A Guide to Probably Approximately Correct Bounds for ~~Neural Networks~~ Graph Neural Networks

Thomas Walker

Summer 2023

Contents

1 Introduction

The geometric deep learning blueprint [4] has amassed a lot of success as a method for structuring machine learning techniques. It has enabled the innovation and investigation of machine learning algorithms as well as supporting the rise of graph neural networks (GNNs) as a tool for solving problems in a variety of different domains. Therefore, it is of interest to understand how the PAC learning framework can be utilised in this new setting. The investigation of this has started with [3] who derive PAC bounds for graph convolutional networks (GCNs) and message-passing graph neural networks (MPGNNs). Here we will detail the results given by [3] and understand how to proceed in developing PAC bounds for GNNs. What we will see is that the results of [3] are indeed a generalization of the results obtained for neural networks. This is to be expected as GNNs can also be seen as a generalisation of neural networks.

2 Preliminaries

2.1 Problem Setup

Throughout, the K-class graph classification problem is considered. Each sample, $z = (A, X, y)$ for the problem is a triplet.

1. A, the graph adjacency matrix.
2. $X \in \mathbb{R}^{n \times h_0}$, the node feature matrix.
    - n is the number of nodes.

- h_0 is the input feature dimension.

3. $y \in \mathbb{R}^{1\times K}$, the label.

We will adopt the following notation.

1. $\mathbb{N}_k^+$ will denote the first k positive integers.
2. $|\cdot|_p$ will note the vector p-norm.
3. $\|\cdot\|_p$ will denote the operator norm induced by the p-norm.
4. $\|\cdot\|_F$ will denote the matrix Frobenius norm.

We set up the K-class graph classification problem in the following way.

- We suppose we have a sample space $\mathcal{Z}$.
- A node feature matrix $X \in \mathcal{X}$ comes from the node feature space.
- A adjacency matrix $A \in \mathcal{G}$ comes from the graph space.
- We suppose a distribution $\mathcal{D}$ is defined on $\mathcal{Z}$ so that $z \overset{\text{i.i.d}}{\sim} \mathcal{D}$.
- A model $f_{\mathbf{w}} \in \mathcal{H}$ comes from a hypothesis class.
- The training set is $S = \{z_1, \ldots, z_m\}$.
- The following assumptions are made to derive our bounds.
    1. Data consists of i.i.d samples from the unknown distribution $\mathcal{D}$.
    2. The maximum hidden dimension across all layers is h.
    3. For a node feature matrix X, and for all $i \in \mathbb{N}_n^+$ we have

$$X[i,:] \in \mathcal{X}_B = \left\{ x \in \mathbb{R}^{h_0} : \sum_{j=1}^{h_0} x_j^2 \leq B^2 \right\}.$$

    4. We only consider simple graphs with the maximum node degree being at most $d-1$.

2.2 Graph Preliminaries

2.2.1 GCNs

Graph convolutional neural networks for the K-class graph classification problem are defined as follows.

- k^{th} Graph Convolution Layer, $H_k = \sigma_k\left(\tilde{L}H_{k-1}W_k\right)$.
    - $k \in \mathbb{N}_{l-1}^+$.
    - $H_k \in \mathbb{R}^{n\times h_k}$ are node representations, with $H_0 = X$.
    - $\tilde{L} = D^{-\frac{1}{2}}\tilde{A}D^{\frac{1}{2}}$ with $\tilde{A} = I + A$ is the graph Laplacian.
    - $\sigma_k = \max(0, x)$ is the ReLU non-linearity.
- Readout Layer, $H_l = \frac{1}{n}\mathbf{1}_n H_{l-1}W_l$.
    - $\mathbf{1}_n \in \mathbb{R}^{1\times n}$ is a vector of all ones.
    - l is the number of layers.
    - W_j is the j^{th} layer weight matrix.

2.2.2 MPGNNs

The message-passing GNNs we consider are the following.

- k^{th} step Message Computation, $M_k = g\left(C_{\text{out}}^{\top} H_{k-1}\right)$.
  - $k \in \mathbb{N}_{l-1}^{+}$.
  - $H_k \in \mathbb{R}^{n \times h_k}$ are node representations, with $H_0 = \mathbf{0}$.
  - $C_{\text{out}} \in \mathbb{R}^{n \times c}$, where c is the number of edges. $C_{\text{out}}[i, j] = 1$ indicates that the outgoing node of the j^{th} edge is the i^{th} node.
  - $g : \mathbb{R}^h \to \mathbb{R}^h$ is a non-linear mapping with Lipschitz constant C_g, and $g(\mathbf{0}) = \mathbf{0}$.
- k^{th} step Message Aggregation, $\bar{M}_k = C_{\text{in}} M_k$.
  - $k \in \mathbb{N}_{l-1}^{+}$.
  - $C_{\text{in}} \in \mathbb{R}^{n \times c}$, where $C_{\text{in}}[i, j] = 1$ indicates the the incoming node of the j^{th} edge is the i^{th} node.
- k^{th} step Node State Update, $H_k = \phi\left(XW_1 + \rho\left(\bar{M}_k\right) W_2\right)$.
  - $k \in \mathbb{N}_{l-1}^{+}$.
  - W_j is the j^{th} layer weight matrix.
  - $\phi : \mathbb{R}^h \to \mathbb{R}^h$ is a non-linear mapping with Lipschitz constant C_ϕ, and $\phi(\mathbf{0}) = \mathbf{0}$.
  - $\rho : \mathbb{R}^h \to \mathbb{R}^h$ is a non-linear mapping with Lipschitz constant C_ρ, and $\rho(\mathbf{0}) = \mathbf{0}$.
- Readout Layer, $H_l = \frac{1}{n} \mathbf{1}_n H_{l-1} W_l$.
  - $\mathbf{1}_n \in \mathbb{R}^{1 \times n}$ is a vector of all ones.
  - l is the number of layers.

Each of the non-linear mappings can be generalised to maps defined over matrix states.

- $\tilde{g} : \mathbb{R}^{n \times h} \to \mathbb{R}^{n \times h}$.
- $\tilde{\phi} : \mathbb{R}^{n \times h} \to \mathbb{R}^{n \times h}$.
- $\tilde{\rho} : \mathbb{R}^{n \times h} \to \mathbb{R}^{n \times h}$.

Definition 2.1. *The percolation complexity is given by*

$$\mathcal{C} = C_g C_\phi C_\rho \|W_2\|_2.$$

2.3 Loss Function

The multi-class γ-margin loss function is used to define the generalisation error for which the bounds are constructed. Specifically, for $\gamma > 0$ and $f_{\mathbf{w}}(X, A) = H_l$, the generalisation error is given by

$$L_\gamma(f_{\mathbf{w}}) = \mathbb{P}_{z \sim \mathcal{D}}\left(f_{\mathbf{w}}(X, A)[y] \leq \gamma + \max_{j \neq y}\left(f_{\mathbf{w}}(X, A)[j]\right)\right).$$

Similarly, the empirical generalisation error is given by

$$\hat{L}_\gamma(f_{\mathbf{w}}) = \frac{1}{m}\left|\left\{x_i \in S : f_{\mathbf{w}}(X, A)[y] \leq \gamma + \max_{j \neq y}\left(f_{\mathbf{w}}(X, A)[j]\right)\right)\right\}\right|.$$

2.4 PAC-Bayes

The PAC-Bayes used in this work comes from [1].

Theorem 2.2. *Let $f_{\mathbf{w}} : \mathcal{X} \to \mathbb{R}^K$ be a model with parameters $\mathbf{w}$. Let P be a distribution over the parameters that is independent of the training set $S = \{z_i\}_{i=1}^m$, which is an* i.i.d *sample from a distribution $\mathcal{D}$. For any $\mathbf{w}$, construct a posterior $Q(\mathbf{w}+\mathbf{u})$ where $\mathbf{u}$ is a random perturbation vector such that*

$$\mathbb{P}\left(\max_{x\in\mathcal{X}} |f_{\mathbf{w}+\mathbf{u}}(x) - f_{\mathbf{w}}(x)|_\infty < \frac{\gamma}{4}\right) > \frac{1}{2}.$$

Then for any $\gamma, \delta > 0$, we have that

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left(L_0(f_{\mathbf{w}}) \leq \hat{L}_\gamma(f_{\mathbf{w}}) + \sqrt{\frac{2\mathrm{KL}(Q(\mathbf{w}+\mathbf{u}), P) + \log\left(\frac{8m}{\delta}\right)}{2(m-1)}}\right).$$

3 Generalization Bounds

Theorem 3.1 (GCN Generalization Bounds)**.** *For any $B > 0$, $l > 1$, let $f_{\mathbf{w}} \in \mathcal{H} : \mathcal{X} \times \} \to \mathbb{R}^K$ be a l layer GCN. Then for any $\delta, \gamma > 0$, with probability at least $1-\delta$ over an* i.i.d *sampled training set of size m over $\mathcal{D}$ we have that*

$$L_0(f_{\mathbf{w}}) \leq \hat{S}_\gamma + 0\left(\sqrt{\frac{B^2 d^{l-1} l^2 h \log(lh) \prod_{i=1}^l \|W_i\|_2^2 \sum_{i=1}^l \left(\frac{\|W_i\|_F^2}{\|W_i\|_2^2}\right) + \log\left(\frac{ml}{\delta}\right)}{\gamma^2 m}}\right).$$

Theorem 3.2 (MPGNN Generalisation Bound)**.** *For any $B > 0$, $l > 1$, let $f_{\mathbf{w}} \in \mathcal{H} : \mathcal{X} \times \} \to \mathbb{R}^K$ be a l step MPGCN. Then for any $\delta, \gamma > 0$, with probability at least $1-\delta$ over an* i.i.d *sampled training set of size m over $\mathcal{D}$ we have that*

$$L_0(f_{\mathbf{w}}) \leq \hat{S}_\gamma + 0\left(\sqrt{\frac{B^2\left(\max\left(\zeta^{-(l+1)}, (\lambda\xi)^{\frac{l+1}{l}}\right)\right)^2 l^2 h \log(lh) |\mathbf{w}|_2^2 + \log\left(\frac{m(l+1)}{\delta}\right)}{\gamma^2 m}}\right),$$

where

- $\zeta = \min\left(\|W_1\|_2, \|W_2\|_2, \|W_l\|_2\right)$,
- $|\mathbf{w}|_2^2 = \|W_1\|_F^2 + \|W_2\|_F^2 + \|_l\|_F^2$,
- $\lambda = \|W_1\|_2 \|W_l\|_2$, *and*
- $\xi = C_\phi \frac{(d\mathcal{C})^{l-1}-1}{d\mathcal{C}-1}$.

3.1 Connection to MLPs/CNNs

We note that MLPs/CNNs are a special case of GNNs.

- We can treat each i.i.d sample as a single-node graph, such that conventional tasks become graph-level tasks. Such that $d = 1$ and $\tilde{L} = I$.

Using this view we can restate the content of Theorem 3.1 with ReLU actions as,

$$L_0(f_{\mathbf{w}}) \leq \hat{L}_\gamma + O\left(\sqrt{\frac{B^2 l^2 h \log(lh) \prod_{i=1}^l \|W_i\|_2^2 \sum_{i=1}^l \left(\frac{\|W_i\|_F^2}{\|W_i\|_2^2}\right) + \log\left(\frac{ml}{\delta}\right)}{\gamma^2 m}}\right)$$

which coincides with the results derived in [1].

4 Conclusion

The above analysis works for other bounded loss functions.

The limitations of this work include the following.

1. Bounds are vacuous in practice.
2. Gaussian posteriors are assumed to obtain analytic forms for the KL-divergence. However, the posteriors of a learning process are likely to be non-Gaussian.
3. The analysis done is agnostic to the optimization algorithm.

Future directions of work include the following.

1. What other statistics, besides maximum node degree, have an impact on the generalisation of GNNs?
2. Can this analysis be done over other GNN architectures?
3. Can the compression techniques of [2] be transferred to the GNN framework?
4. What is the impact of the optimization algorithms like SGD on the generalisation ability of GNNs?
5. Would graph structures play a role in the analysis of optimization?

References

[1] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. “A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks”. In: *CoRR* (2017).

[2] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. “Stronger generalization bounds for deep nets via a compression approach”. In: *CoRR* (2018).

[3] Renjie Liao, Raquel Urtasun, and Richard S. Zemel. “A PAC-Bayesian Approach to Generalization Bounds for Graph Neural Networks”. In: *CoRR* (2020).

[4] Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: *CoRR* (2021).