the reward we obtain is formed by the reward of the arm we selected. In the Constrained GNN Bandit problem, in addition to maximizing the cumulative reward, we also need to control the cost of each round as small as possible, ideally smaller than the budget we set. In summary, our questions are set as follows :

$$\max_{a} \quad \mathbb{E}[\sum_{t=1}^{T} R(c(t), a_t)]$$
$$\text{s.t.} \quad \mathbb{E}[\sum_{t'=1}^{\tau} U(c(t'), a_{t'})] \leq \mathbb{E}[\sum_{t'=1}^{\tau} B(t')], \forall \tau,$$

where the vector $a \in \mathbb{R}^T, a = [a_1, a_2, \cdots a_T]$, a_t represents the action index we select in round t, $t = 1, 2, \cdots, T$. And budget $B(t) \in [0, b_{\max}]$ with $\mathbb{E} = b, \forall t \in [T]$. For simplicity, we consider constraints for one type of cost $U(\cdot, \cdot)$.

5.3 Algorithm Design

In order to solve such a problem, we have made some innovations based on Neural-PD (Shangshang et al. [2023]), the algorithm apply multi-layer perceptrons to estimate the mean reward and mean cost of each arms and use primal-dual algorithm to choose the optimal arms. However, this algorithm cannot be used to solve the Graph Structured problem.

Therefore, we propose a new algorithm **GNN-PD** which can be applied to solve the Constrained GNN Bandits problem. Since the Neural-PD algorithm is used to solve the Neural Constrained Combinatorial Bandits problem, our improvements mainly focus on how to introduce the GNN network.

In the algorithm we design, we use GNN to estimate the mean reward and mean cost of each arm and use Primal-Dual algorithm to control the balance between them, which means to achieve reward maximization while subjecting to anytime cumulative constraints. The obtained algorithm is as follows in Algorithm 5:

5.4 Theoretical Analysis

To evaluate the performance of GNN-PD Algorithm, we analyse its the regret and constraint violation. Firstly, we define some concepts to evaluate regret and constraint violation:

Definition of Regret: For any round $\tau \in [T]$,

$$\mathcal{R}(\tau) \triangleq \mathcal{R}^* - \mathbb{E}[\sum_{t=1}^{\tau} R(c(t), a_t) X_j(t)]$$

Definition of Constraint Violation: For any round $\tau \in [T]$,

$$\mathcal{V}(\tau) \triangleq [E(\sum_{t=1}^{\tau}(U(c(t), a_t) - B(t))]^+$$

Algorithm 5 Graph-Neural-Network-Based Primal-Dual (GNN-PD) Algorithm

```
Require: $T, f_{\text{GNN},r}, f_{\text{GNN},u}, \boldsymbol{\theta}_{r,0}, \boldsymbol{\theta}_{u,0}, \lambda, \eta$, Scaling factors $\gamma_t$;
        $Q(1) \leftarrow 0, \Sigma_{r,0} \leftarrow \lambda\mathbf{I}, \Sigma_{u,0} \leftarrow \lambda\mathbf{I}$.
  for $t = 1, \cdots, T$ do
      %% The agent performs all the following steps.
      Observation: Obtain Context $C(t)$ and the budget $B(t)$.

      Parameter Setup: Set tunable parameter
          $\epsilon_t \leftarrow O(\log(1+T)/\sqrt{t})$,
          $V_t \leftarrow O(\log(1+T)\sqrt{t})$

      Gradient Calculation: For each arm $j \in \mathcal{N}$,
          $g_{r,j} \leftarrow g_r(c(t), j|\boldsymbol{\theta}_{r,t-1})$;
          $g_{u,j} \leftarrow g_u(c(t), j|\boldsymbol{\theta}_{u,t-1})$.

      Primal Update: Optimistically estimate $r(c(t), j)$, pessimistically estimate $u(c(t), j), \forall j \in$
  $\mathcal{N}$ :
          $\hat{R}(c(t), j) \leftarrow f_{\text{GNN},r}(c(t), j|\boldsymbol{\theta}_{r,t-1}) + \frac{\gamma_{t-1}}{\sqrt{m}}||g_{r,j}||_{\Sigma^{-1}_{r,t-1}}$;
          $\hat{U}(c(t), j) \leftarrow f_{\text{GNN},u}(c(t), j|\boldsymbol{\theta}_{u,t-1}) - \frac{\gamma_{t-1}}{\sqrt{m}}||g_{u,j}||_{\Sigma^{-1}_{u,t-1}}$;

  Clip $\hat{R}(c(t), j)$ and $\hat{U}(c(t), j)$ to interval $[0, 1]$.

      MaxWeight: Select arms $a_t$:
          $a_t = \text{argmax}_{a\in\mathcal{N}} V_t\hat{R}(c(t), a) - Q(t)\hat{U}(c(t), a)$,

      Feedbacks: Get rewards and costs of selected arms:
          $R(c(t), a_t), U(c(t), a_t)$.

      Dual Update: Update the queue length as follows:
          $Q(t+1) \leftarrow \Big[Q(t) + \hat{U}(c(t), a_t) - B(t) + \epsilon_t\Big]^+$.

      Statistics Update:
          $\Sigma_{r,t} \leftarrow \Sigma_{r,t-1} + \frac{1}{m} g_{r,a_t} g^T_{r,a_t}$
          $\Sigma_{u,t} \leftarrow \Sigma_{u,t-1} + \frac{1}{m} g_{u,a_t} g^T_{u,a_t}$

  Obtained $\boldsymbol{\theta}_{r,t}, \boldsymbol{\theta}_{u,t}$ by training MLPs $f_{\text{GNN},r}, f_{\text{GNN},u}$ using collecte feedbacks.
  end for
```

According to the theoretical analysis of Neural-PD, we can get the estimation of the performance of GNN-PD Algorithm:

$$\mathcal{R}(\mathcal{T}) = \tilde{O}(\sqrt{T})$$
$$\mathcal{V}(\mathcal{T}) = O(1)$$

The results indicate that the regret bound of the algorithm is sublinear, which means it can converge to the optimal arm. Also, the algorithm can achieve a zero constraint violation.

In detail, the constraint violation of the algorithm satisfies the equation below:

$$\mathcal{V}(\tau) = \begin{cases} \tilde{O}(\log^2(1+T)) & \tau \leq \tilde{O}(\log^2 T) \\ 0 & \text{otherwise} \end{cases}$$

That means GNN-PD requires $\tilde{O}(\log^2(T))$ rounds to reach zero constraint violation because it takes a necessary online learning procedure to estimate the cost function. That is, if we know the cost function or it is trivial (e.g. constant), we an eliminate the dependence on T.

5.5 Numerical Results

In order to verify the correctness of the above theoretical analysis, we simulate the performance of the algorithm through program simulation in this section. Here is the setting of hyperparameters below:

Graph Neural Network In the following, we choose a two-layer MLP with width $m = 2048$ to construct a Graph Neural Network (GNN). We use Adam optimizer to optimize GNN and set the regularization parameter $\lambda = 1$, step size $\eta = 10^{-3}$.

Synthetic Dataset We consider a contextual graph bandit model: number of arms $N = 200$, number of nodes of each graph: $n = 5$. Dimension of node feature: $d = 100$. Therefore, the dimension of graph feature $D = 500$ and the context $c(t) \in \mathbb{R}^{N\times D} = \mathbb{R}^{200\times 500}$. The edges in these graphs are randomly generated.

Reward Model The mean reward of each arm is setting as: $r(c(t), j) = |\left\langle \boldsymbol{\theta}^r, c_j^T(t)\right\rangle|$, where $c_j(t)$ is the jth row of $c_(t)$ and column vector $\boldsymbol{\theta}^r = [\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}}, \cdots] \in \mathbb{R}^D = \mathbb{R}^{500}$.

Constraint Model The mean reward of each arm is setting as: $u(c(t), j) = |\left\langle \boldsymbol{\theta}^u, c_j^T(t)\right\rangle|$, where $c_j(t)$ is the jth row of $c(t)$ and column vector $\boldsymbol{\theta}^u = [\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}}, \cdots] \in \mathbb{R}^D = \mathbb{R}^{500}$.

Noises The output of reward and cost we set is with a Gaussian noise. And the Gaussian noise is subject to a 0 mean Gaussian distribution and the variance is $\sigma^2 = 10^{-4}$.

Budget In our setting, the budget of each round is always a constant, its value is the mean cost of all the arms in the corresponding round: $B(t) = \sum_{j=1}^{N} u(c(t), j)/N$.

Under the above setting, we obtain the following results through simulation as Figure 2 and Figure 3:

In Fig. 2, we present the regret performances of our algorithm GNN-PD by blue line, while the green line represents the exploitation strategy. From the diagram, we can conclude that our algorithm can show the convergence trend of sublinear, which also verifies the proof that the algorithm converges to $\tilde{O}(\sqrt{T})$ in our theoretical analysis. And Fig. 3 shows that our algorithm can achieve the performance of zero constraint violation in a short round.

The above simulation experiments prove that our algorithm can not only converge to the sublinear regret bound, but also achieve zero-violation in a short round. These results prove the excellent performance of our algorithm GNN-PD in Constraint Graph Structured Problem.

6 Conclusion & Future Work

In this report, we generalize the content in the chosen topic and reproduce the code for it. Starting from the setting of this paper, we expand our work into two parts: GNN bandit with graph feedback and constraint GNN bandit. For GNN bandit with graph feedback, we have two different UCB-type algorithms. From the point of view of the exploration, we have different designs in the exploration term. In future work, we can explore how to better correlate the exploration term with the structure of the social graph, instead of simply transferring accounts with degrees.

For the Constrained GNN Bandits Algorithm, we can try to make some innovations on the current method of control: primal-dual to improve its performance, and propose a more detailed theoretical derivation for the performance of the algorithm. Moreover, we find that if the number of edges of the graph is too large, it may cause regret to become too large. Therefore, we hope to try to solve this problem: try to effectively reduce the regret bound when the number of edges of the graph is large. In addition, we can try to apply our Constrained GNN Bandits Algorithm in some practical scenarios, such as the evaluation of protein or chemical material properties and medicine performance.

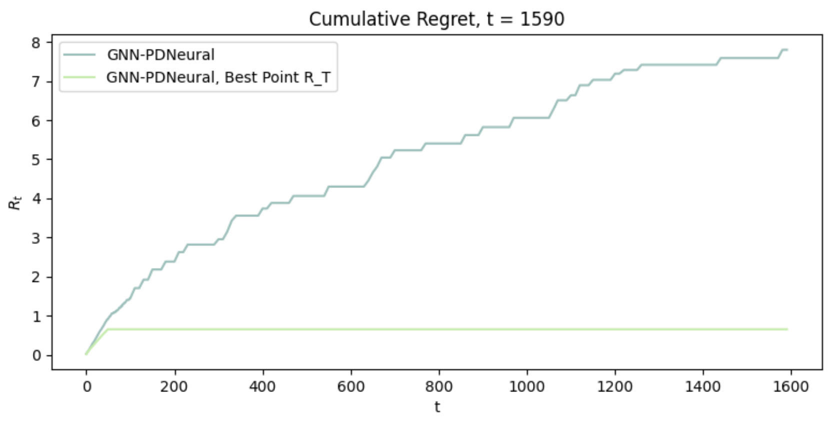

Figure 2: GNN-PD Regret

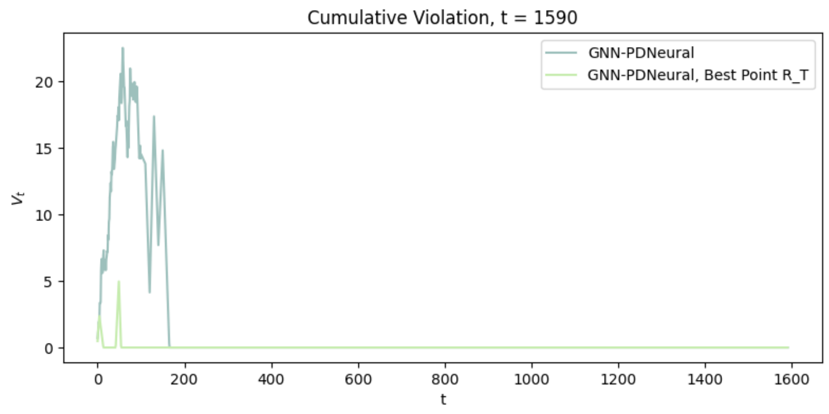

Figure 3: GNN-PD Constraint Violation

7 Contribution

- Xu Bowen: Read and discuss the original paper and related papers. Design the algorithms of Constrained Graph Neural Network Bandits Algorithm: GNN-PD and its theoretical analysis. Implement the code simulation of GNN-PD. Participate in the final report writing.
- Zhang Tianyi: Read and discuss the original paper and related papers. Reproduce the original paper code. Design the algorithms in graph feedback parts. Participate in the final report writing.
- Zhong Wenfu: Read and discuss the original paper and related papers. Implemented experiments of algorithms. Design the algorithms in graph feedback parts. Understand and complete the theoretical derivation of the paper .Participate in the final report writing.

References

N. Alon, N. Cesa-Bianchi, C. Gentile, and Y. Mansour. From bandits to experts: A tale of domination and independence. *Advances in Neural Information Processing Systems*, 26, 2013.

S. Bian, S. Wang, Y. Tang, and Z. Shao. Social-aware edge intelligence: A constrained graphical bandit approach. In *GLOBECOM 2022-2022 IEEE Global Communications Conference*, pages 6372–6377. IEEE, 2022.

S. Caron, B. Kveton, M. Lelarge, and S. Bhagat. Leveraging side observations in stochastic bandits. *arXiv preprint arXiv:1210.4839*, 2012.

S. Chowdhury and A. Gopalan. On kernelized multi-armed bandits, Apr 2017.

A. Durand, C. Achilleos, D. Iacovides, K. Strati, G. D. Mitsis, and J. Pineau. Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis. In *Machine learning for healthcare conference*, pages 67–82. PMLR, 2018.

K. Guo and M. J. Buehler. A semi-supervised approach to architected materials design using graph neural networks. *Extreme Mechanics Letters*, 41:101029, Oct 2020. doi: 10.1016/j.eml.2020.101029. URL `http://dx.doi.org/10.1016/j.eml.2020.101029`.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

D. Jiang, Z. Wu, C.-Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu, and T. Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13(1), Feb 2021. doi: 10.1186/s13321-020-00479-8. URL `http://dx.doi.org/10.1186/s13321-020-00479-8`.

P. Kassraie, A. Krause, and I. Bogunovic. Graph neural network bandits. *arXiv preprint arXiv:2207.06456*, 2022.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

S. Mannor and O. Shamir. From bandits to experts: On the value of side-observations. *Advances in Neural Information Processing Systems*, 24, 2011.

W. Shangshang, B. Simeng, L. Xin, and S. Ziyu. Neural constrained combinatorial bandits. *Infocom 2023 IEEE International Conference on Computer Communications*, 2023.

N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *IEEE Transactions on Information Theory*, page 3250–3265, Jan 2012. doi: 10.1109/tit.2011.2182033. URL `http://dx.doi.org/10.1109/tit.2011.2182033`.

S. S. Villar, J. Bowden, and J. Wason. 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):199, 2015.

Q. Wu, H. Wang, Q. Gu, and H. Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538, 2016.

Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen. Graph convolutional networks with markov random field reasoning for social spammer detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, page 1054–1061, Jun 2020. doi: 10.1609/aaai.v34i01.5455. URL `http://dx.doi.org/10.1609/aaai.v34i01.5455`.

D. Zhou, L. Li, and Q. Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.