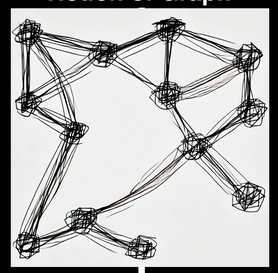
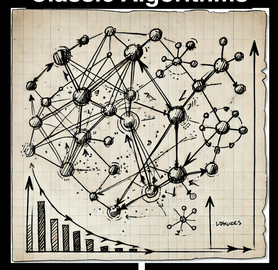
Have We Gone Too Far?

(Pics generated by ChatGPT)

Notion of Graph

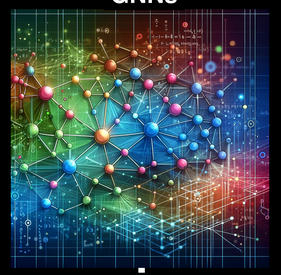


Classic Algorithms



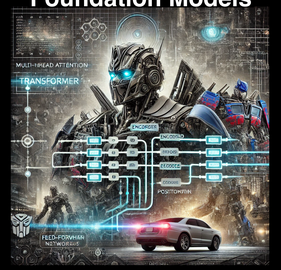
(What Happens?)

GNNs



(Maybe?)

Foundation Models



Codes

How Graph Neural Networks Learn: Lessons from Training Dynamics

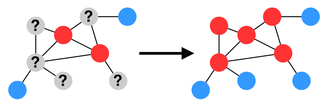
Chenxiao Yang[1], Qitian Wu[1], David Wipf[2], Ruoyu Sun[3], Junchi Yan[1]

[1] Shanghai Jiao Tong University [2] Amazon Web Services [3] The Chinese University of Hong Kong

ICML Vienna 2024

Embarrassingly Simple Algorithm > GNNs

- **Setup**



Adjacency $\mathbf{A} \in \mathbb{R}^{n \times n}$ (n = # nodes), labels $\mathbf{Y} \in \mathbb{R}^{n_l}$ (n_l: # labeled nodes), predictions for labeled/unlabeled nodes $[\mathbf{F}, \mathbf{F}'] \in \mathbb{R}^{n}$.

- **Residual Propagation Algorithm**

Forward message passing is all this algorithm does:

$$[\mathbf{R}_{t+1}, \mathbf{R}'_{t+1}] = -\eta \mathbf{A}^K [\mathbf{R}_t, \mathbf{0}] + [\mathbf{R}_t, \mathbf{R}'_t]$$

Residuals, a.k.a. errors, defined as $\mathbf{R} = \mathbf{F} - \mathbf{Y} \in \mathbb{R}^{n_l}$ (for training set), $\mathbf{R}' = \mathbf{F}' - \mathbf{0} \in \mathbb{R}^{n-n_l}$ (for testing set). At initialization, $\mathbf{R}_0 = -\mathbf{Y}$, $\mathbf{R}'_0 = \mathbf{0}$.

1. Interpretation: the prediction for an unlabeled node $\boldsymbol{x}'$

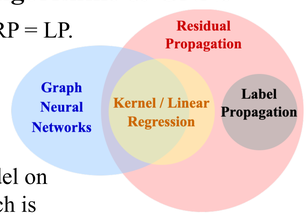
(Ground-truth) label propagation

$$f_{t+1}(\boldsymbol{x}') = f_t(\boldsymbol{x}') + \eta \sum_{\boldsymbol{x} \in \mathbf{X}} \mathbf{A}^k(\boldsymbol{x}, \boldsymbol{x}')(y(\boldsymbol{x}) - f_t(\boldsymbol{x}))$$

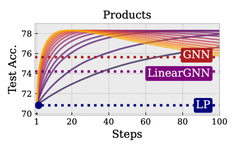
2. Extensions: $\mathbf{A}^k$ could be replaced by other matrices (not necessarily PSD or symmetric) and incorporate node feature, e.g. $\mathbf{A}^2\mathbf{K}(\mathbf{X}, \mathbf{X})\mathbf{A}^5$.

- **Connection with Classic Algorithms & GNNs**

1. **Label Propagation:** first step of RP = LP.
2. **Kernel Regression:** RP relax the constraints on kernel matrices; RP does not need to converge; complexity $\mathcal{O}(n^3) \rightarrow \mathcal{O}(e)$.
3. **Network Embedding:** linear model on spectral decomposition of $\mathbf{A}$ (which is computationally very expensive) induces RP.
4. **GNN:** discussed later ….



- **Partial Result**



- **Implications**

1. **Performance of classic algorithms can be boosted to the same level as GNNs.**
2. **Explanation of generalization can be potentially decoupled with the model.**

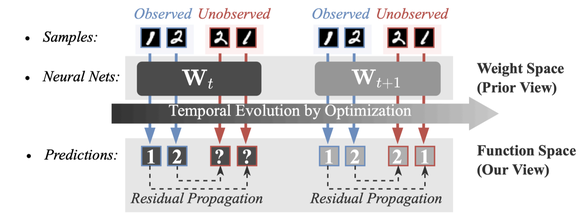
Optimization and Generalization of GNNs

Q: What is the central commonality of these effective methods?
→ The particular way the graph structure is leveraged in training matters !

- **Graph Implicit Bias in GNNs' Optimization**

GD Dynamics: $\dfrac{\partial [\mathbf{R}_t, \mathbf{R}'_t]}{\partial t} = -\eta\, \boldsymbol{\Theta}_t(\bar{\mathbf{X}}, \bar{\mathbf{X}}; \mathbf{A})\, [\mathbf{R}_t, \mathbf{0}]$

Def (Node-Level GNTK) $\boldsymbol{\Theta}_t^{(\ell)}(\boldsymbol{x}, \boldsymbol{x}'; \mathbf{A}) = \nabla_{\mathbf{W}} f(\boldsymbol{x}; \mathbf{A})^\top \nabla_{\mathbf{W}} f(\boldsymbol{x}'; \mathbf{A})$



Role of Graph Structure in Optimization

(*Transformation*) $\bar{\boldsymbol{\Theta}}^{(\ell)} = \boldsymbol{\Theta}^{(\ell-1)} \odot \dot{\boldsymbol{\Sigma}}^{(\ell)} + \bar{\boldsymbol{\Sigma}}^{(\ell)}$

(*Propagation*) $\begin{cases} \boldsymbol{\Sigma}^{(\ell)} = \mathbf{A}\, \bar{\boldsymbol{\Sigma}}^{(\ell)} \mathbf{A} \\ \boldsymbol{\Theta}^{(\ell)} = \mathbf{A}\, \bar{\boldsymbol{\Theta}}^{(\ell)} \mathbf{A}. \end{cases}$

and illustrative examples showing cases where the GNTK is equivalent to some special forms of the adjacency.

- **Generalization Analysis**

Def (Alignment & Optimal Kernel)

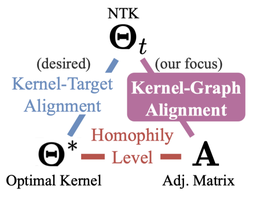
$A(\mathbf{K}_1, \mathbf{K}_2) \triangleq \langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F / (\|\mathbf{K}_1\|_F \|\mathbf{K}_2\|_F)$

$\boldsymbol{\Theta}^* \triangleq \bar{\mathbf{Y}} \bar{\mathbf{Y}}^\top$

Homophily Level: $A\left(\mathbf{A}, \boldsymbol{\Theta}^*\right)$

Kernel-Target Alignment: $A\left(\boldsymbol{\Theta}_t, \boldsymbol{\Theta}^*\right)$

Kernel-Graph Alignment: $A\left(\boldsymbol{\Theta}_t, \mathbf{A}\right)$



Result 1 (following standard analysis)

$$\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{P}}\left[l\left(f(\boldsymbol{x}), y\right)\right] = \mathcal{O}\left(\sqrt{1 - cn_l^{-1} A(\mathbf{A}, \boldsymbol{\Theta}^*)} + \sqrt{n_l^{-1} \log(\delta^{-1})}\right)$$

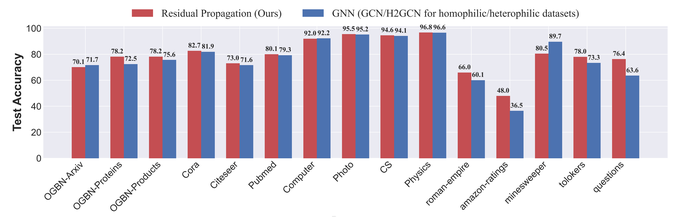
Result 2 (removing the iid assumption)

When the data generating distribution satisfies certain homophily properties, GNN is the Bayesian optimal model that minimizes the population risk.

Experimental Verification

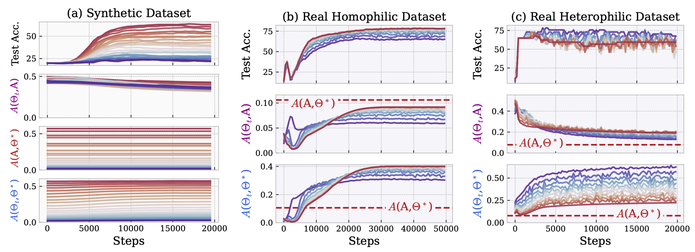
- **Evaluation of Residual Propagation**

Model	Feat.	Arxiv Validation	Arxiv Test	Proteins Validation	Proteins Test	Products Validation	Products Test	# Param.
MLP	X	57.65 ± 0.12	55.50 ± 0.23	77.06 ± 0.14	72.04 ± 0.48	75.54 ± 0.14	61.06 ± 0.08	$O(\ell m^2)$
LinearGNN	X, A	70.67 ± 0.02	69.39 ± 0.11	66.11 ± 0.87	62.89 ± 0.11	88.97 ± 0.01	74.21 ± 0.04	$O(de)$
GNN	X, A	73.00 ± 0.17	71.74 ± 0.29	79.21 ± 0.18	72.51 ± 0.35	92.00 ± 0.03	75.64 ± 0.21	$O(tm^2)$
LP	A	70.14 ± 0.00	68.32 ± 0.00	83.02 ± 0.00	74.73 ± 0.00	90.91 ± 0.00	74.34 ± 0.00	0
RP (ours)	A	71.37 ± 0.00	70.06 ± 0.00	85.19 ± 0.00	78.17 ± 0.00	91.31 ± 0.00	78.25 ± 0.00	0
Speedup / step		× 14.48		× 14.00		× 12.46		
Time to Acc.		× 0.01461		× 0.00008		× 0.00427		
Memory		× 0.094		× 0.363		× 0.151		



1. Simple non-parametric algorithm rivals with GNNs on 15 benchmarks (including homophilic and heterophilic ones).
2. RP is significantly more efficient and scalable (up to **10,000× speedup**, 10× less memory, easily run on **million-scale dataset**).
3. RP is especially effective on large datasets (where it secretly has more parameters than most GNNs).

- **Training Dynamics: How GNNs Handle Heterophily?**



1. GNN's NTK inherently aligns with the graph structure (i.e. kernel-graph alignment or KGA).
2. On homophilic graphs, **better KGA implies better generalization**.
3. GNN's NTK evolves during training and also gradually aligns with the optimal kernel (i.e. the ideal graph), indicating how GNN adaptively handles heterophily.