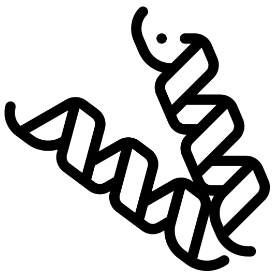
Graph Neural Network Bandits

Parnian Kassraie, Andreas Krause, Ilija Bogunovic

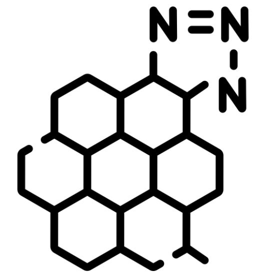
ETH zürich UCL

Learning on Graph Structured Data



Protein Design

Drug Discovery



Molecule Synthesis

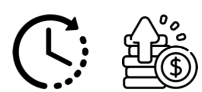
$$G^* \in \arg\max_{G \in \mathcal{G}} f^*(G)$$

Choose G_t

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

Repeat

Costly



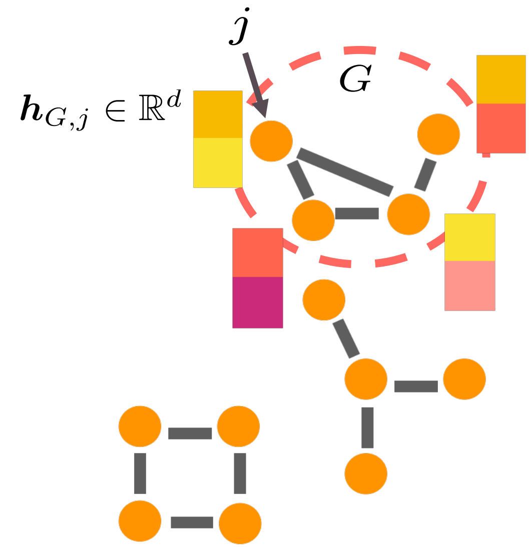
→ Need to be sample efficient!

→ Model as a bandit problem

image source: flaticon

ETH zürich UCL

Problem Setting



$\mathcal{G}$ Finite set of undirected graphs with N nodes
Each graph has node features

f^* Real-valued and regular (contained within an RKHS)
Invariant to node permutations

$$f^*(c \cdot G) = f^*(G)$$

if you indexed the graphs in your dataset in a different way, it would not matter.

Can you use GNNs to efficiently maximize such functions on such domains?

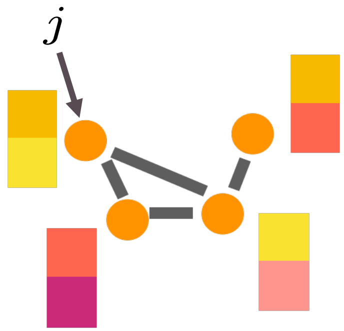
Bandit Objective $R_T = \sum_{t=1}^{T} f^*(G^*) - f^*(G_t)$

Sublinear if converges to maxima
Small if sample efficient

Why a GNN?

$\boldsymbol{h}_{G,j} \in \mathbb{R}^d$

Space of N-node graph with d-dim node features

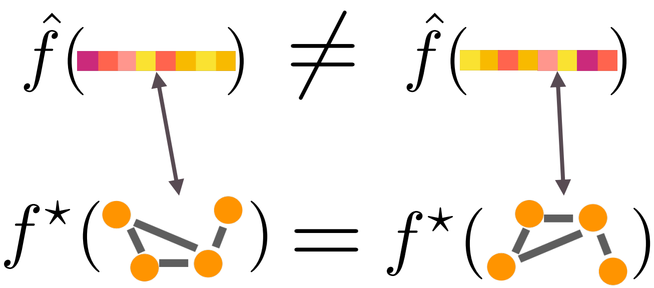


ignore graph structure

$\mathbb{S}^{Nd-1}$

Structure-agnostic reward estimator

while,



$\Rightarrow$ Sample inefficiency

Use a reward estimator which is invariant to these block permutations,

GNN with 1 conv layer,

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

How to use GNNs

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

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

run SGD on

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

2) Use $\nabla_{\boldsymbol{\theta}} f_{\text{GNN}}$ to capture the uncertainty over these estimates

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

gram matrix $\boldsymbol{H}_{t-1}$

width m

3) Use confidence sets as a guide to choose actions

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

Because:

Theorem

GNN confidence sets are valid if the used network is wide enough, i.e. with high probability

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

Our algorithm

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 $f^* \in \mathcal{H}_{k_{\mathrm{GNN}}}$ and has a bounded norm. If the used GNN is wide enough, then with high probability

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

$\log(N)$ and $\sqrt{\log(|\mathcal{G}|)}$ dependency

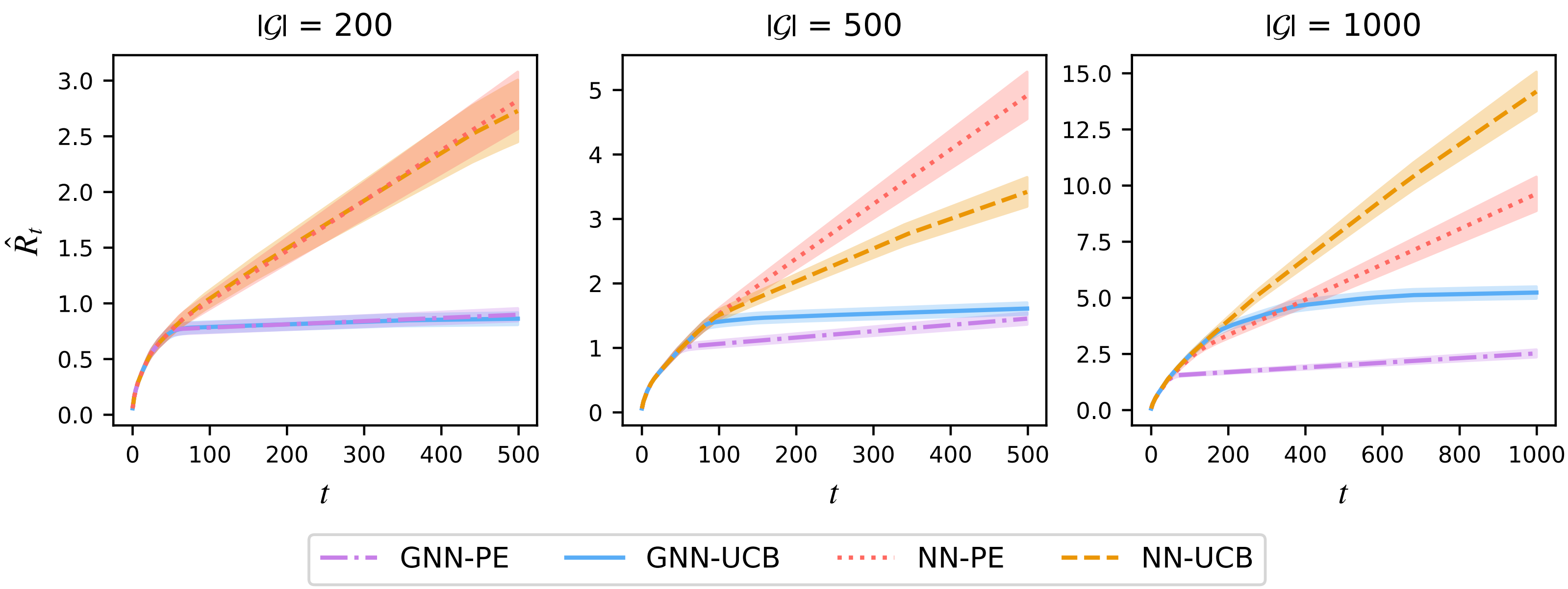
Naively using Neural UCB gives

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

T: the bandit horizon,
N: the number of nodes in each graph,
d: the dimension of node features

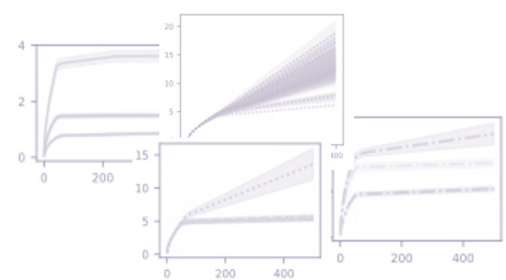
Experiments

Domain: Erdos-Rényi random graphs
Objective: Sampled from $\mathrm{GP}(0, k_{\mathrm{GNN}})$



✓ Outperforms NN methods
✓ Scales well to domains of large graphs
✓ Scales well to large domains of graphs

Checkout the paper for more



ETH zürich UCL

Thank you.