Graph Neural Network Bandits

Parnian Kassraie, Andreas Krause, Ilija Bogunovic

Summary

GNNs can be efficiently employed to solve bandit problems on a large domain of large graphs, e.g. for drug discovery. They save you from exponential dependency on the number of nodes.

- Problems such as protein design, molecule and drug discovery, involve solving

$$G^* \in \arg\max_{G\in\mathcal{G}} f^*(G)$$

where $f^\star$is unknown, and sampling from it is costly.

- Challenges in such applications are scaling to large domains, and to graphs with many nodes.

- We model this as bandit optimization on graphs, and propose how GNNs can be used to design scalable algorithms.

Problem Setting

- Iteratively evaluate the function via noisy observations

$$y_t = f^*(G_t) + \epsilon_t$$

i.i.d. zero-mean sub-Gaussian noise

- $\mathcal{G}$ Finite set of undirected graphs with N nodes
  Each graph has node features $\boldsymbol{h}_G = (\boldsymbol{h}_{G,j})_{j=1}^N \in \mathbb{R}^{Nd}$

- f^* Real valued, regular, invariant to node permutations

$$f^*(c\cdot G) = f^*(G) \quad \forall G\in\mathcal{G},\, c\in P_N$$

$$||f^\star||_{\bar{k}} \leq B$$

- Choose a kernel that is permutation invariant, but also efficient to compute

$h_{c\cdot G} := (h_{G,c(j)})_{j=1}^N$

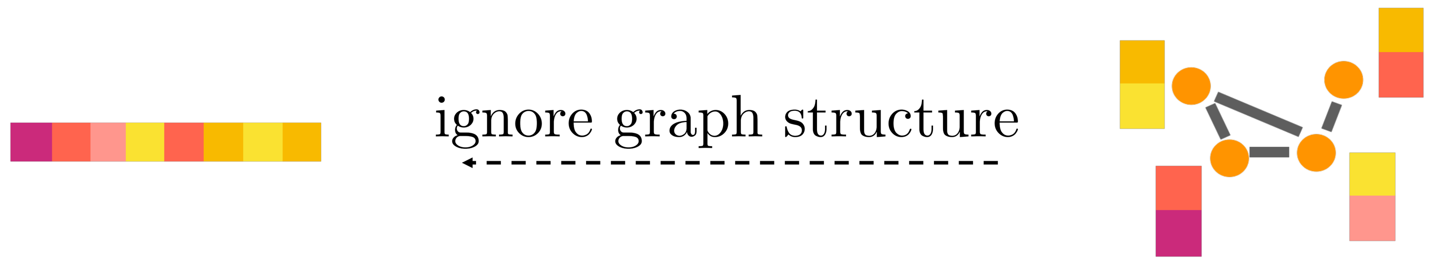
$$\bar{k}(G,G') = \frac{1}{|P_N|^2}\sum_{c,c'\in P_N} k(\bar{\boldsymbol{h}}_{c\cdot G}, \bar{\boldsymbol{h}}_{c'\cdot G'})$$

- Setting k as an NTK creates symmetries which gives

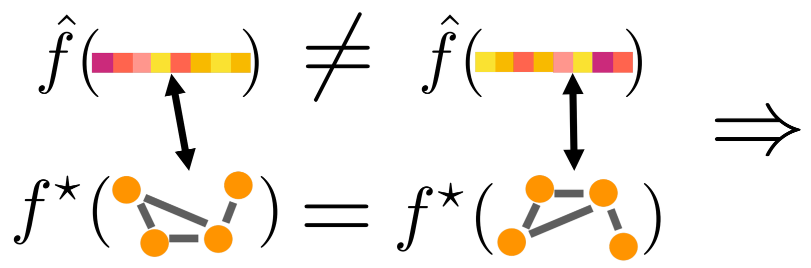
$$\bar{k} = k_{\mathrm{GNN}} \qquad k = \frac{1}{N}\sum_{j=1}^{N} k_{\mathrm{NN}}\left(\bar{h}_{G,j}, \bar{h}_{G',j}\right)$$

- The Graph Neural Tangent Kernel is expressive, efficient to compute, and additive permutation invariant.

Permutation Invariance Trick



Many block-permutations of this vector yield the same reward. If the estimator is agnostic to this strucutre,



⇢ Use a reward estimator which is invariant to these block permutations.

$$\text{GNN with 1 conv layer,} \qquad f_{\mathrm{GNN}}(G;\boldsymbol{\theta}) = f_{\mathrm{GNN}}(c\cdot G;\boldsymbol{\theta})$$

GNN Confidence Sets

- Train $f_{\mathrm{GNN}}(G;\boldsymbol{\theta})$ to estimate $f^*(G)$

Lazy initialization + SGD on

$$\mathcal{L}(\theta) = \frac{1}{t}\sum_{i<t}\left(f_{\mathrm{GNN}}(G_i,\theta) - y_i\right)_2^2 + m\lambda\|\theta-\theta^0\|_2^2$$

$$\hat{\mu}_{t-1}(G) := f_{\mathrm{GNN}}(G;\hat{\boldsymbol{\theta}}_{t-1})$$

- Use $\nabla_{\boldsymbol{\theta}} f_{\mathrm{GNN}}$ to capture the uncertainty over the outputs

$$\hat{\sigma}^2_{t-1}(G) := \frac{\nabla f^T_{\mathrm{GNN}}(G)}{\sqrt{m}}\left(\lambda\boldsymbol{I} + \boldsymbol{H}_{t-1}\right)^{-1}\frac{\nabla f_{\mathrm{GNN}}(G)}{\sqrt{m}}$$

- Construct Confidence sets,

$$\mathcal{C}_{t-1}(G,\delta) = [\hat{\mu}_{t-1}(G) \pm \beta_t\hat{\sigma}_{t-1}(G)]$$

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

Theorem

GNN confidence sets are valid if the used network is wide enough, i.e. with probability greater than $1-\delta$

$$f^*(G) \in \mathcal{C}_{t-1}(G,\delta), \qquad \forall G\in\mathcal{G}$$

- Confidence sets as a proxy for the reward!

Regret Guarantee

- Cumulative regret as a measure of sample efficiency

$$R_T = \textstyle\sum_{t=1}^T f^*(G^*) - f^*(G_t)$$

$$R_T/T \to 0 \text{ as } T\to\infty$$

GNN-PE
- Has episodic structure
- Uses $\mathcal{C}_{t-1}(G,\delta)$
  - To maintain set of plausible maximizer graphs
  - To choose the next graph

Theorem

Suppose $f^* \in \mathcal{H}_{k_{\mathrm{GNN}}}$ with a norm bounded by B. Then with probability greater than $1-\delta$,

$$R_T = \tilde{\mathcal{O}}\left(T^{\frac{2d-1}{2d}}\log^{\frac{1}{2d}} T\left(B + \sqrt{\log|\mathcal{G}|/\delta}\right)\right)$$

if the used GNN is wide enough. $\log N$ dependency

- Naively using a NN gives

$$\tilde{\mathcal{O}}\left(T^{\frac{2Nd-1}{2Nd}}\log^{\frac{1}{2Nd}} T\right)$$

Experiments

- Training GNNs in lazy regime is challenging. Stopping criterion for gradient descent plays a crucial role in achieving sublinear regret.

⇢ We devise a history-dependent stopping criterion.