Graph Neural Tangent Kernel: Fusing Graph Neural Networks with Graph Kernels

Simon S. Du, Kangcheng Hou, Barnabas Poczos, Ruslan Salakhutdinov, Ruosong Wang, Keyulu Xu

Presenter: Jian Kang

04/01/2020

Outline

- **Introduction**
- Related Work
- Technical Details
  - Background: Infinite-Width Networks and Neural Tangent Kernel
  - Graph Neural Tangent Kernel (GNTK)
- Experimental Results
- Conclusion

Classification on Graphs

- Requirement: effectively exploit the structure of graphs
- Major classes of methods
  - Graph Kernels: build feature vectors based on combinatorial properties of input graphs
    - Examples: random walk kernel, WL subtree kernel, graphlet kernel
  - Graph Neural Networks: use multi-layer structures with non-linear activation functions to aggregate local information and to extract high-order features
    - Examples: GCN, GraphSAGE, GIN, JK-Net

Pros and Cons

Graph Kernels

- Pros
  - convex optimization: easy to train
  - explicit expressions: easy to analyze theoretical guarantees
- Cons
  - handcrafted features: not powerful enough to capture high-order information

Graph Neural Networks

- Pros
  - non-handcrafted high-order features
- Cons
  - non-convex optimization
    - bad stability
    - limited theoretical understanding

Pros and cons of GKs and GNNs are complementary.

Can we build a model that enjoys the best of both worlds?

Outline

- Introduction
- **Related Work**
- Technical Details
  - Background: Infinite-Width Networks and Neural Tangent Kernel
  - Graph Neural Tangent Kernel (GNTK)
- Experimental Results

Related Work: Connecting Kernels and NNs

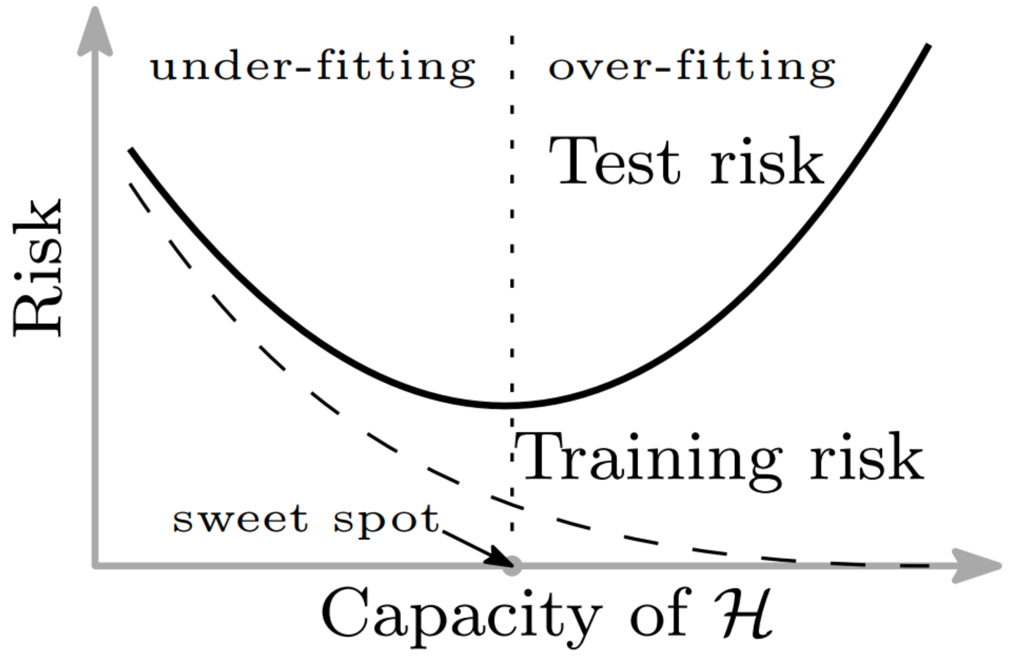
- NNs as Gaussian processes
  - single-layer fully-connected neural network (FCNN) [Neal, 1996]
  - multi-layer FCNN [Lee et al., 2018]
  - convolutional neural network (CNN) [Novak et al., 2019]
- NNs as neural tangent kernels
  - FCNN [Jacot et al., 2018]
  - CNN [Arora et al., 2019]

Outline

- Introduction
- Related Work
- **Technical Details**
  - **Background: Infinite-Width Networks and Neural Tangent Kernel**
  - Graph Neural Tangent Kernel (GNTK)
- Experimental Results

Training Error vs. Test Error: Traditional Machine Learning

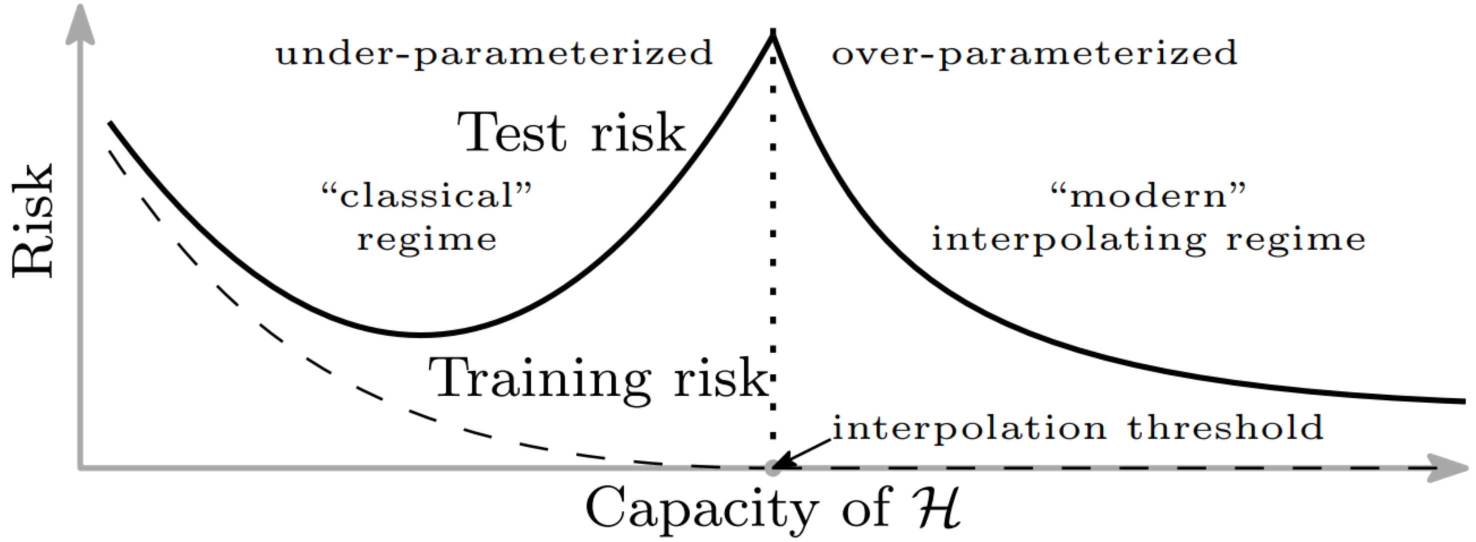
- Goal: find a ‘sweet spot’ for the model complexity
  - ‘big’ enough to achieve reasonably good training error
  - ‘small’ enough to control the generalization gap*



* generalization gap = |training error - test error|

Training Error vs. Test Error: Deep Learning

- DNNs are over-parameterized -> high model complexity
- 'Double descent' phenomenon
  - training error: near zero
  - test error: low
- Question: how does an infinite-width network perform?



Infinite-Width Network

- Why infinite-width network?
  - infinite limit usually reveals interesting properties when studying a problem in math/physics
  - previous work on connecting infinite-width network with Gaussian processes (single-layer FCNN [Neal, 1996], multi-layer FCNN [Lee et al., 2018])

Infinite-Width Network

- Considering standard supervised learning,
  - n training data samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$
  - squred loss function $\ell(\boldsymbol{\theta}) = 1/2 \sum_{i=1}^n (f(\boldsymbol{\theta}, \boldsymbol{x}_i) - y_i)^2$
  - minimizing by gradient descent with infinitesimal step size (aka. gradient flow)
    - gradient flow: steepest descent curve in continuous time, defined as $\frac{d\boldsymbol{\theta}(t)}{dt} = -\nabla\ell\big(\boldsymbol{\theta}(t)\big)$
- Let $\boldsymbol{u}(t) = \big(f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)\big)_{i\in[n]}$ and $\boldsymbol{y} = (y_i)_{i\in[n]}$, then the dynamics of $\boldsymbol{u}(t)$ is equal to $-\boldsymbol{H}(t) \cdot (\boldsymbol{u}(t) - \boldsymbol{y})$
  - $[\boldsymbol{H}(t)]_{ij} =< \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{\theta}(t), \boldsymbol{x}_j\right)}{\partial \boldsymbol{\theta}} >$
  - easy to prove: simple differentiation with chain rule

Neural Tangent Kernel (NTK)

- Recall $[\boldsymbol{\Theta}(t)]_{ij} =< \frac{\partial f(\boldsymbol{\theta}(t), \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{\theta}(t), \boldsymbol{x}_j\right)}{\partial \boldsymbol{\theta}} >$
- If width goes to infinity,
  - $\boldsymbol{\Theta}(t)$ remains constant during training (i.e., equal to $\boldsymbol{\Theta}(0)$)
    - proof by induction, the variations of activations and weights shrinks as the width grows
    - see detailed proof in [Jacot et al., 2018]
  - under random initialization on infinite-width network, $\boldsymbol{\Theta}(0)$ converges to a deterministic kernel matrix $\boldsymbol{\Theta}$
- Neural Tangent Kernel (NTK): the deterministic kernel matrix $\boldsymbol{\Theta}$, where $[\boldsymbol{\Theta}(t)]_{ij} =< \frac{\partial f(\boldsymbol{\theta}, \boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}, \frac{\partial f\left(\boldsymbol{\theta}, \boldsymbol{x}_j\right)}{\partial \boldsymbol{\theta}} >$

NTK Formulas

- Building upon [Lee et al., 2018], let us define

$$\Sigma^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}',$$
$$\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2},$$
$$\Sigma^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\sigma(u)\,\sigma(v)\right].$$
$$\dot{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right]$$

- We have the final NTK expression of FCNN as $\Theta^{(L)}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h=1}^{L+1} \left(\Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \cdot \prod_{h'=h}^{L+1} \dot{\Sigma}^{(h')}(\boldsymbol{x}, \boldsymbol{x}') \right)$
  - written in a recursive manner, we have

$$\Theta_\infty^{(L+1)}(x, x') = \Theta_\infty^{(L)}(x, x')\dot{\Sigma}^{(L+1)}(x, x') + \Sigma^{(L+1)}(x, x')$$

- Can be derived by first-order Taylor expansion on NN function
- See detailed derivation in [Jacot et al., 2018] or [Arora et al., 2019]

Outline

- Introduction
- Related Work
- **Technical Details**
  - Background: Infinite-Width Networks and Neural Tangent Kernel
  - **Graph Neural Tangent Kernel (GNTK)**
- Experimental Results

Graph Neural Networks

- BLOCK operation
  - aggregate features over the neighborhood $\mathcal{N}(u) \cup \{u\}$ of node u
  - transform the aggregated features with non-linearity

$$\mathrm{BLOCK}^{(\ell)}(u) = \sqrt{\frac{c_\sigma}{m}} \cdot \sigma\left(\boldsymbol{W}_\ell \cdot c_u \sum_{v \in \mathcal{N}(u) \cup \{u\}} \boldsymbol{h}_v^{(\ell-1)}\right)$$

\# BLOCK = 1

$$\mathrm{BLOCK}^{(\ell)}(u) = \sqrt{\frac{c_\sigma}{m}} \sigma\left(\boldsymbol{W}_{\ell,2} \sqrt{\frac{c_\sigma}{m}} \cdot \sigma\left(\boldsymbol{W}_{\ell,1} \cdot c_u \sum_{v \in \mathcal{N}(u) \cup \{u\}} \boldsymbol{h}_v^{(\ell-1)}\right)\right)$$

\# BLOCK = 2

- READOUT operation: get the representation of an entire graph

$$\boldsymbol{h}_G = \mathrm{READOUT}\left(\left\{\boldsymbol{h}_u^{(L)}, u \in V\right\}\right) = \sum_{u \in V} \boldsymbol{h}_u^{(L)}$$

without jumping knowledge

$$\boldsymbol{h}_G = \mathrm{READOUT}^{\mathrm{JK}}\left(\left\{\boldsymbol{h}_u^{(\ell)}, u \in V, \ell \in [L]\right\}\right) = \sum_{u \in V} \left[\boldsymbol{h}_u^{(0)}; \ldots; \boldsymbol{h}_u^{(L)}\right]$$

with jumping knowledge

Graph Neural Tangent Kernel (GNTK)

- Goal: translate GNN architecture to GNTK
  - Key: how to translate BLOCK and READOUT operations
- Intuition: recall NTK formulas

$$\Sigma^{(0)}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}',$$

$$\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}, \boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2},$$

$$\Sigma^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\sigma(u)\,\sigma(v)\right].$$

$$\dot{\Sigma}^{(h)}(\boldsymbol{x}, \boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{(h)}\right)} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right]$$

$$\Theta_\infty^{(L+1)}(x, x') = \Theta_\infty^{(L)}(x, x')\dot{\Sigma}^{(L+1)}(x, x') + \Sigma^{(L+1)}(x, x').$$

Graph Neural Tangent Kernel (GNTK)

- Connect $\boldsymbol{x}^T\boldsymbol{x}$ with BLOCK and READOUT
  - Initialization

$$\left[\boldsymbol{\Theta}_{(1)}^{(0)}(G,G')\right]_{uu'} = \left[\boldsymbol{\Sigma}_{(1)}^{(0)}(G,G')\right]_{uu'} = \left[\boldsymbol{\Sigma}^{(0)}(G,G')\right]_{uu'} = \boldsymbol{h}_u^\top \boldsymbol{h}_{u'}$$

  - BLOCK operation

$$\left[\boldsymbol{\Sigma}_{(0)}^{(\ell)}(G,G')\right]_{uu'} = c_u c_{u'} \sum_{v\in\mathcal{N}(u)\cup\{u\}} \sum_{v'\in\mathcal{N}(u')\cup\{u'\}} \left[\boldsymbol{\Sigma}_{(R)}^{(\ell-1)}(G,G')\right]_{vv'},$$

$$\left[\boldsymbol{\Theta}_{(0)}^{(\ell)}(G,G')\right]_{uu'} = c_u c_{u'} \sum_{v\in\mathcal{N}(u)\cup\{u\}} \sum_{v'\in\mathcal{N}(u')\cup\{u'\}} \left[\boldsymbol{\Theta}_{(R)}^{(\ell-1)}(G,G')\right]_{vv'}.$$

  - READOUT operation

$$\Theta(G,G') = \begin{cases} \sum_{u\in V, u'\in V'} \left[\boldsymbol{\Theta}_{(R)}^{(\ell)}(G,G')\right]_{uu'} & \text{without jumping knowledge} \\ \sum_{u\in V, u'\in V'} \left[\sum_{\ell=0}^{L} \boldsymbol{\Theta}_{(R)}^{(\ell)}(G,G')\right]_{uu'} & \text{with jumping knowledge} \end{cases}.$$

$$\Sigma^{(0)}(\boldsymbol{x},\boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}',$$

$$\boldsymbol{\Lambda}^{(h)}(\boldsymbol{x},\boldsymbol{x}') = \begin{pmatrix} \Sigma^{(h-1)}(\boldsymbol{x},\boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x},\boldsymbol{x}') \\ \Sigma^{(h-1)}(\boldsymbol{x}',\boldsymbol{x}) & \Sigma^{(h-1)}(\boldsymbol{x}',\boldsymbol{x}') \end{pmatrix} \in \mathbb{R}^{2\times 2},$$

$$\Sigma^{(h)}(\boldsymbol{x},\boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\Lambda}^{(h)})} \left[\sigma(u)\,\sigma(v)\right].$$

$$\dot{\Sigma}^{(h)}(\boldsymbol{x},\boldsymbol{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\Lambda}^{(h)})} \left[\dot{\sigma}(u)\dot{\sigma}(v)\right]$$

$$\Theta_\infty^{(L+1)}(x,x') = \Theta_\infty^{(L)}(x,x')\dot{\Sigma}^{(L+1)}(x,x') + \Sigma^{(L+1)}(x,x').$$

Translation between GNN and GNTK

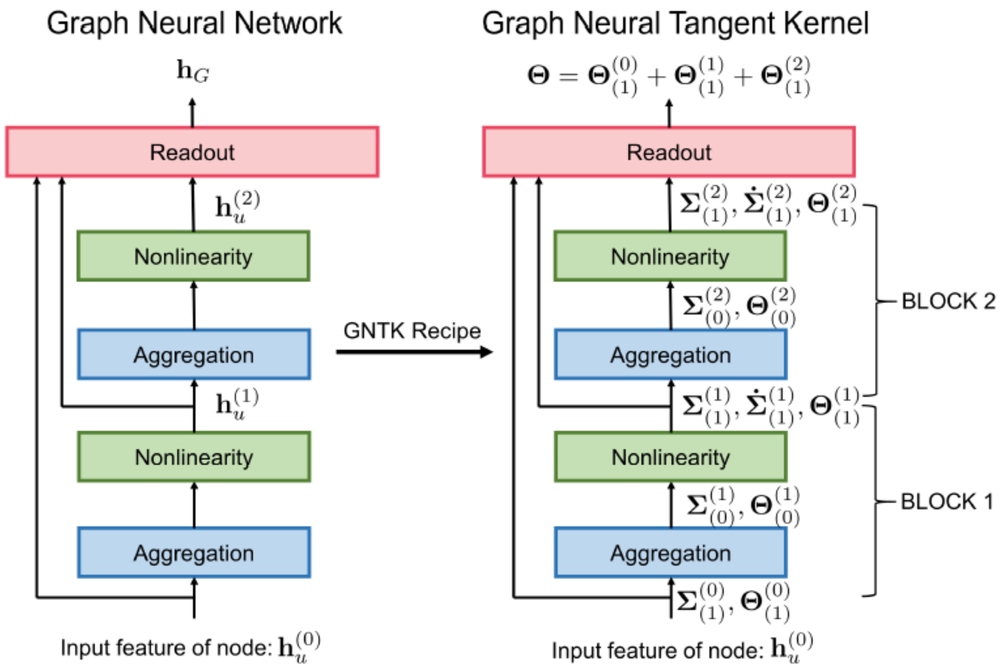


Figure 1: **Illustration of our recipe that translates a GNN to a GNTK.** For a GNN with $L = 2$ BLOCK operations, $R = 1$ fully-connected layer in each BLOCK operation, and jumping knowledge, the corresponding GNTK is calculated as follow. For two graphs G and G', we first calculate $\left[\boldsymbol{\Theta}^{(0)}_{(1)}(G,G')\right]_{uu'} = \left[\boldsymbol{\Sigma}^{(0)}_{(1)}(G,G')\right]_{uu'} = \left[\boldsymbol{\Sigma}^{(0)}(G,G')\right]_{uu'} = \boldsymbol{h}_u^\top \boldsymbol{h}_{u'}$. We follow the kernel formulas in Section 3 to calculate $\boldsymbol{\Sigma}^{(\ell)}_{(0)}, \boldsymbol{\Theta}^{(\ell)}_{(0)}$ using $\boldsymbol{\Sigma}^{(\ell-1)}_{(R)}, \boldsymbol{\Theta}^{(\ell-1)}_{(R)}$ (Aggregation) and calculate $\boldsymbol{\Sigma}^{(\ell)}_{(r)}, \dot{\boldsymbol{\Sigma}}^{(\ell)}_{(r)}, \boldsymbol{\Theta}^{(\ell)}_{(r)}$ using $\boldsymbol{\Sigma}^{(\ell)}_{(r-1)}, \boldsymbol{\Theta}^{(\ell)}_{(r-1)}$ (Nonlinearity). The final output is $\Theta(G,G') = \sum_{u\in V, u'\in V'} \left[\sum_{\ell=0}^{L} \boldsymbol{\Theta}^{(\ell)}_{(R)}(G,G')\right]_{uu'}$.

Theoretical Analysis

- Considering standard supervised learning setup,
  - n training data $\{(G_i, y_i)\}_{i=1}^{n}$ drawn iid from an underlying distribution $\mathcal{D}$
  - a simple GNN with a single BLOCK operation and a READOUT operation (without jumping knowledge)
  - scaling factor $c_u = \left(\left\|\sum_{v\in\mathcal{N}(u)\cup\{u\}} \boldsymbol{h}_v\right\|_2\right)^{-1}$
  - kernel matrix $\boldsymbol{\Theta}$ (assumed to be invertible)
- For a graph G (i.e., a test point), the prediction of kernel regression with GNTK is

$$f_{ker}(G) = [\Theta(G, G_1) \dots \Theta(G, G_n)]^T \boldsymbol{\Theta}^{-1} \boldsymbol{y}$$

Theoretical Analysis

- Building upon [Bartlett and Mendelson, 2002],

Theorem 4.1 (Bartlett and Mendelson [2002]). *Given n training data $\{(G_i, y_i)\}_{i=1}^n$ drawn i.i.d. from the underlying distribution $\mathcal{D}$. Consider any loss function $\ell : \mathbb{R} \times \mathbb{R} \to [0,1]$ that is 1-Lipschitz in the first argument such that $\ell(y, y) = 0$. With probability at least $1 - \delta$, the population loss of the GNTK predictor can be upper bounded by*

$$L_{\mathcal{D}}\left(f_{ker}\right) = \mathbb{E}_{(G,y)\sim\mathcal{D}}\left[\ell(f_{ker}(G), y)\right] = O\left(\frac{\sqrt{\boldsymbol{y}^\top \boldsymbol{\Theta}^{-1}\boldsymbol{y} \cdot \operatorname{tr}\left(\boldsymbol{\Theta}\right)}}{n} + \sqrt{\frac{\log(1/\delta)}{n}}\right).$$

- Remarks
  - data-dependent population risk bound (related to $\boldsymbol{\Theta}$ and $\boldsymbol{y}$) derived using Rademacher complexity
  - if $\boldsymbol{y}^T\boldsymbol{\Theta}^{-1}\boldsymbol{y}$ and $\operatorname{tr}(\boldsymbol{\Theta})$ can be bounded, GNTK that corresponds to this GNN can learn the target function with polynomial number of samples

Theoretical Analysis

- Bounding $\boldsymbol{y}^T \boldsymbol{\Theta}^{-1} \boldsymbol{y}$

Theorem 4.2. *For each $i \in [n]$, if the labels $\{y_i\}_{i=1}^n$ satisfy*

$$y_i = \alpha_1 \sum_{u \in V} \left(\overline{\boldsymbol{h}}_u^\top \beta_1 \right) + \sum_{l=1}^{\infty} \alpha_{2l} \sum_{u \in V} \left(\overline{\boldsymbol{h}}_u^\top \beta_{2l} \right)^{2l}, \tag{3}$$

where $\overline{\boldsymbol{h}}_u = c_u \sum_{v \in \mathcal{N}(u) \cup \{u\}} \boldsymbol{h}_v$, $\alpha_1, \alpha_2, \alpha_4, \ldots \in \mathbb{R}$, $\beta_1, \beta_2, \beta_4, \ldots \in \mathbb{R}^d$, and $G_i = (V, E)$, then we have

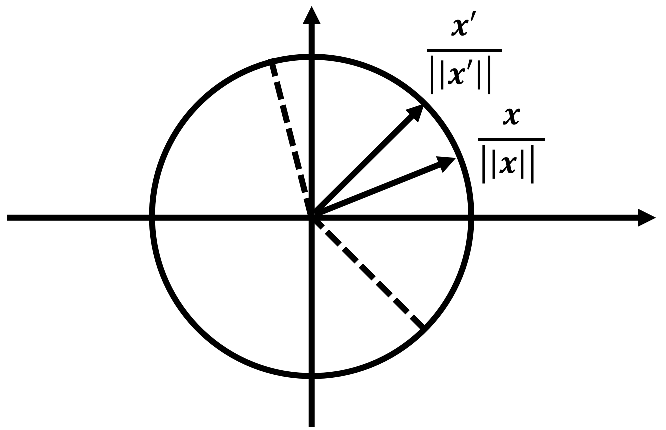
$$\sqrt{\boldsymbol{y}^\top \boldsymbol{\Theta}^{-1} \boldsymbol{y}} \leq 2|\alpha_1| \cdot \|\beta_1\|_2 + \sum_{l=1}^{\infty} \sqrt{2\pi}(2l-1)|\alpha_{2l}| \cdot \|\beta_{2l}\|_2^{2l}.$$

- Proof sketch
  - By definition, if we use ReLU activation, we have

$$\Theta(G, G') = \sum_{u \in V, u' \in V'} \left(\left[\Sigma_{(0)}^{(1)}(G, G') \right]_{uu'} \left[\dot{\Sigma}_{(1)}^{(1)}(G, G') \right]_{uu'} + \left[\Sigma_{(1)}^{(1)}(G, G') \right]_{uu'} \right)$$

$$\left[\dot{\Sigma}_{(1)}^{(1)}(G, G') \right]_{uu'} = \frac{\pi - \arccos\left(\left[\Sigma_{(0)}^{(1)}(G, G') \right]_{uu'} \right)}{2\pi},$$

$$\left[\Sigma_{(1)}^{(1)}(G, G') \right]_{uu'} = \frac{\pi - \arccos\left(\left[\Sigma_{(0)}^{(1)}(G, G') \right]_{uu'} \right) + \sqrt{1 - \left[\Sigma_{(0)}^{(1)}(G, G') \right]_{uu'}^2}}{2\pi}.$$



  - Let $\Theta = \Theta_1 + \Theta_2$ and $\Theta_1(G, G') = \sum_{u \in V, u' \in V'} \left[\Sigma_{(0)}^{(1)}(G, G') \right]_{uu'} \left[\dot{\Sigma}_{(1)}^{(1)}(G, G') \right]_{uu'}$
  - Since $\arcsin(x) = \sum_{l=0}^{\infty} \frac{(2l-1)!!}{(2l)!!} \cdot \frac{x^{2l+1}}{2l+1}$, we can bound $\boldsymbol{\Theta}_1$

Theoretical Analysis

- Proof sketch (cont'd)
  - Since $\mathbf{\Theta_2}$ in $\mathbf{\Theta} = \mathbf{\Theta_1} + \mathbf{\Theta_2}$ is PSD (because it is a kernel matrix), we have
$$\boldsymbol{y}^\top \boldsymbol{\Theta}^{-1} \boldsymbol{y} \leq \boldsymbol{y}^\top \boldsymbol{\Theta}_1^{-1} \boldsymbol{y}.$$
  - If we rewrite $y_i = \alpha_1 \sum_{u \in V} \left(\overline{h}_u^\top \beta_1\right) + \sum_{l=1}^{\infty} \alpha_{2l} \sum_{u \in V} \left(\overline{h}_u^\top \beta_{2l}\right)^{2l} = y_i^{(0)} + \sum_{l=1}^{\infty} y_i^{(2l)}$, we can get $\boldsymbol{y} = \boldsymbol{y}^{(0)} + \sum_{l=1}^{\infty} \boldsymbol{y}^{(2l)}$
  - Let $\Phi^{(2l)}(\cdot)$ be the feature map of the polynomial kernel of degree $2l$, we have
$$\sqrt{\boldsymbol{y}^\top \boldsymbol{\Theta}^{-1} \boldsymbol{y}} \leq \sqrt{\boldsymbol{y}^\top \boldsymbol{\Theta}_1^{-1} \boldsymbol{y}} \leq \sqrt{\left(\boldsymbol{y}^{(0)}\right)^\top \boldsymbol{\Theta}_1^{-1} \boldsymbol{y}^{(0)}} + \sum_{l=1}^{\infty} \sqrt{\left(\boldsymbol{y}^{(2l)}\right)^\top \boldsymbol{\Theta}_1^{-1} \boldsymbol{y}^{(2l)}}.$$

When $l = 0$, we have
$$\sqrt{\left(\boldsymbol{y}^0\right)^\top \boldsymbol{\Theta}_1^{-1} \boldsymbol{y}^0} \leq 2|\alpha_1| \|\beta_1\|_2.$$
When $l \geq 1$, we have
$$\sqrt{\left(\boldsymbol{y}^{2l}\right)^\top \boldsymbol{\Theta}_1^{-1} \boldsymbol{y}^{2l}} \leq \sqrt{2\pi}(2l-1)|\alpha_{2l}| \left\|\boldsymbol{\Phi}^{2l}\left(\beta_{2l}\right)\right\|_2.$$
Notice that
$$\left\|\boldsymbol{\Phi}^{2l}\left(\beta_{2l}\right)\right\|_2^2 = \left(\boldsymbol{\Phi}^{2l}\left(\beta_{2l}\right)\right)^\top \boldsymbol{\Phi}^{2l}\left(\beta_{2l}\right) = \|\beta_{2l}\|_2^{4l}.$$
Thus,
$$\sqrt{\boldsymbol{y}^\top \boldsymbol{\Theta}^{-1} \boldsymbol{y}} \leq 2|\alpha_1| \|\beta_1\|_2 + \sum^{\infty} \sqrt{2\pi}(2l-1)|\alpha_{2l}| \|\beta_{2l}\|_2^{2l}.$$

Theoretical Analysis

- Bounding $\mathrm{tr}(\boldsymbol{\Theta})$

Theorem 4.3. *If for all graphs $G_i = (V_i, E_i)$ in the training set, $|V_i|$ is upper bounded by $\overline{V}$, then $\mathrm{tr}(\boldsymbol{\Theta}) \leq O(n\overline{V}^2)$. Here n is the number of training samples.*

- Proof sketch
  - recall $\Theta(G,G') = \sum_{u\in V, u'\in V'} \left(\left[\boldsymbol{\Sigma}_{(0)}^{(1)}(G,G')\right]_{uu'} \left[\dot{\boldsymbol{\Sigma}}_{(1)}^{(1)}(G,G')\right]_{uu'} + \left[\boldsymbol{\Sigma}_{(1)}^{(1)}(G,G')\right]_{uu'}\right)$ where

$$\left[\boldsymbol{\Sigma}_{(0)}^{(1)}(G,G')\right]_{uu'} = c_u c_{u'} \left(\sum_{v\in\mathcal{N}(u)\cup\{u\}} \boldsymbol{h}_v\right)^\top \left(\sum_{v'\in\mathcal{N}(u')\cup\{u'\}} \boldsymbol{h}_{v'}\right) = \overline{\boldsymbol{h}}_u^\top \overline{\boldsymbol{h}}_{u'}$$

$$\left[\dot{\boldsymbol{\Sigma}}_{(1)}^{(1)}(G,G')\right]_{uu'} = \frac{\pi - \arccos\left(\left[\boldsymbol{\Sigma}_{(0)}^{(1)}(G,G')\right]_{uu'}\right)}{2\pi},$$

$$\left[\boldsymbol{\Sigma}_{(1)}^{(1)}(G,G')\right]_{uu'} = \frac{\pi - \arccos\left(\left[\boldsymbol{\Sigma}_{(0)}^{(1)}(G,G')\right]_{uu'}\right) + \sqrt{1 - \left[\boldsymbol{\Sigma}_{(0)}^{(1)}(G,G')\right]_{uu'}^2}}{2\pi}$$

  - by definition, we have $\left\|\overline{h_u}\right\|_2 = 1$
  - we can easily find out that $\left[\Sigma_{(0)}^{(1)}(G,G')\right]_{uu'} \leq 1$, $\left[\Sigma_{(1)}^{(1)}(G,G')\right]_{uu'} \leq 1$ and $\left[\dot{\Sigma}_{(1)}^{(1)}(G,G')\right]_{uu'} \leq 1/2$
  - then $\Theta(G,G') \leq 2|V||V'|$, which implies $\mathrm{tr}(\boldsymbol{\Theta}) \leq 2n\bar{V}^2$

Outline

- Introduction
- Related Work
- Technical Details
  - Background: Infinite-Width Networks and Neural Tangent Kernel
  - Graph Neural Tangent Kernel (GNTK)
- **Experimental Results**

Experimental Setting

- Datasets
  - bioinformatics: MUTAG, PTC, NCI1, PROTEINS
  - social network: COLLAB, IMDB-BINARY, IMDB-MULTI
- Pre-processing
  - categorical features -> one-hot encoding
  - use degree as input feature if no available node feature (i.e., topology only)

Effectiveness Results

- Task: graph classification

	Method	COLLAB	IMDB-B	IMDB-M	PTC	NCI1	MUTAG	PROTEINS
GNN	GCN	79.0 ± 1.8	74.0 ± 3.4	51.9 ± 3.8	64.2 ± 4.3	80.2 ± 2.0	85.6 ± 5.8	76.0 ± 3.2
	GraphSAGE	–	72.3 ± 5.3	50.9 ± 2.2	63.9 ± 7.7	77.7 ± 1.5	85.1 ± 7.6	75.9 ± 3.2
	PatchySAN	72.6 ± 2.2	71.0 ± 2.2	45.2 ± 2.8	60.0 ± 4.8	78.6 ± 1.9	**92.6 ± 4.2**	75.9 ± 2.8
	DGCNN	73.7	70.0	47.8	58.6	74.4	85.8	75.5
	GIN	80.2 ± 1.9	75.1 ± 5.1	52.3 ± 2.8	64.6 ± 7.0	82.7 ± 1.7	89.4 ± 5.6	**76.2 ± 2.8**
GK	WL subtree	78.9 ± 1.9	73.8 ± 3.9	50.9 ± 3.8	59.9 ± 4.3	**86.0 ± 1.8**	90.4 ± 5.7	75.0 ± 3.1
	AWL	73.9 ± 1.9	74.5 ± 5.9	51.5 ± 3.6	–	–	87.9 ± 9.8	–
	RetGK	81.0 ± 0.3	71.9 ± 1.0	47.7 ± 0.3	62.5 ± 1.6	84.5 ± 0.2	90.3 ± 1.1	75.8 ± 0.6
	GNTK	**83.6 ± 1.0**	**76.9 ± 3.6**	**52.8 ± 4.6**	**67.9 ± 6.9**	84.2 ± 1.5	90.0 ± 8.5	75.6 ± 4.2

Table 1: **Classification results (in %) for graph classification datasets.** GNN: graph neural network based methods. GK: graph kernel based methods. GNTK: fusion of GNN and GK.

Efficiency Results

- Training on IMDB-B datasets with TITAN X GPU,
  - GIN takes 19 mins
  - GNTK takes 2 mins
- Not much details mentioned in the paper

Ablation Study

- Effects of # BLOCK operations and the scaling factor c_u

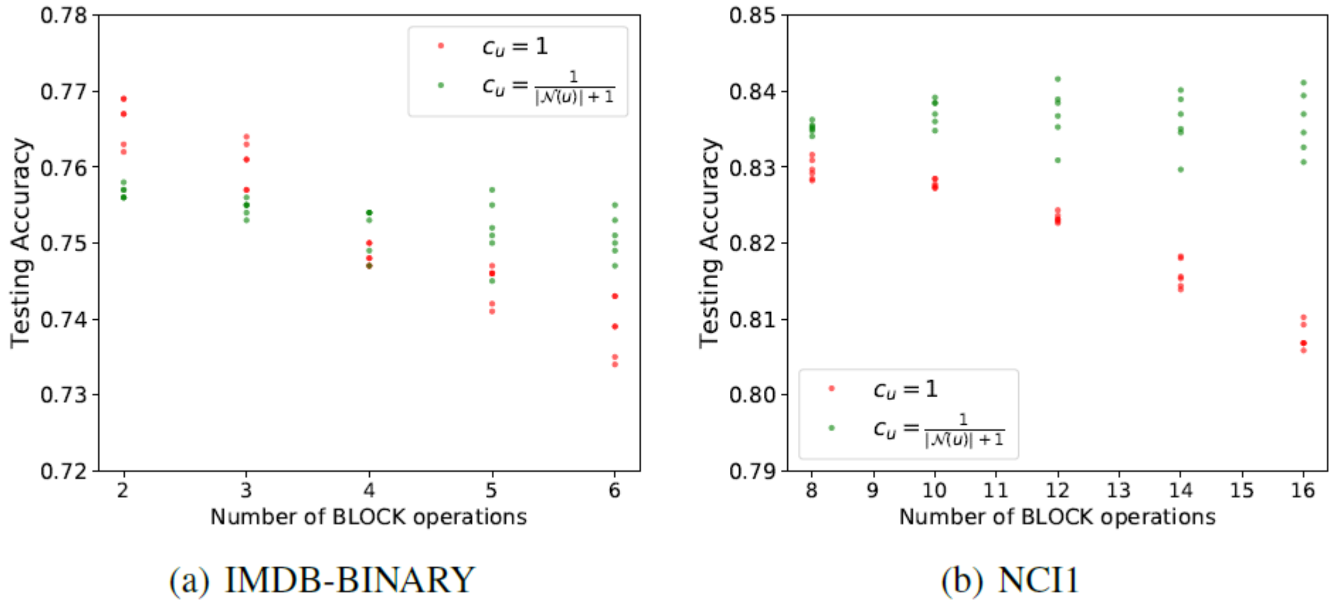
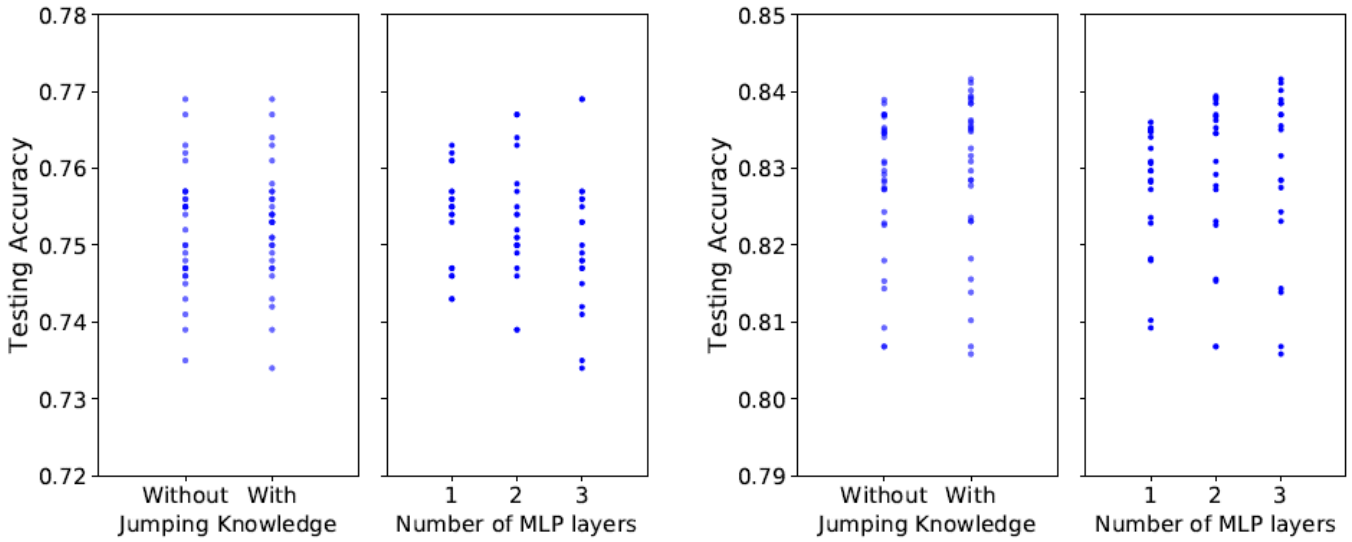


Figure 2: **Effects of number of BLOCK operations and the scaling factor c_u on the performance of GNTK.** Each dot represents the performance of a particular GNTK architecture. We divide different architectures into different groups by number of BLOCK operations. We color these GNTK architecture points by the scaling factor c_u. We observe the test accuracy is correlated with the dataset and the architecture.

Ablation Study

- Effects of jumping knowledge and # MLP layers



(a) IMDBBINARY (b) NCI1

Figure 3: **Effects of jumping knowledge and number of MLP layers on the performance of GNTK.** Each dot represents the test performance of a GNTK architecture. We divide all GNTK architectures into different groups, according to whether jumping knowledge is applied, or number of MLP layers.

References

- Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., & Wang, R. (2019). On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*.
- Bartlett, P. L., & Mendelson, S. (2002). Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*.
- Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*.
- Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., & Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*.
- Neal, R. M. (1996). Priors for infinite networks. In *Bayesian Learning for Neural Networks*. Springer, New York, NY.
- Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., ... & Sohl-Dickstein, J. (2018). Bayesian deep convolutional networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*.