Generalization bounds and size generalization for graph neural networks

by

Emmanuel Sales

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Generalization bounds and size generalization for graph neural networks

submitted by **Emmanuel Sales** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Nicholas Harvey, Professor, Computer Science, UBC
Supervisor

Renjie Liao, Professor, Electrical and Computer Engineering, UBC
Supervisory Committee Member

Abstract

Graph neural networks (GNNs) are a class of machine learning models that relax the independent and identically distributed (i.i.d.) assumption between data points that underlies most machine learning models.

Theoretical understanding of these models involves analyzing generalization bounds, a theoretical framework for finding the provable discrepancies between expected train and test loss. We make advancements in state-of-the-art PAC-Bayes generalization bounds for GNNs using insights from graph theory and random matrix theory, and perform experiments for validation.

One of the most important directions in the study of modern theoretical machine learning is the analysis of out-of-distribution error; that is, error measured particularly on examples from a distinct distribution from the training distribution. In particular for the graph learning setting and GNNs, there are importatnt questions that can be explored about size generalization, the capacity for a graph neural network to make predictions on graphs much larger than seen on its training set.

We develop a theoretical framework for size generalization with the analysis of graph learning settings where GNNs can easily perform size generalization, and develop probabilistic theorems analyzing some measures of generalization error, building off of the work done in the PAC-Bayes analysis.

Lay Summary

Graph neural networks (GNNs) are machine learning models that take into account the relationships and networks between different data points. They have seen much use in applications such as social network analysis and molecular science, since their introduction and throughout the proliferation of deep learning.

As with many modern machine learning models, practice has outpaced theory, with models being shown to be usable in new innovations before the research community is able to analyze them theoretically, that is, in a completely provable fashion. In this work, we review the state of theoretical understanding of graph neural networks, and make some improvements to the theorems currently at the state of the art. We then introduce the beginnings of a new theoretical framework to understand a key question of interest regarding the models: if a GNN is trained on small networks, what can they say about larger ones?

Preface

This thesis is an original, unpublished work done in collaboration with Dr. Nicholas Harvey and Dr. Renjie Liao.

The original results of this work are present in Chapters 9, 10, and 11. The main body of the work in Chapter 9 were developed in collaboration with Dr. Harvey and written by myself, with input from Dr. Liao, whose work we are extending. The experiments described in Chapter 10 were designed and performed by myself, with Dr. Liao providing support for the computation of the results of the experiments by providing the code used in [1] which was modified by me for the purpose of my own experiments. Dr. Liao also provisioned the computational resources needed to run the associated code. The proofs in Chapter 11 were developed and written by myself in collaboration with Dr. Harvey.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to extend my deepest gratitude to my supervisor, Dr. Nick Harvey, whose patient guidance has shaped my academic journey for close to half a decade. Nick – the principles that I have learned from your mentorship will be invaluable to me for the rest of my life.

I would like to thank Dr. Renjie Liao, whose research was the starting point for the original contributions in this work, and who has been an exceptional and insightful collaborator throughout the process since his new arrival at UBC in January.

I would like to thank Yueh-Hua Tu, maintainer of the graph neural network Julia library `GeometricFlux.jl`, for helping me make contributions to it. With his guidance I was able to solidify my understanding of GNNs at a level that was immensely helpful for this work.

To my parents, Rafaela and Aureo, I say once again that your unconditional love and support is beyond that which I can repay. The amount of sacrifice and hard work that you have had to perform, during a global pandemic and throughout all our life, is something that still falls outside the realm of my comprehension. *Mahal ko kayo, ngayon at magpakailanman.*

I would like to thank my Tita Lydia and cousin Marissa for their love and support for me as well as an important source of inspiration. The perseverance in life you have shown throughout these years has always been a steady light for me.

I would like to thank my fellow students, of all kinds, in the Algorithms and Machine Learning labs, from whom I have so enjoyed the collegial environment they have fostered, and whose work has served as constant inspiration. I am in your debt, and I wish you all the best in your own journeys in research.

In addition to those that were immediately adjacent to my work, I must also acknowledge that all creative pursuits are reflections of all those around the creator. A collection of others have been instrumental in allowing me to complete this particular pursuit, with each of their unique ways of care, camaraderie, and lived example. To all the friends who have been with me throughout – though you may not yet realize it, you each had an instrumental role in allowing me to complete this work. Thank you immensely and I wish you all the best out of life.

Dedication

Dear Évariste,

I did not know you, nor have I delved deep into the details of your expansive work (abstract algebra isn't quite my forté), but because of technology I have been able to see a glimpse of your life; please pardon the intrusion.

My mind is nowhere near as beautiful as yours, but I understand what it must have felt like to conceive of a wonder of an idea, only to have it come out a completely different way. Like your mind is a storm, perceiving the things that lie beyond words, and that nobody will ever understand the depths of it. From what I can tell, I do not think anyone truly did. Not the ones who killed you, ending a life so young. Not the people who imprisoned you. Not Poisson, not even Cauchy.

With all the chaos you experienced in your life, Évariste, my genuine hope is that at some point, you might have experienced something like the beauty of the scene near the hill in Montmartre on a sunny day, next to Sacré-Coeur – the only thing I remember from my own brief time in Paris. For I think such experiences are all that we know are real, at the end of each day.

We extend our gratitude to you. May the world always know the name of Évariste Galois, and may your soul yet have many more names.

I dedicate this work to all those who embody the soul of our esteemed colleague Évariste – those who strive for knowledge and their ideas, and strive to be understood.

Chapter 1

The learning task

Machine learning is a field of study that draws from many different areas including computer science and statistics and may be viewed as the study of using algorithms to *infer* models from data, with a model in the most general sense being a function that yields some sort of insight about the underlying data used as input.

We will define this framework more formally using the notation of Shalev-Shwartz and Ben-David [2], which defines a *statistical learning framework* with the following mathematical constructs:

- The domain set, $\mathcal{X}$, which contains objects that we wish to examine.
- The label set, $\mathcal{Y}$. In Shalev-Shwartz and Ben-David [2] they define this as being restricted to a two-element set like $\{-1, 1\}$ or $\{0, 1\}$ but many machine learning tasks involve having the labeling space much larger than this, either by having a multitude of possible labels (as in image classification) or even a continuous space (as in regression tasks like predicting a quantity associated with a protein).
- Training data, which consists of pairs $(x_1, y_1), \ldots, (x_m, y_m)$ that each exist in $\mathcal{X} \times \mathcal{Y}$.
- Given the training data, a **learning algorithm** outputs a function, $h : \mathcal{X} \to \mathcal{Y}$.

- We assume that there is some "ground truth" labelling function $f : \mathcal{X} \to \mathcal{Y}$, such that $y_i = f(x_i)$ for all i, and that the function h generated by the learning algorithm is approximating f as closely as possible. Furthermore, we assume that the training instances x_i are generated independently and identically distributed under a probability distribution $\mathcal{D}$.

- A loss function, $\ell(h)$, which is a measure of how different the function outputted by the algorithm is from the true learning function. For classification functions like those that operate under the assumptions of SSBD this can be simply the probability under $\mathcal{D}$ that an example x has $f(x) \neq h(x)$. For continuous outputs another natural loss function is expected squared loss, $\ell(h) = \mathbb{E}_{\mathcal{D}}\left[(f(x) - h(x))^2\right]$.

This is a very powerful characterization of machine learning models, as it can apply to a plethora of model classes and learning tasks. However it must be noted that a few key assumptions underpin this entire framework. This does not admit certain models that are better suited to certain kinds of problems or data, particularly when the nature of the data violates the framework's assumptions. Of particular note is the IID assumption under which the x_i examples are generated. Many real-world problems do not follow this framework; often an observable object in the world would have predictable quantities which do not only depend on its own quantities but also on the quantities of other objects with which it has relations. For example, if you are trying to predict the position of an atom within a molecule after a certain time step, it is not enough to consider just the chemical properties and current position of the single atom; it is also necessary to examine those properties of the atom *as well as* the properties other atoms close to it, to account for their interaction.

Chapter 2

The learning task for graphs

2.1 Graph preliminaries

The mathematical construct of the graph is a universal model for pairwise relationships. A graph G is a set of *vertices* V and a set of *edges* $E \subseteq V \times V$, with two vertices v_1, v_2 said to be connected if $(v_1, v_2) \in E$. For a given graph $G \in \mathcal{G}$ we can also denote its vertices by $V(G)$ and edges $E(G)$.

Unless otherwise specified we assume graphs are undirected and without multi-edges. For the purposes of machine learning, a graph may have a set of node features $x_v : V \to \mathbb{R}^{d_v}$, and edge features, $x_e : E \to \mathbb{R}^{d_e}$.

Given graph-structured data, a natural task that arises from a relaxation the above framework is that of *node classification* (or regression). Formally, that means that we have the following constructs:

- Graph data, $\{G_i\}_{i=1}^m \in \mathcal{G}$, where $\mathcal{G}$ is the set of all graphs. For each i, $G_i = (V_i, E_i)$, the vertex and edge sets of graph i.

- Node feature data, $x : \mathcal{V} \to \mathcal{X}$, $\mathcal{X} \subseteq \mathbb{R}^d$ for all vertices $v \in V(G_i), \forall i \in [m]$. The graph structure also admits a neighbor function $\mathcal{N}(v) = \{u \in V(G_i) : (v, u) \in E(G_i)\}$.

- Node labels, $y : \mathcal{V} \to \mathcal{Y}$, with $\mathcal{Y}$ being the set of labels, either discrete or continuous

- A ground truth function f, which is defined in a different way from the above framework due to the graph structure; the definition would be a function f, which maps *graphs* to labels. Outputs h of the learning algorithm will be of the same type as f. It is possible that the function f is associated with an underlying *vertex labelling*, $f_V : V(G) \to \mathcal{Y}$, of which f would be an aggregation like $f(G) = \text{mode}_{v \in V(G)}(f_V(v))$.

- An equivalent formulation of f is in terms of its node features and adjacency matrix: $f : \mathcal{X}^n \times R^{n \times n} \to \mathcal{Y}^n$. Permutation invariance and equivariance may apply extra constraints; there is more discussion below.

- A loss function $\ell : \mathcal{G} \to \mathbb{R}$, for example defined in the classification case as $\ell(G) = \frac{1}{|V(G)|} \sum_{i=1}^{|V(G)|} \mathcal{I}[f(G)_i \neq h(G)_i]$.

This framework relaxes the IID assumption on the node data because the function does not need to take in the data for each vertex in isolation; instead, it takes in the whole graph. By doing this, for any given vertex the function may take into account the properties of its neighbours, its 2-neighbours, or even any selection of other nodes in the graph that may be useful.

Not all functions f that operate on that domain and range are necessarily permissible, however. If two graphs have exactly the same nodes but with the labels permuted, then the function should produce the same output. This property of functions f that do this is called *permutation invariance*. Formally, it is defined in terms of permutation matrices and the graphs' adjacency matrices.

- Two graphs G_1 and G_2 with adjacency matrices A_1 and A_2 are considered isomorphic to each other if there exists a permutation matrix P such that $A_2 = PA_1P^\top$. A permutation matrix $P \in \mathbb{R}^{n \times n}$ is defined such that every entry is 0 or 1, and there is exactly one 1 entry in each row and each column.

- A graph function in node feature-adjacency matrix form $f : \mathcal{X}^n \times R^{n \times n} \to \mathcal{Y}^n$ is considered permutation equivariant if $\forall P, X, A$, we have $f(PX, PAP^\top) = Pf(X, A)$.
- A loss function $\ell : \mathcal{X}^n \times R^{n \times n} \to \mathbb{R}$ is considered permutation invariant if $\forall P, X, A$, we have $\ell(PX, PAP^\top) = \ell(X, A)$.

Note that in the above framework, if both the ground-truth function f and the learned function h are permutation-equivariant, then the function ℓ is permutation-invariant.

Chapter 3

Neural Networks and Graph Neural Networks

Neural networks are perhaps the most popular and important machine learning models of the modern era; this is because of their exceptional performance in certain important tasks, such as image recognition [3], recommendation systems [4], and natural language understanding, translation, and generation [5], which have been put into use by many applications and websites that are used by billions of people.

The definition of a neural network is presented by [2] as a directed acyclic graph (DAG). Furthermore, the nodes in this graph are further subdivided into disjoint sets called *layers*, $V_1, \ldots, V_l$, such that any edge in this DAG connects a node in V_i to a node in V_{i+1} for some $i \in \{1, \ldots, l-1\}$. This DAG is then coupled with a weight function: $f : E \to \mathbb{R}$, which denotes weights applied to information as it is passed from one layer to another.

The first layer V_1 is the input to the neural network; the output of the neural network is $\mathbb{R}^{d_{\text{out}}}$, where d_{out} is the number of nodes in the final layer V_l. The output is computed by the as

$$a_{t+1,j}(\mathbf{x}) = \sum_{i=1}^{|V_t|} w(V_{t,i}, V_{t+1,j}) o_{t,i}(\mathbf{x})$$

and

$$o_{t+1,j}(\mathbf{x}) = \sigma(a_{t+1,j})$$

where σ is a fixed nonlinear function; some popular choices of σ have included the hyperbolic tangent (tanh) function, the sigmoid function $\sigma(z) = \frac{1}{1-e^{-z}}$, and the rectified linear unit, $\sigma(z) = \max\{0, z\}$.

Neural networks have been considered as essentially performing a feature transformation followed by a supervised learning objective, learning a representation of the underlying data unto which something resembling a regression model is applied. In that respect, ordinary feedforward neural networks make an implicit assumption that the underlying data is independent and identically distributed, and thus do not capture structural dependencies between individual data points.

3.1 Definition of graph neural networks

Graph neural networks are a generalization of the neural network to data with variable structures and dependencies on other data points, achieving this by adopting an architecture suited to the graph learning task described in Section 2.

Input data are graphs, $G \in \mathcal{G}$, and takes the form of node features and edge features, where a feature vector in $\mathbb{R}^d$ is associated with either a vertex or edge in the graph.

The message passing algorithm is defined in its most general form by the recurrence relation [6]

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}(\mathbf{h}_u^{(k)}, \text{AGGREGATE}(\{\mathbf{h}_v^{(k)} | v \in \mathcal{N}(u)\})) \tag{3.1}$$

It can be argued that the defining feature of each specific GNN variant depends on the choice of UPDATE and AGGREGATE functions. A very basic GNN can be derived by setting the AGGREGATE function to be a sum over all of the neighbors' node features and the UPDATE function to an application of self- and neighbour-weights followed by a nonlinear transformation σ (e.g. σ = ReLU). Formally this can be defined as

$$\text{AGGREGATE}(\mathcal{N}(u)) = \mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v \tag{3.2}$$

$$\mathbf{h}_u^{(i+1)} = \text{UPDATE}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \sigma\left(\mathbf{W}_{\text{self}}\mathbf{h}_u^{(i)} + \mathbf{W}_{\text{neighbor}}\mathbf{m}_{\mathcal{N}(u)}\right) \tag{3.3}$$

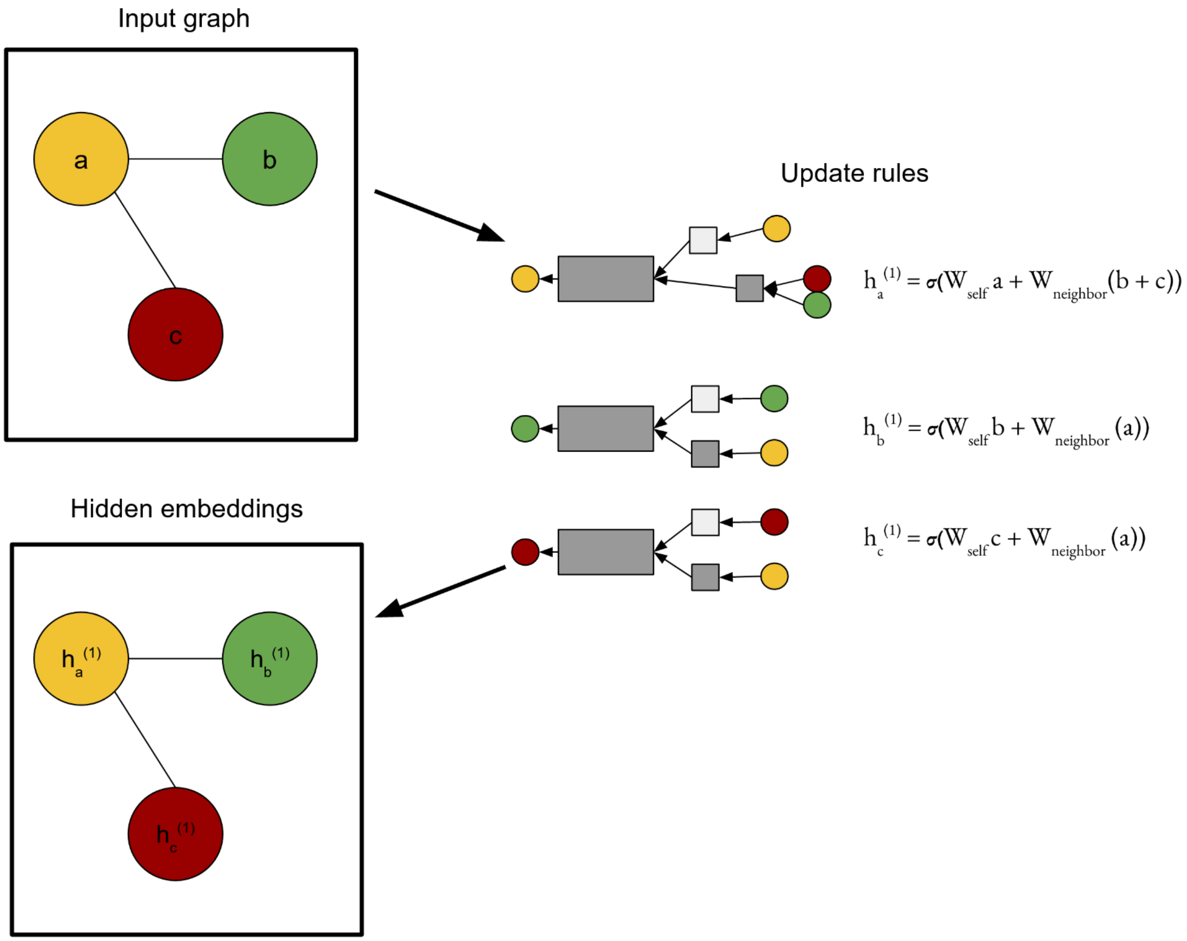
Input graph
a
b
c
Update rules
$h_a^{(1)} = \sigma(W_{self}\, a + W_{neighbor}(b + c))$
$h_b^{(1)} = \sigma(W_{self}\, b + W_{neighbor}\,(a))$
$h_c^{(1)} = \sigma(W_{self}\, c + W_{neighbor}\,(a))$
Hidden embeddings
$h_a^{(1)}$
$h_b^{(1)}$
$h_c^{(1)}$

Figure 3.1: An illustration of the neighbor updates in a GNN layer.

Chapter 4

Graph convolutional networks (GCNs)

4.1 Convolutional neural networks

Convolutional neural networks are a model that developed independently of GNNs and are the cornerstone of machine learning models that operate on image data, including the famous AlexNet [3], which was the first CNN-based model to win the ImageNet challenge, achieving human-like accuracy in image classification. (A CNN-based model has won the competition every time it has been held since that year.)

CNNs are a model that can capture the dependencies between spatially contiguous pixels of an image, which, like GNNs, relax the IID assumption by considering the dependencies between pixels and their neighbors. However, you cannot reduce a CNN to a GNN, at least not with our current definitions of a graph. By inputting image data into a GNN layer, each layer would work on a fixed graph structure, where there are N^2 nodes corresponding to pixels in an $N \times N$ resolution image, and each pixel would be connected via an edge to each of its spatial neighbors. This would lose the specific spatial relation between the pixel of interest and each of its neighbors (i.e. you would lose the information that pixel $(1, 2)$ is below pixel $(1, 1)$), because GNNs require permutation invariance.

Just as in a GNN, the computation of the model output for each node (pixel) involves some sort of “message passing” to a node from each of its neighbors (spatially close pixels). For each node, the output is calculated as the convolution between the fixed-size convolutional kernel and the vectors/node features in $\mathbb{R}^3$ corresponding to the color values of each pixel in the neighborhood.

4.2 Definition of the GCN

GNNs can be thought of as analogous to convolutional neural networks and both can be categorized as part of a class of geometric deep learning methods. To go from the design of a CNN to that of a GCN, one must acknowledge the consideration that the structure of a node's neighbourhood is not fixed, and thus the model cannot distinguish between different classes neighbours.

Therefore, rather than a kernel that acknowledges the specific kind of spatial difference between a pixel and its neighbours, the GCN layer only can incorporate information from the nodes themselves. In this sense we can choose functions for AGGREGATE and UPDATE that can mirror those of the convolutional neural network. The formula is below according to [6]:

$$\mathbf{h}_u^{(k+1)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v^{(k)}}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(v)|}}\right) \tag{4.1}$$

This model is the same as the basic GNN model shown in earlier sections, but with added normalizations. While a convolution kernel has specific weighting information depending on the spatial relation between the pixels examined, the GCN simply adds information by including information about the degree (neighbourhood size) of each node in the neighbourhood of u. Despite the introduction of this model as a generalization of a convolutional neural network, Kipf and Welling formulated the model in the form of a first-order approximation of the Chebyshev polynomial of a graph Laplacian, which I will introduce in the next section.

4.3 Connection to graph spectral theory

The graph Laplacian matrix is defined as

$$L = D - A$$

where A is the adjacency matrix and D is the degree matrix. Similarly, the normalized graph Laplacian is

$$\tilde{L} = I - D^{-1/2} A D^{-1/2}$$

where each entry A_{ij} is 1 if $i = j$ and $\frac{1}{d_i}$ if vertices i and j share an edge.

The graph Laplacian and its normalized counterpart are so named because they are analogous to the Laplacian operator $\Delta f = \nabla^2 f$, and may be considered as an operator that characterizes how a graph signal varies across its nodes. The analogy is clear when considering the graph Laplacian of a chain graph of length l. Notice that the rows of L in this case consist of all zeros except for a single block of the form $\begin{bmatrix} \cdots & -1 & 2 & -1 & \cdots \end{bmatrix}$. When appropriately normalized, this corresponds to a linear approximation and discretization of the second derivative operator, when each of the nodes corresponds to an equally-spaced point along the real line (i.e., the kth node corresponds to the point $x = k/l$).

On the real line, the eigenfunctions of the Laplacian operator correspond to the Fourier basis of sinusoidal functions, which have the property that linear combinations of the functions present in this basis can represent any periodic function.

Similarly, any spectral convolution on a graph, $g_\theta \star x$, can be defined as the application of a signal can be defined as

$$g_\theta \star x = U g_\theta(\Lambda) U^\top x$$

where U is the orthogonal matrix with columns equivalent to the eigenvectors of the normalized laplacian L, with g_θ being defined now as a function of the eigenvalues λ or more simply as a function over the eigenvalue matrix Λ.

Evaluating every eigenvalue is computationally expensive, so a truncated version of the signal can be calculated by taking only the first K eigenvalues (rescaled):

$$g_{\theta'}(\Lambda) = \sum_{k=0}^{K} \theta'_k T_k(\hat{\Lambda})$$

where $\hat{\Lambda}$ are the rescaled eigenvalues, being the eigenvalues of the rescaled Laplacian $\hat{L} = \frac{2}{\lambda_{\max}} \tilde{L} - I$.

The GCN formula arises when $\lambda_{\max}$ is approximated to be 2 and K is set to be equal to 1, giving

$$g_{\theta'} \star x = \theta'_0 x - \theta'_1 D^{-1/2} A D^{-1/2} x$$

or, if turned into an update rule, exactly the same formula as Equation 4.1.

It is important to note that in the original paper, while the formulation of the GCN formula as a graph-spectral derivation is correct, it is not grounded in a theoretical argument about the model class's expressivity. There are no theorems in the paper about why $\lambda_{\max}$ is approximated to 2 or why specifically they needed to truncate the Chebyshev polynomial sum at $K = 1$. It instead was verified empirically, posting results that were better than the state-of-the-art graph neural network models at the time. Since then and before, there has been work analyzing the expressivity of these models, which we will begin to discuss in the following sections.

Chapter 5

Introduction to algorithmic learning theory

5.1 Universal approximation for neural networks

The main theoretical underpinning for the expressivity of neural networks begins with the universal approximation theorem for feedforward neural networks, which dates back as early as 1989 [7].

Theorem 5.1. *Let σ be a continuous sigmoidal function. Then finite sums of the form*

$$G(x) = \sum_{j=1}^{N} \alpha_j \sigma(y_j^\top x + \theta_j)$$

are dense in $C(I_n)$, meaning that for any continuous function f on a compact interval I_n, there exists G of the above form such that $|G(x) - f(x)| < \epsilon$ for any $\epsilon > 0$.

The sum of sigmoidal functions in Cybenko's expression is equivalent to a one-layer feedforward neural network of arbitrary width and single output (the property for multiple outputs can be shown using the property for each coordinate and a smaller ϵ).

5.2 PAC theory and VC-dimension

One may be tempted to think that neural networks are theoretically “proven to work” because of the universal approximation theorem, but the statement of the proof still leaves many mysteries. For one, from the proof itself it doesn’t give a characterization for how many parameters are needed to approximate a given function to a desired accuracy ϵ, save for the number of nodes in a single, arbitrarily-wide hidden layer. The vast majority of neural networks in practical usage today are deep rather than wide; the universal approximation proof gives us no reasoning as to why this might be the case. With this in mind, a more granular theory is needed in order to characterize neural networks and to inform model selection when building neural network architectures.

The general framework for learning tasks can be seen as a more general form from the definition in Section 1.1. Given data from a training set Z consisting of data from $\mathcal{X}$ and outputs from $\mathcal{Y}$, we seek to find hypothesis functions h from a function class $\mathcal{H}$ that estimates the ground truth function $f : \mathcal{X} \to \mathcal{Y}$ up to a particular error.

PAC (standing for Probably Approximately Correct)-learning theory is a formal learning model introduced by Valiant in 1984 [8] that serves as a property of whether or not a particular hypothesis class can achieve this. The definition, as restated by Shalev-Shwartz and Ben-David [2], is as follows:

Definition 5.1 (PAC-learnability)**.** *A hypothesis class $\mathcal{H}$ is PAC-learnable if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ (taking arguments ϵ and δ), such that: for every $\epsilon, \delta \in (0,1)$, for every distribution $\mathcal{D}$ over ground set $\mathcal{X}$, and for every labelling function $f : \mathcal{X} \to \{0,1\}$, then if the realizable assumption holds then when running the learning algorithm on $\geq m_{\mathcal{H}}$ iid examples generated by $\mathcal{D}$ and labeled by f, the algorithm returns a hypothesis $h \in \mathcal{H}$ such that, with probability $1-\delta$, the expected loss $L_{\mathcal{D},f}(h) \leq \epsilon$*

In the above an important assumption is made, namely that the function is realizable, i.e. that for every training set $S \in \mathcal{X}^n$ there exists a member $h \in \mathcal{H}$ such that $L_{S,f}[h] = 0$, and it is very easy to see that this case will not be true for all such learning problems. A more general definition is made some time after:

Definition 5.2 (PAC-learnability (agnostic case))**.** *A hypothesis class $\mathcal{H}$ is agnostic PAC-learnable if there exists a function $m_{\mathcal{H}}$ and a learning algorithm with the property: for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when running the learning algorithm on more than $m_{\mathcal{H}}(\epsilon, \delta)$ iid examples generated by $\mathcal{D}$, the algorithm returns an $h \in \mathcal{H}$ such that with probability greater than $1 - \delta$, $L_{\mathcal{D}}(h) \leq \min_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') + \epsilon$.*

PAC-learnability is closely related to the concept of VC-dimension, which is defined as follows:

Definition 5.3 (VC-dimension)**.** *The VC-dimension of a hypothesis class $\mathcal{H}$ is as follows*

$$VCdim(\mathcal{H}) = \max_{S \subseteq \mathcal{X}} \{|S| : \{(h(s_1), \ldots, h(s_{|S|})) : h \in \mathcal{H}\} = \{0,1\}^{|S|}\}$$

Here, the condition on the right is known as the hypothesis class $\mathcal{H}$ shattering the set S.

It has been shown that if the VC-dimension of $\mathcal{H}$ is finite and equal to D, then it is agnostically PAC-learnable with $m(\delta, \epsilon) = \Theta\left(\frac{D + \ln \frac{1}{\delta}}{\epsilon^2}\right)$.

The VC-dimension of feedforward neural networks with ReLU activations is proven to be $\Omega(PL \log \frac{P}{L})$ by Bartlett, Harvey, et al. [9] in 2019, where P is the number of parameters and L is the number of layers. This means that the sample size m for a neural network with parameters P and L and desired accuracy described by (ϵ, δ) is

$$m_{P,L,\epsilon,\delta} = O\left(\frac{1}{\epsilon^2}\left(PL \log \frac{P}{L} + \ln \frac{1}{\delta}\right)\right)$$

Chapter 6

PAC-Bayes theory

The PAC-Bayes framework augments the regular PAC framework by inducing a *prior* distribution over the hypothesis class. A distribution of functions is created by creating a distribution among its indexing parameters; for example, it is equivalent to discuss a distribution of neural networks of a fixed architecture and a distribution of its weights.

Perhaps the "fundamental theorem" of PAC-Bayes theory is that of McAllester [10]. Its statement is as follows:

Theorem 6.1 (PAC-Bayes generalization gap [10])**.** *In the multi-class classification setting with K classes, let*

$$L_{\mathcal{D},\gamma}(f) = \Pr_{x\sim\mathcal{D}}\left[f(x)[y] \leq \max_{j\neq y} f(x)[j] + \gamma\right]$$

be the generalization error, with $f(x)$ outputting a vector of class values in $\mathbb{R}^K$ and γ being a fixed threshold. Similarly, let

$$L_{S,\gamma}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left[f(x_i)[y_i] \leq \max_{j\neq y_i} f(x_i)[j] + \gamma\right]$$

be the empirical loss function over particular training set S.

Given any hypothesis class of functions f parameterized by weights w, for any *prior P and posterior Q over w, for any $\delta \in (0,1)$, we examine the expected losses when picking a hypothesis from Q, which we denote as $L_{\mathcal{D},\gamma}(Q) = \mathbb{E}_{f\sim Q}[L_{\mathcal{D},\gamma}(f)]$ for the expected true data distribution loss and $L_{S,\gamma}(Q) = \mathbb{E}_{f\sim Q}[L_{S,\gamma}(f)]$ for the expected empirical loss. With probability $\geq 1-\delta$ over training sets of size m, we have:*

$$L_{\mathcal{D},\gamma}(Q) \leq L_{S,\gamma}(Q) + \sqrt{\frac{D_{KL}(Q\|P) + \ln\frac{2m}{\delta}}{2(m-1)}} \tag{6.1}$$

Although PAC-Bayes relies on a prior over the weights, unlike Bayesian learning the bounds of PAC-Bayes hold even if the prior is not correct [11]. It is still true that the tightness of the bound depends on the choice of P and Q. Intuitively, it can be observed that distributions that are better able to reflect the reality of the distribution of weights and the mechanism of training are able to produce tighter bounds.

Though it is possible to select two distributions P and Q that have low KL-divergence, that can come at the cost of increasing $L_{\mathcal{D},\gamma}(Q)$. For example, selecting $Q = P$ (perhaps, to have a KL-divergence of 0) is equivalent to doing no training at all, and thus would in all likelihood increase $L_{\mathcal{D},\gamma}(Q)$ significantly.

A prominent example of a posterior distribution is posited by Neyshabour et al. [12], in which they derive a PAC-Bayes bound by setting the posteriors as “perturbation distributions”, defined as follows. If the functions f are parameterized by weights $\mathbf{w}$, and P thus can be seen as a distribution over $\mathbf{w}$, then a perturbation distribution posterior Q is another such distribution denoted by $\mathbf{w}+\mathbf{u}$ where $\mathbf{u}$ comes from a distribution that satisfies certain properties (e.g. they are Gaussian). A generalization bound can be formulated from these as follows.

Theorem 6.2 (Perturbation PAC-Bayes bound [12] If $\mathbf{u}$ is a perturbation distribution such that $\mathbb{P}_{\mathbf{u}}[\max_{x\in\mathcal{X}} |f_{\mathbf{w}+\mathbf{u}}(x) - f_{\mathbf{w}}(x)|_\infty < \gamma/4)$. $\geq 1/2$*, then we have for all* $\mathbf{w}$*, with probability* $\geq 1-\delta$*:*

$$L_{\mathcal{D},0}(f_{\mathbf{w}}) \leq L_{S,\gamma}(f_{\mathbf{w}}) + 4\sqrt{\frac{D_{KL}(Q\|P) + \ln\frac{6m}{\delta}}{m-1}} \tag{6.2}$$

PAC-Bayes analysis has been applied to the case where the hypothesis class is that of feedforward, ReLU-activation neural networks [12]. The bound derived is as such.

Theorem 6.3 (PAC-Bayes bound for RELU-activation neural networks [12])**.** *For any $\delta \in (0,1)$, for any γ used in the multi-class margin loss, and for any $\tilde{\beta}$ we have with probability $\geq 1-\delta$ over training sets of size m:*

$$L_{\mathcal{D},0}(f_{\mathbf{w}}) \leq L_{S,\gamma}(f_{\mathbf{w}}) + \mathcal{O}\left(\sqrt{\frac{B^2 d^2 h \ln(dh) \prod_{i=1}^{d} \|W_i\|_2^2 \sum_{i=1}^{d} \frac{\|W_i\|_F^2}{\|W_i\|_2^2} + \ln \frac{dm}{\delta}}{\gamma^2 m}}\right) \tag{6.3}$$

Since there is an m term in the denominator and the numerator only has a logarithmic dependence on m, this means that as the training set size increases toward infinity the generalization gap approaches 0.

Chapter 7

Expressivity of GNNs

The generalization of learning tasks on fixed-structured data begat a few differing perspectives for quantifying the expressivity of models that work on graphs. There are different kinds of questions that one can use as the basis to form of a quantitative assessment, namely:

- How well can a graph model generalize from data?
- How well does the model differentiate different graph structures?
- How well is the model able to identify substructures of a graph?

There exists a universal approximation for GNNs on the class of permutation-invariant set functions, as well as work on how well they can count substructures and differentiate between non-isomorphic graphs. There is also a PAC-Bayes generalization bound for GNNs, which this work offers a new improvement upon. However, little work exists giving a more precise characterization of which node functions are easily representable by GNNs beyond universal approximation.

7.1 Substructure tests and isomorphism

The problem of GNNs' ability to identify graph structures has been explored in works such as [13] and [14]. If a given graph neural network is able to, given two graphs g_1 and g_2, always return the same output when they are isomorphic, and always return different outputs when they are not, then running two graphs through the GNN becomes an algorithm capable of solving the graph isomorphism decision problem in polynomial time. It has been proven, however, that a single layer of a graph neural network can be no more powerful than the 1-Weisfeiler-Lehman test (1-WL) by [13]; the 1-WL test is not powerful enough to differentiate between all nonisomorphic structures. As a corollary, a multi-layer GNN with k total layers is shown to be no more powerful than the k-WL test.

7.2 Universal approximation of GNNs

A universal approximation theorem of the function class of GNNs is proven in Xu et al., 2020 [15]. The framework is, given $\epsilon > 0$ and permutation-invariant featured-graph function $f : \mathcal{G} \to \mathbb{R}$, there exists a message-passing graph neural network NN that is able to approximate f on a compact space $\mathcal{X}$ of node features such that $\|f - \mathrm{NN}\|_\infty \leq \epsilon$. The proof of this relies on a reduction of the learning task to a learning task on set functions, as well as reducing the GNN model to a Deep Sets model [16], another class of machine model that exhibits (and is built around) a universal approximation theorem for permutation-invariant functions on sets.

7.3 VC-dimension and Rademacher Complexity generalization bounds for GNNs

Scarselli et al. [17] showed that the VC-dimension of a GNN taking input feature vectors in $\mathbb{R}^d$ with maximum hidden embedding dimension h has VC-dimension $O(h^6 N)$, where N is the maximum number of nodes in any of the input graphs, and consequently having generalization error scale by $\tilde{O}(h^6 N/\sqrt{m})$.

Garg et al. [18] tighten this bound further by using a form of analysis pertaining to a concept known as Rademacher complexity. Their theorem involves complicated parameter definitions that are beyond the scope of this thesis, so we do not present it in full detail.

In the following sections we will present the current state-of-the art theoretical understanding of GNNs particularly in the generalization bound setting, and make our own original improvements.

Chapter 8

Towards a tighter theory of GNN generalization

The current state of the art generalization bound for GNN was formulated by [1] using PAC-Bayes theory. The work uses the PAC-Bayes theorem from [12] that pertains to general models parameterized by weights $\mathbf{w}$ is used and applies it to both graph convolutional networks and general message-passing GNNs.

We will present the framework and theorem in detail. The analysis was done with respect to loss function $L_{\mathcal{D},\gamma}[f]$ and $L_{S,\gamma}[f]$, pertaining to a *multi-class margin loss* with respect to a fixed margin parameter γ. These are defined as

$$L_{\mathcal{D},\gamma}[f_w] = \mathbb{P}_{z\sim\mathcal{D}}\left(f_w(X,A)[y] \leq \gamma + \max_{j\neq y} f_w(X,A)[j]\right) \tag{8.1}$$

$$L_{S,\gamma}[f_w] = \frac{1}{m}\sum_{(X_i,A_i)\in S}\left(f_w(X_i,A_i)[y] \leq \gamma + \max_{j\neq y} f_w(X_i,A_i)[j]\right) \tag{8.2}$$

These definitions are analogous to those in the general PAC-Bayes setting as in Section 6, but adapted to a setting with graphs as data points. Here y is the ground truth, and m is the training set size.

Perturbation bounds are established for each of the GCN and general message-passing GNN, leading to generalization bounds formulated as in [12] in terms of the spectral norms of each of the weights, as well as in terms of the maximum degree of input graphs d.

Theorem 8.1 (GCN Generalization bound [1])**.** *For any $B > 0$, $l > 1$, let $f_w \in \mathcal{H} : \mathcal{X} \times \mathcal{G} \to \mathbb{R}^k$ be an l-layer GCN. Then with probability $\geq 1 - \delta$ over the choice of an iid size-m training set S from the data distribution $\mathcal{D}$, we have for any w:*

$$L_{\mathcal{D},0}(f_w) \leq L_{S,\gamma}(f_w) + \mathcal{O}\left(\sqrt{\frac{B^2 d^{l-1} l^2 h \log(lh) \prod_{i=1}^{l} \|W_i\|_2^2 \sum_{i=1}^{l} (\|W_i\|_F^2 / \|W_i\|_w^2) + \log \frac{ml}{\delta}}{\gamma^2 m}}\right) \tag{8.3}$$

In the course of this research, a particular point of improvement was indentified with this bound, particularly the exponential dependence on the maximum degree d of dataset graphs. The improvements made are discussed in the following section.

Chapter 9

Improvement on GNN PAC-Bayes bound

Improving on Liao et al.'s [1] work, we are able to prove the following:

Theorem 9.1. *For any $B > 0$, $l > 1$, let $f_w \in \mathcal{H} : \mathcal{X} \times \mathcal{G} \to R^k$ be an l-layer GCN. Then with probability $\geq 1 - \delta$ over the choice of an iid size-m training set S from the data distribution $\mathcal{D}$, we have for any w:*

$$L_{\mathcal{D},0} \leq L_{\mathcal{S},\gamma} + \mathcal{O}\left(\sqrt{\frac{B^2 d l^2 \beta^2 (h + \ln(l)) \prod_{i=1}^{l} \|W_i\|_2^2 \sum_{i=1}^{l} \frac{\|W_i\|_F^2}{\|W_i\|_2^2} + \ln \frac{m}{\delta}}{\gamma^2 m}}\right) \tag{9.1}$$

The proof is as follows, and makes up the remainder of the chapter.

9.1 Improvement on degree dependency

In Liao et al. [1], a generalization bound is attained on graph convolutional networks; this bound is dependent on a bound on the maximum perturbation of the function value when a perturbation U is applied to the weights W, presented in that paper's Lemma 3.1. The bound is as follows

$$|f_{w+u}(X, A) - f_w(X, A)|_2 \leq eBd^{\frac{l-1}{2}} \left(\prod_{i=1}^{l} \|W_i\|_2 \right) \sum_{k=1}^{l} \frac{\|U_k\|_2}{\|W_k\|_2} \tag{9.2}$$

The primary goal of this set of improvements is to reduce the factor of $d^{\frac{l-1}{2}}$. For each layer, let $H_i \in \mathbb{R}^{|V| \times h}$ be the matrix containing the hidden embeddings of all of the nodes in its rows, with h being the hidden dimension. In the process of the proof of Theorem 9.1, we are able to show the following:

$$\begin{aligned}
\Phi_j &= \max_i |H_j[i,:]|_2 \leq d^{\frac{j}{2}} B \prod_{i=1}^{j} \|W_i\|_2 && (9.3) \\
\Psi_j &= \max_i |H'_j[i,:] - H_j[i,:]|_2 \\
&\leq Bd^{\frac{j}{2}} \left(\prod_{i=1}^{j} \|W_i\|_2 \right) \sum_{k=1}^{j} \frac{\|U_k\|_2}{\|W_k\|_2} \left(1 + \frac{1}{l}\right)^{j-k} && (9.4) \\
|\Delta_l|_2 &= \left| \frac{1}{n} \mathbf{1}_n H'_{l-1}(W_l + U_l) - \frac{1}{n} \mathbf{1}_n H_{l-1} W_l \right|_2 \\
&\leq eBd^{\frac{l-1}{2}} \left(\prod_{i=1}^{l} \|W_i\|_2 \right) \left[\sum_{k=1}^{l} \frac{\|U_k\|_2}{\|W_k\|_2} \right] && (9.2)
\end{aligned}$$

We begin to simplify these bounds by removing the dependency on $d^{\frac{j}{2}}$, replacing it instead with a fixed power of $d^{1/2}$ that remains constant for every layer, and thus in the final result of Equation 9.2 as well.

Theorem 9.2. *For all $1 \leq j \leq l - 1$, we have:*

$$\begin{aligned}
\Phi_j &\leq \sqrt{d} B \prod_{i=1}^{k} \|W_i\|_2 && (9.5) \\
\Psi_j &\leq \left(1 + \left(1 + \frac{1}{l}\right)^j\right) B \sqrt{d} \left(\prod_{i=1}^{j} \|W_i\|_2 \right) && (9.6)
\end{aligned}$$

Finally,

$$|f_{w+u}(X, A) - f_w(X, A)|_2 = |\Delta_l|_2 \leq \left(e + 1 + \frac{2}{l}\right) B \sqrt{d} \prod_{i=1}^{l} \|W_i\|_2 \tag{9.7}$$

The proof follows from a lemma about the 2-norm of any node representation at any layer:

Lemma 9.3. *We have, for all $k \in [n]$ and for $j \in [l]$:*

$$|H_j[u, :]|_2 \leq B \sqrt{deg(u)} \left(\prod_{i=1}^{j} \|W_i\|_2\right) \tag{9.8}$$

Proof: We prove this by induction. By definition $|H_0[u, :]|_2 \leq B$ and thus

$$|H_0[u]| \leq \sqrt{\deg(u)} B \prod_{k=1}^{0} \|W_k\|_2.$$

We assume that for all u, we have

$$H_{j-1}[u, :] \leq \sqrt{\deg(u)} B \prod_{k=1}^{j-1} \|W_i\|_2.$$

From these statements we are able to deduce

$$\begin{aligned}
|H_j[u,:]| &\leq \sum_{v\in\mathcal{N}_u} \tilde{L}[u,v]|H_{j-1}[v,:]|_2\|W_j\|_2 \\
&\leq \sum_{v\in\mathcal{N}_u} \frac{1}{\sqrt{\deg(u)\deg(v)}}\left[\sqrt{\deg(v)}B\prod_{k=1}^{j-1}\|W_k\|_2\right]\|W_j\|_2 \\
&= \sum_{v\in\mathcal{N}_u} \frac{1}{\sqrt{\deg(u)}}B\left(\prod_{k=1}^{j-1}\|W_k\|_2\right)\|W_j\|_2 \\
&= \frac{\deg(u)}{\sqrt{\deg(u)}}B\prod_{k=1}^{j}\|W_k\|_2 \\
&= \sqrt{\deg(u)}B\prod_{k=1}^{j}\|W_k\|_2
\end{aligned} \tag{9.9}$$

In these inequalities we use the fact that $\tilde{L}[i,j] = (A+I)_{ij}/\sqrt{\deg(i)\deg(j)}$, and we assume the simple case where there are unweighted edges so that $(A+I)_{ij}$ is 1 if and only if nodes i and j are connected and 0 otherwise.

By Lemma 9.3, we have that $\Phi_j = \max_i |H_j[i,:]|_2 \leq \sqrt{d}B\prod_{i=1}^{j}\|W_i\|_2$, which is exactly the result of equation (9.5).

Claim: For all $v \in [n]$, $|\Delta_j[v,:]|_2 \leq B\sqrt{\deg(v)}\left(1+\frac{1}{l}\right)^j\left(\prod_{i=1}^{j}\|W_i\|\right)\left(\sum_{i=1}^{j}\frac{\|U_i\|}{\|W_i\|}\right)$.

Proof: We use induction assuming this is true for Δ_{j-1}. We then have

$$\begin{aligned}
|\Delta_j[v,:]|_2 &\le \sum_{u\in\mathcal{N}(v)} \tilde{L}[v,u]|H'_{j-1}[u,:] - H_{j-1}[u,:]|_2\|W_j + U_j\|_2 + \sum_{u\in\mathcal{N}(v)} \tilde{L}[v,u]|H_{j-1}[u,:]|_2\|U_j\|_2 \\
&\le \left[B\left(1+\frac{1}{l}\right)^{j-1}\left(\prod_{i=1}^{j-1}\|W_i\|\right)\left(\sum_{i=1}^{j-1}\frac{\|U_i\|_2}{\|W_i\|_2}\right)\|W_j+U_j\| + B\|U_j\|\prod_{i=1}^{j-1}\|W_i\|\right] \qquad (9.10)
\end{aligned}$$

$$\left(\sum_{u\in\mathcal{N}(v)} \tilde{L}[v,u]\sqrt{\deg(u)}\right)$$

$$\begin{aligned}
&= B\sqrt{\deg(v)}\prod_{i=1}^{j-1}\|W_i\|\left[\|W_j+U_j\|\left(1+\frac{1}{l}\right)^{j-1}\left(\sum_{i=1}^{j-1}\frac{\|U_i\|_2}{\|W_i\|_2}\right)+\|U_j\|\right] \\
&= B\sqrt{\deg(v)}\prod_{i=1}^{j}\|W_i\|\left[\frac{\|W_j+U_j\|_2}{\|W_j\|_2}\left(1+\frac{1}{l}\right)^{j-1}\left(\sum_{i=1}^{j-1}\frac{\|U_i\|_2}{\|W_i\|_2}\right)+\frac{\|U_j\|_2}{\|W_j\|_2}\right] \\
&\le B\sqrt{\deg(v)}\prod_{i=1}^{j}\|W_i\|\left[\left(1+\frac{1}{l}\right)^{j}\left(\sum_{i=1}^{j-1}\frac{\|U_i\|_2}{\|W_i\|_2}\right)+\frac{\|U_j\|_2}{\|W_j\|_2}\right] \\
&\le B\sqrt{\deg(v)}\prod_{i=1}^{j}\|W_i\|\left(1+\frac{1}{l}\right)^{j}\left(\sum_{i=1}^{j}\frac{\|U_i\|_2}{\|W_i\|_2}\right) \qquad (9.11)
\end{aligned}$$

Δ_l has a slightly different formulation but it has a very similar bound:

$$\begin{aligned}
|\Delta_l|_2 &= \left|\frac{1}{n}\mathbf{1}_n\left(\tilde{L}H'_{l-1}(W_l+U_l) - \frac{1}{n}\mathbf{1}_n\tilde{L}H_{l-1}(W_l)\right)\right|_2 \\
&= \frac{1}{n}\left|\mathbf{1}_n\tilde{L}(H'_{l-1}-H_{l-1})(W_l+U_l) + \mathbf{1}_n\tilde{L}H_{l-1}(U_l)\right|_2 \\
&\le \frac{1}{n}\sum_{i=1}^{n}|\Delta_{l-1}[i,:]|_2\|W_l+U_l\|_2 + \frac{1}{n}\sum_{i=1}^{n}|H_{l-1}[i,:]|_2\|U_l\|_2 \\
&\le B\sqrt{d}\prod_{i=1}^{l-1}\|W_i\|\left(1+\frac{1}{l}\right)^{l-1}\left(\sum_{i=1}^{l-1}\frac{\|U_i\|_2}{\|W_i\|_2}\right)\|W_l+U_l\| \\
&\qquad + B\sqrt{d}\|U_l\|_2\prod_{i=1}^{l-1}\|W_i\|_2 \\
&\le B\sqrt{d}\prod_{i=1}^{l}\|W_i\|\left[\left(1+\frac{1}{l}\right)^{l}\left(\sum_{i=1}^{l-1}\frac{\|U_i\|}{\|W_i\|}\right)+\frac{\|U_l\|}{\|W_l\|}\right] \\
&\le B\sqrt{d}\prod_{i=1}^{l}\|W_i\|\left(1+\frac{1}{l}\right)^{l}\left(\sum_{i=1}^{l}\frac{\|U_i\|}{\|W_i\|}\right) \\
&\le eB\sqrt{d}\prod_{i=1}^{l}\|W_i\|\left(\sum_{i=1}^{l}\frac{\|U_i\|}{\|W_i\|}\right)
\end{aligned}\tag{9.12}$$

From this we have proven a tighter bound on the final output of the GNN under perturbation, which we will use to calculate probabilistic and generalization bounds.

9.2 Improvement on probabilistic bounds using random matrix theory

In [1], for all $i \in [l]$, with l being the number of layers, the prior and the distribution of the perturbations $U_i \in \mathbb{R}^{d_{i+1}\times d_i}$,, where all hidden dimensions d_i are upper-bounded by a value h, were generated by a normal distribution $\mathcal{N}(0,\sigma^2 I)$, and give probabilistic bounds on the operator norms $\|U_i\|$ as $P\left(\forall i, \|U_i\| \le t\right)$ with probability greater than $1 - 2lh\exp -t^2/2h^2$. We improve these bounds using theorems on random matrices from work on high-dimensional probability, namely [19].

Theorem 9.4 ((Theorem 4.4.5 in [19]). *Let A be a matrix in $\mathbb{R}^{m\times n}$, where the entries A_{ij} are independent, mean-zero, sub-Gaussian random variables. Then, for all $t > 0$ we have*

$$\|A\| \leq CK(\sqrt{m} + \sqrt{n} + t)$$

with probability $\geq 1 - \exp(-t^2)$, where $K = \max_{i,j} \|A_{ij}\|_{\psi_2}$ and C is some constant.

In the above theorem the norm $\|X\|_{\psi_2}$ is defined as $\inf\{t : \mathbb{E}[\exp(X^2/t^2)] \leq 2\}$. In Example 2.5.8 in [19], it is shown that if $X \sim \mathcal{N}(0, \sigma^2)$ then it has $\|X\|_{\psi_2} \leq C\sigma$.

Corollary 9.4.1. *If $U \in \mathbb{R}^{m\times n}$ is a random matrix generated with the distribution $\mathcal{N}(0, \sigma^2 I)$ (i.e. all entries are independent and identically distributed Gaussian random variables), then we have*

$$\|U\| \leq \sigma(\sqrt{m} + \sqrt{n} + t)$$

with probability at least $1 - 2\exp(-t^2)$.

With a change of variable, we are able to calculate the following:

$$\begin{aligned}
P\left(\forall i. \|U_i\|_2 \leq t\right) &\geq 1 - P(\exists i, \|U_i\| > t) \\
&\geq 1 - \sum_{i=1}^{l} P(\|U_i\| > t) \\
&\geq 1 - 2l \exp\left(\left(\frac{t}{C\sigma} - 2\sqrt{h}\right)^2\right)
\end{aligned}$$

And by setting the right-hand side to 1/2, we obtain:

$$t = C\sigma(2\sqrt{h} + \sqrt{\ln(4l)})$$

Using the above equation combined with our bound we are able to get

$$
\begin{aligned}
|f_{w+u}(X,A) - f_w(X,A)|_2 &\leq eB\sqrt{d}l\left(\prod_{i=1}^{l}\|W_i\|_2\right)\sum_{k=1}^{l}\frac{\|U_k\|_2}{\|W_k\|_2} \\
&= eB\sqrt{d}\beta^l l\sum_{k=1}^{l}\frac{\|U_k\|_2}{\beta} \\
&\leq eB\sqrt{d}\beta^{l-1}l(\sigma(2\sqrt{h}+\sqrt{\ln(4l)})) \\
&\leq e^2B\sqrt{d}\tilde{\beta}^{l-1}(\sigma(2\sqrt{h}+\sqrt{\ln(4l)})) \leq \frac{\gamma}{4} \qquad (9.13)
\end{aligned}
$$

Here $\tilde{\beta}$ is an estimated of β such that $|\beta - \tilde{\beta}| \leq \beta/l$ that can be generated *a priori*; we discuss this in a later subsection.

We can set $\sigma = \frac{\gamma}{4e^2B\tilde{\beta}\sqrt{d}C\left(2\sqrt{h}+\sqrt{\ln(4l)}\right)}$ to satisfy the final inequality. From this we can calculate the KL-divergence between the posterior and the prior:

$$
\begin{aligned}
\mathrm{KL}(Q\|P) = \frac{|w|_2^2}{2\sigma^2} &= \frac{16e^4B^2dl^2\beta^{2(l-1)}\left(2\sqrt{h}+\sqrt{\ln(4l)}\right)^2}{2\gamma^2}\sum_{i=1}^{l}\|W_i\|_F \\
&\leq \mathcal{O}\left(\frac{B^2d\beta^{2l}l^2(h+\ln(l))}{\gamma^2}\sum_{i=1}^{l}\frac{\|W_i\|_F^2}{\beta^2}\right) \\
&\leq \mathcal{O}\left(B^2dl^2\,(h+\ln(l))\,\frac{\prod_{i=1}^{l}\|W_i\|^2}{\gamma^2}\sum_{i=1}^{l}\frac{\|W_i\|_F^2}{\|W_i\|^2}\right) \qquad (9.14)
\end{aligned}
$$

From this we are able to calculate the generalization bound and thus prove the theorem.

$$
L_{\mathcal{D},0} \leq L_{\mathcal{S},\gamma} + \mathcal{O}\left(\sqrt{\frac{B^2dl^2(h+\ln(l))\prod_{i=1}^{l}\|W_i\|_2^2\sum_{i=1}^{l}\frac{\|W_i\|_F^2}{\|W_i\|_2^2}+\ln\frac{m}{\delta}}{\gamma^2m}}\right) \qquad (9.15)
$$

9.3 Selecting parameter $\tilde{\beta}$

The prior normal distribution's variance parameter σ^2 is dependent on β, but β cannot be used in its calculation because that information is only known after model training. Instead, we can select a parameter $\hat{\beta}$ such that $|\beta - \hat{\beta}| \leq \frac{1}{l}\beta$ and thus $\frac{1}{e}\beta^{l-1} \leq \hat{\beta}^{l-1} \leq e\beta^{l-1}$ (as per equation 33 in [1]).

As in [1] we only have to consider values of β in the range

$$\left(\frac{\gamma}{2B\sqrt{d}}\right)^{1/l} \leq \beta \leq \left(\frac{\gamma\sqrt{m}}{2B\sqrt{d}}\right)^{1/l}$$

as otherwise the generalization bound holds trivially because $L_{\mathcal{D},0} \leq 1$ by definition.

If we consider values of $\hat{\beta}$ that cover this interval then by union bound we are still able to get a high probability; the covering C needs to have $|C| = \frac{l}{2}(m^{\frac{1}{2l}} - 1)$.

Chapter 10

Experiments

If the generalization gap bound is low enough then it would be possible for algorithms that exist that may provide insight about the learning problem before training. With this insight, it may be possible to greatly simplify the currently costly and difficult model selection process by being able to compare different models' expected performance before having to start a computation-intensive training process. Similar preprocessing mechanisms exist for linear/integer/quadratic/mixed-integer program optimizers like CPLEX and company, where the solver is able to select the fastest or most accurate algorithm depending on the parameters of the problem.

10.1 Experimental methodology

We conduct experiments in order to compare our PAC-Bayes bound to the original PAC-Bayes bound in [1], on multiple datasets also used in that paper. These datasets consist of six generated datasets of random graph data, three social network datasets (IMDBBINARY and IMDBMULTI of data from the Internet Movie Database, and COLLAB, a dataset of academic collaborations), and a bioinformatics dataset, PROTEINS, from [16].

Experiments were done on multi-class classification tasks involving each dataset, with prior PAC-Bayes bound coming from [1] as well as our formula proven in Section **??** being calculated after training a GCN-based neural network for a set number of epochs.

We use the following formula for the generalization bound from [1], using an explicit constant factor:

$$\mathrm{GenGap}(B, d, l, \{W_i\}_{i=1}^{l}) = \sqrt{42 \cdot \frac{B^2 d^{l-1} l^2 \ln(4lh) \prod_{i=1}^{l} \|W_i\|_2^2 \sum_{i=1}^{l} \frac{\|W_i\|_F^2}{\|W_i\|_2^2}}{\gamma^2 m}} \tag{10.1}$$

Similarly, the formula used for the new PAC-Bayes generalization bound is

$$\mathrm{GenGap}(B, d, l, \{W_i\}_{i=1}^{l}) = \sqrt{42 \cdot \frac{B^2 d l^2 (h + \ln(l)) \prod_{i=1}^{l} \|W_i\|_2^2 \sum_{i=1}^{l} \frac{\|W_i\|_F^2}{\|W_i\|_2^2}}{\gamma^2 m}} \tag{10.2}$$

Testing was done for different network depths, with values of $l = 4$ and $l = 6$.

The setup works by training a GCN-based neural network architecture on each of the datasets, then computing the theoretical bounds at the end of training.

10.2 Results

Tables below are for calculated bounds in the case of 4 layers (Table 10.1) and 6 layers (Table 10.2).

	Liao et al. (2021) [1]	New bound
ER-1	14.985735 ±0.071	11.727639 ±0.071
ER-2	15.972616 ±0.068	12.080796 ±0.068
ER-3	16.533779 ±0.038	12.285284 ±0.038
ER-4	16.869156 ±0.000	12.391819 ±0.000
SBM-1	15.859210 ±0.080	12.030569 ±0.080
SBM-2	15.503044 ±0.066	11.892126 ±0.066
IMDBBINARY	17.370839 ±0.079	12.458184 ±0.079
IMDBMULTI	16.466189 ±0.038	11.977553 ±0.038
COLLAB	19.773157 ±0.009	13.574678 ±0.009
PROTEINS	14.011104 ±0.079	10.753008 ±0.079

Table 10.1: Table of results, 4 layers (log values)

	Liao et al. (2021)	New bound
ER-1	20.123308 ±0.046	13.607115 ±0.046
ER-2	21.814967 ±0.017	14.031327 ±0.017
ER-3	22.782537 ±0.008	14.285547 ±0.008
ER-4	23.377554 ±0.021	14.422881 ±0.021
SBM-1	21.605130 ±0.037	13.947847 ±0.037
SBM-2	21.028283 ±0.047	13.806447 ±0.047
IMDBBINARY	24.648394 ±0.058	14.823084 ±0.058
IMDBMULTI	23.373624 ±0.105	14.396352 ±0.105
COLLAB	28.261012 ±0.106	15.864055 ±0.106
PROTEINS	19.421826 ±0.059	12.905633 ±0.059

Table 10.2: Table of results, 6 layers (log values)

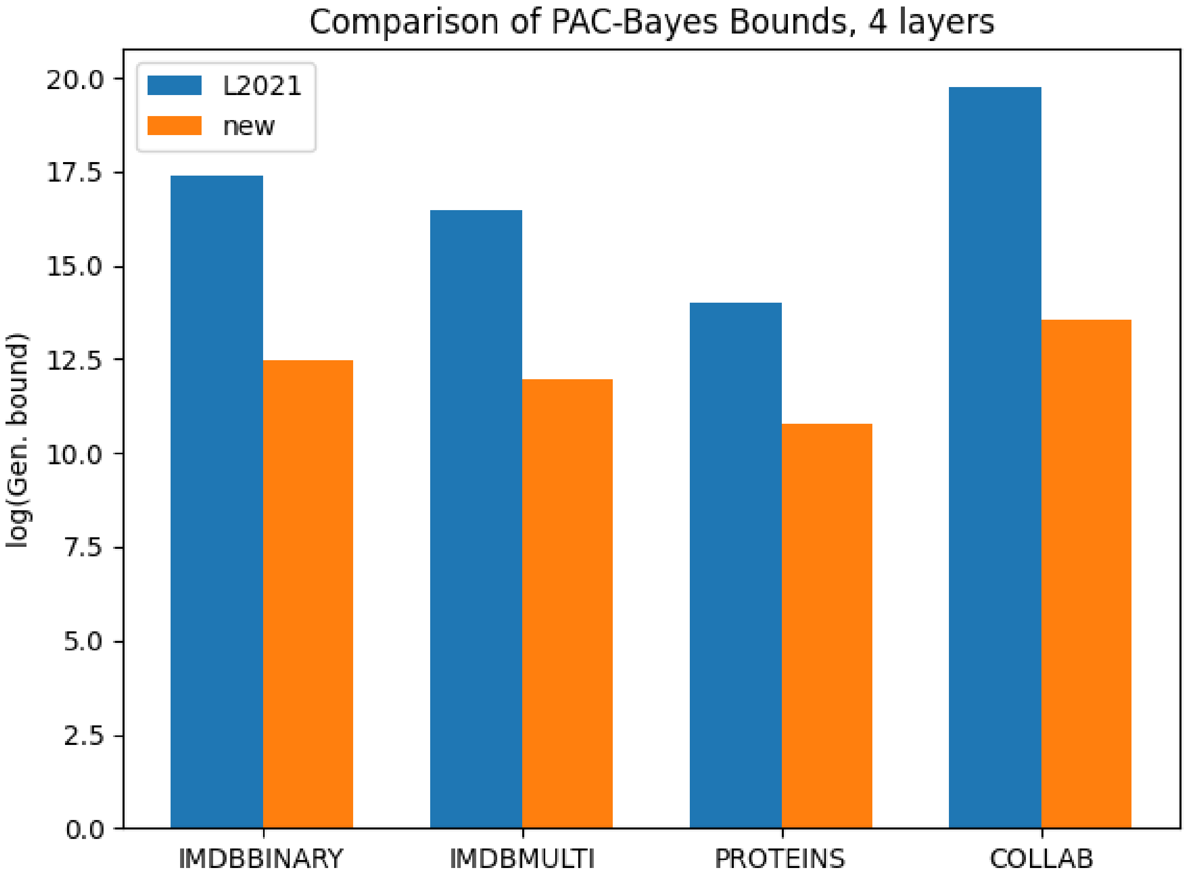

Figure 10.1: Generalization bound values on real-world data sets, 4 layers

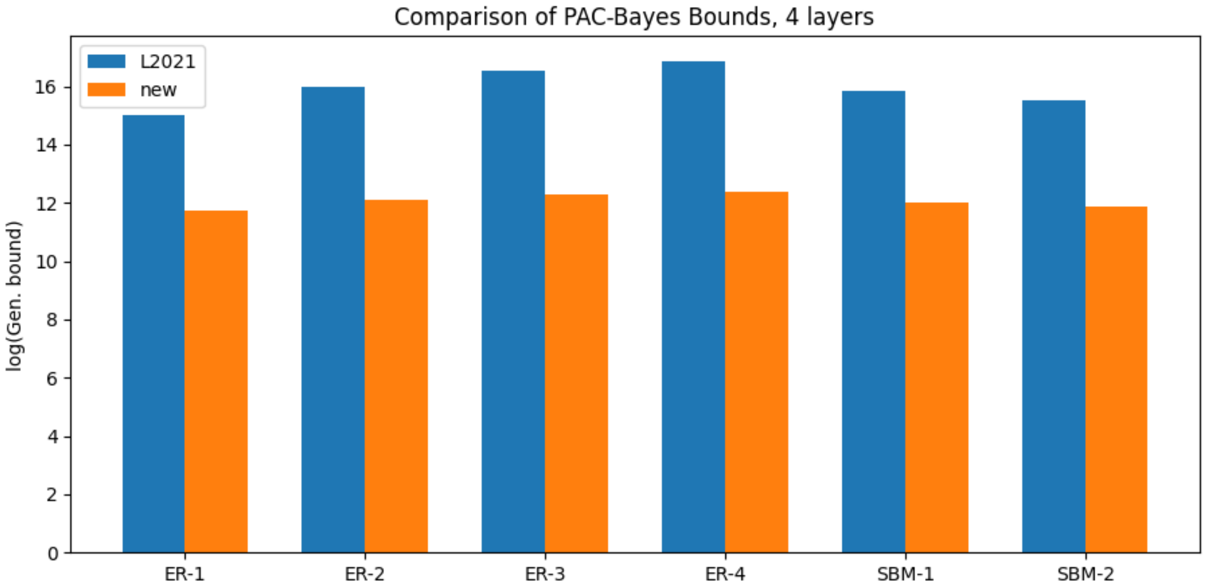

Figure 10.2: Generalization bound values on synthetic graph data sets, 4 layers

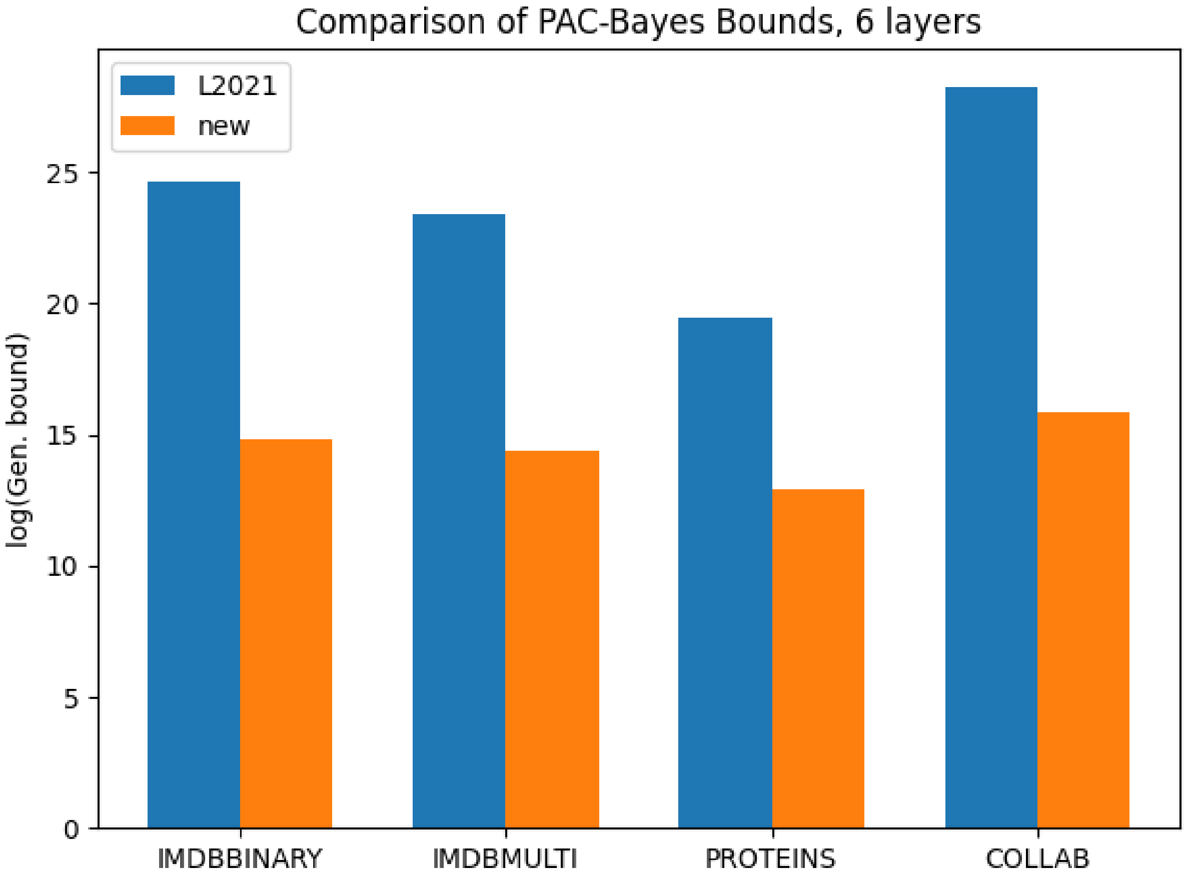

Figure 10.3: Generalization bound values on real-world data sets, 6 layers

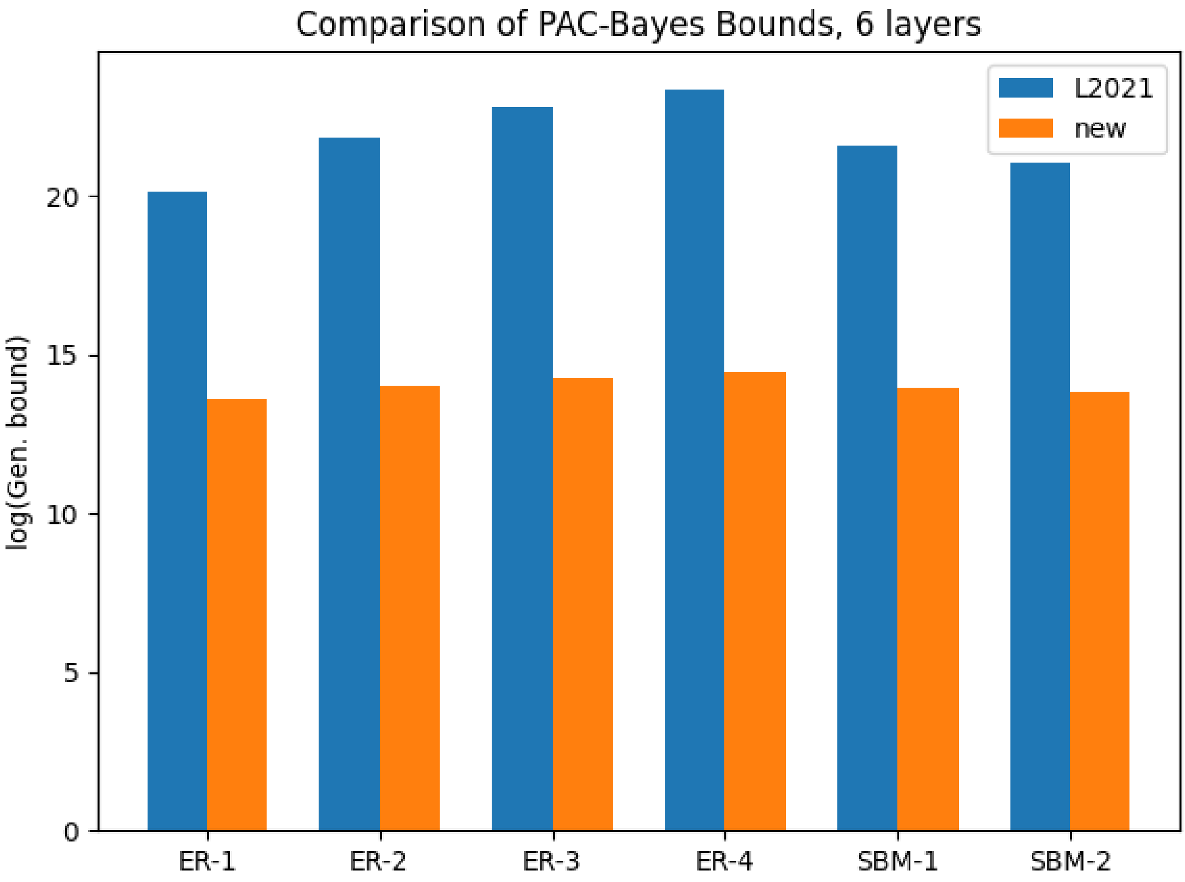

Figure 10.4: Generalization bound values on synthetic graph data sets, 6 layers

10.3 Discussion

Unilaterally, the new bound is a strict improvement on the bound in [1] whenever there is more than one layer in the network, i.e. $l > 1$. The results show that in all cases that the new bound is several orders of magnitude lower than the old bound. As the number of layers l increases, the improvements are of higher magnitude, as the previous results had an exponential dependence on l while our work only has a linear dependence.

The results shown from this experiment still show that if this paradigm is feasible, there is still much work left to be done, as the generalization bounds from both the original paper of [1] and the improved bounds in this work still generate large enough bounds to need to be expressed in log-form. But improvements on theoretical generalization bounds could be important steps toward the realization of this goal.

The values obtained during the course of experiments are still quite large, large enough that we need to express them with log values rather than the actual values. As can be seen in the formula for both generalization bounds, there is still an exponential dependence on the maximum hidden dimension h (due to the dependence on spectral norms $\|W_i\|_2$. The number of training examples needed to reduce the absolute numbers sufficiently thus also depend exponentially on h.

While these high bound values may be the case for this result as well as many others in the state of the art of learning theory, it is still important to decrease the dependencies on the parameters as much as possible.

Chapter 11

Towards developing a theory for size generalizaton

In PAC and PAC-Bayes we examine the relationship between a true/population distribution $\mathcal{D}$ and training sets S that come from the empirical distribution of iid samples, $\mathcal{D}^m$.

In the previous setting, the support of the distribution $\mathcal{D}$ is $\mathcal{X} \times \mathcal{Y}$, consisting of the dataset graphs ($\mathcal{X}$ = the set of graphs) as well as labels (multiclass, $\mathcal{Y} = [n]$ for some n).

PAC-Bayes (and PAC) theory pertain to statements of the form: for all $f \in \mathcal{H}$, we have with probability $\geq 1 - \delta$:

$$L_{\mathcal{D}}[f] \leq L_{\mathcal{D}^m}[f] + A(\mathcal{X}, \mathcal{Y}, f, \delta)$$

where $L_{\mathcal{P}}[f]$ for an arbitrary distribution $\mathcal{P}$ is $\mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(f; x, y)]$ for some loss function ℓ. In particular all of the theorems of PAC and PAC-Bayes hinge on the particular relationship between the distributions on the LHS and RHS of the expression: the one on the right is the empirical sampling distribution of the one on the left, with size m.

In the new graph-sampling regime, we assume that the dataset distribution $\mathcal{X} \times \mathcal{Y}$ comes from a particular process dependent on a very large graph $\mathcal{G}$ (with square symmetric adjacency matrix A), where $\mathcal{X}$ consists entirely of subgraphs of $\mathcal{G}$ that can be derived by random walks of a certain length.

Consider two such lengths N and M, with $M \gg N$, and the corresponding dataset distributions $\mathcal{D}_0$ for the length-N subgraphs and $\mathcal{D}_1$ for the length-M subgraphs. We are interested in proving a statement of the form: for all $f \in \mathcal{H}$, we have with probability $\geq 1 - \delta$:

$$L_{\mathcal{D}_1}[f] \leq L_{\mathcal{D}_0}[f] + B(\mathcal{G}, N, M, f, \delta)$$

and perhaps as a corollary:

$$L_{\mathcal{D}_1}[f] \leq L_{\mathcal{D}_0^n}[f] + A(\cdots) + B(\cdots)$$

Given that $\mathcal{H}$ is a class of GNNs, this corollary (and the setup that preceded it) would give a theoretical framework and answer to the question of: given a dataset of small graphs, how well could a trained GNN generalize to a larger graph?

11.1 Homophilous graphs and graph signals

A learning task that is natural to examine for the sake of understanding size generalization is that of node classification. Suppose, for example, that each node is a person, the neighborhood information is social connections, and the task is to identify whether a cluster of people primarily identify as being part of Group A or Group B.

Homophily [20] is a concept in network science that refers to the property that like nodes group together, which for the node-labelling case means that nodes that are neighbours are likely to have the same label. Size generalization is natural when the labelling of the nodes exhibits homophily; we will call this a homophilic **signal**. The presence of a homophilic signal implies that the labels of the nodes are unlikely to change during the course of a long random walk on the graph.

It is also important to note that homophily also is a concept that applies to the graph topology, as not every possible graph structure can be given a labelling that

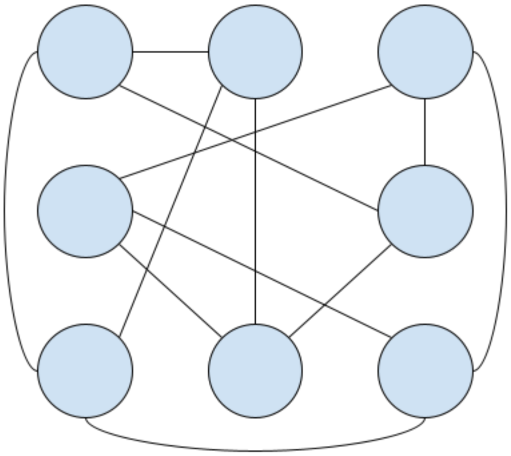

Figure 11.1: An example of a small expander graph. Any labelling of its nodes cannot exhibit homophily.

exhibits homophilic properties. An example of one such topology where homophily is not possible is an expander graph [21], as in Figure 11.1, where nodes have either random or random-like edges connected to a constant number of other nodes in the entire graph. In this case, any labelling of the nodes only have limited implications on the homophily of the signal.

A setting with more homophily is akin to a barbell graph, as in Figure 11.2, where there are two densely connected components, and comparatively few edges connecting the two dense regions. If the signal of interest lines up with these divisions generated by the topology, then it is natural to see that it exhibits a homophilic property.

Results in graph theory give a mathematical description of homophily, in the form of spectral analysis of a graph's Laplacian matrix.

Cheeger's inequality [21] is a theorem from spectral graph theory that pertains to partitions of graphs, which have a natural connection to binary-valued signals on graphs (one side of the partition gets value 0, the other 1). Central to the theorem is the concept of conductance $\phi(S)$ (where S is a subset of the vertices of a graph) defined as

$$\phi(S) = \frac{E(S, \bar{S})}{|S|}$$

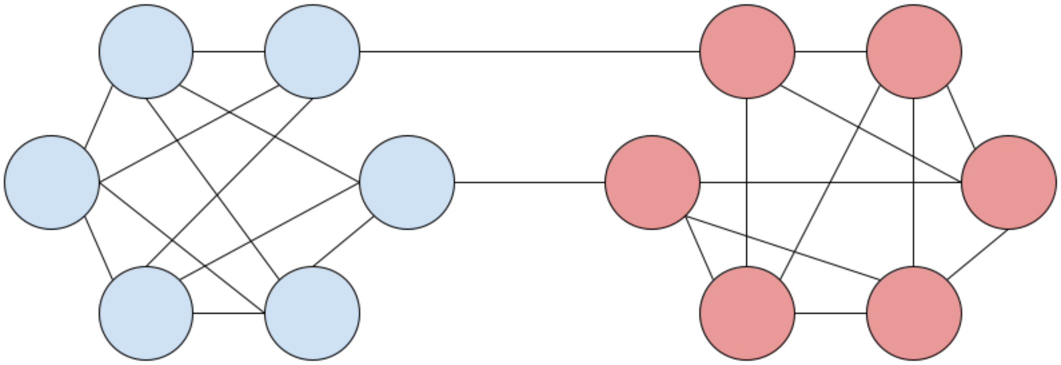

Figure 11.2: Example of a small barbell graph. If a signal is exactly differentiated between the two groups, then it exhibits homophily.

Here $E(S, \bar{S})$ is the set of edges that connect one node in S to a node outside of S. Furthermore, the conductance of the whole graph, $\phi(G)$, is defined as

$$\phi(G) = \min_{|S| \leq \frac{|V|}{2}} \phi(S)$$

i.e. the size of the min-cut. The theorem states that

$$\lambda_2/2 \leq \phi(G) \leq \sqrt{2\lambda_2}$$

where λ_2 is the second-smallest eigenvalue of the normalized Laplacian

$$\tilde{L} = D^{-1/2}(D - A)D^{-1/2}.$$

Cheeger's inequality links the real-valued quantity of λ_2 to the abstract concept of homophily. If λ_2 is small then the conductance of G must also be low, and a low-conductance graph induces a related partition S. If a signal on graph nodes $f : V(\mathcal{G}) \rightarrow \{0, 1\}$ coincides with this partition, i.e. one side of the partition S is generally labelled 0 and the complement $\bar{S}$ is generally labelled 1, then the signal f exhibits homophily.

This property of signals in general also has connections to the spectral properties of the graph's Laplacian matrix. We have the following result linking λ_2 to general graph signals f:

$$\lambda_2 = \min_{f \perp d} \frac{f^\top L f}{f^\top D f}$$

where $f \perp d$ means signals f that are orthogonal to the degree vector d, i.e. $f^\top d = 0$. For d-regular graphs this quantity is equivalent to $\min\limits_{f \perp \mathbf{1}} \frac{f^\top \tilde{L} f}{f^\top f}$.

Thus, upper-bounding λ_2, Cheeger's inequality can take the form

$$\forall f \perp \mathbf{1}, \phi(G) \leq \sqrt{2} \cdot \frac{\sqrt{f^\top L f}}{\|f\|_2}$$

In the following sections we will use this mathematical formalism of homophily to quantitatively identify graph topologies and signals for which size generalization is natural, accompanied by provable probabilistic bounds for size generalization error.

11.2 A probability bound for partition crosses

Recall that the random-walk-sampling out-of-distribution problem involves the distributions of random walk-induced subgraphs of a large graph G of two lengths: the subgraphs from random walks of length N comprise the training set, and we wish to test the resulting trained GNN's performance on subgraphs induced by length-M random walks, where M is much larger than N.

Cheeger's inequality can be related to this problem in the following fashion, at least for the case of d-regular graphs.

Theorem 11.1. *Let $u_1, u_2, \ldots$ be a random walk over the nodes of a connected, d-regular graph G, with u_1 chosen from the stationary distribution of the nodes of G. If $M \leq d/(2^{5/2}\sqrt{\lambda_2})$, then the probability of crossing over the sparsest-cut partition throughout the first M steps of the random walk is under* $1/2$.

Proof. Because u_1 is chosen from the stationary distribution (uniform over vertices, because G is connected and d-regular), then for all $i \geq 1$ the distribution for u_i, u_{i+1} follows the distribution Unif$[E]$, where E is the edge set of the graph.

Let S be the sparsest-cut partition of G. Let X_i be the indicator of the event that the vertex pair is in the set of edges crossing the partition, namely $\mathbf{1}\{(u_i, u_{i+1}) \in E(S, \bar{S})\}$. By linearity of expectation, this means that $E[X_i] = |E(S, \bar{S})|/|E|$.

Furthermore, let Y_k be the cumulative number of edges crossing the partition along the first k steps of the random walk. This is expressed nicely as $Y_k = \sum_{i=1}^{k} X_i$. Thus $E[Y_k] = k\frac{|E(S,\bar{S})|}{|E|}$.

Applying Markov's inequality, we get $\Pr[Y_k \geq tk|E(S, \bar{S})|/|E|] \leq 1/t$. Suppose we wish to examine under what conditions we can ensure that we do not cross over the partition at all in M steps, i.e. $\Pr[Y_M \geq 1] \leq 1/2$. From the inequality above, we are able to get that

$$\Pr\left[Y_M \geq 2M\frac{|E(S, \bar{S})|}{|E|}\right] \leq \frac{1}{2}$$

just by setting $k = M$ and $t = 2$. We then use the following basic fact: if we have an inequality of the form $\Pr[Z \geq z] \leq \frac{1}{2}$, then $\Pr[Z \geq z'] \leq \frac{1}{2}$ for any $z' \geq z$.

Let $E(S)$ denote the set of edges connected to any vertex in S. Because $|E(S)| \leq |E|$, then we have $|E(S, \bar{S})|/|E| \leq |E(S, \bar{S})|/|E(S)|$. Furthermore, since we assume a connected graph, $|E(S)| \geq (d/2)|S|$, and thus $|E(S, \bar{S})|/|E(S)| \leq |E(S, \bar{S})|/[(d/2)|S|]$. [1] Thus using the fact above we can deduce

$$\Pr\left[Y_M \geq 2M\frac{|E(S, \bar{S})|}{(d/2)|S|}\right] \leq \frac{1}{2}$$

Note that $|E(S, \bar{S})|/|S|$ is the conductance of the graph $\phi(G)$, because S was defined to be the sparsest-cut partition of G. Thus we can apply the fact again with Cheeger's inequality to get

$$\Pr\left[Y_M \geq 2M(2/d)\sqrt{2\lambda_2}\right] \leq \frac{1}{2}$$

[1] It is important to note that this specific dependency of $|E(S)|$ on d requires G to be a d-regular graph. If the theorem is to be expanded to more general cases, one may use the simple inequality $|E(S)| \geq |S|$.

And since we are interested in $\Pr[Y_M \geq 1]$, we can thus set $2M\sqrt{2\lambda_2} \leq 1$ to get a necessary condition for M, from which we achieve

$$M \leq \frac{d}{2^{5/2}\sqrt{\lambda_2}}$$

This completes the proof. □

Corollary 11.1.1. *If we want to bound the probability by δ instead of $1/2$, then we have the restriction $M \leq (\delta d)/2^{3/2}\sqrt{\lambda_2}$.*

Proof. Follow the proof of Theorem 11.1, but with $t = \frac{1}{\delta}$ instead of $t = 2$. □

11.2.1 More general case

A quite restricting condition of Theorem 11.1 is that it requires the partition S to be a sparsest cut, namely that it is the partition with the least conductance $\phi(S)$. A very slight modification of the proof yields a quantity that can work for any signal f.

Let f be a boolean (i.e., $\{0,1\}$-valued) signal on the vertices of the graph.

Let the **signal-partition** $S = \{v \in V(G) : f(v) = 1\}$.

We are interested in the probability that a random walk of length M includes an edge that crosses the signal-partition S.

Claim 11.1. *Let f be a boolean signal on the nodes of G with signal-partition S. Let $u_1, u_2, \ldots$ be a random walk on the large graph G. Analogous to Theorem 11.1, let X_i be the indicator event that the ith edge of the random walk is an edge crossing the signal partition, i.e., $X_i = \mathbf{1}\left[(u_i, u_{i+1}) \in E(S, \bar{S})\right]$. Also analogous to Theorem 11.1, let $Y_k = \sum_{i=1}^{k} X_k$ be the number of times that we cross the signal-partition of f in steps of the random walk.*

Let $f' = f - \mathbf{1}(|S|/|V|)$, and let $\alpha = f'^{\top} L f' / \|f'\|_2^2$.

Then $Pr[Y_M \geq 1] \leq \frac{1}{2}$ if $M \leq \frac{d}{2^{5/2}\sqrt{\alpha}}$.

Proof. The quantity f' is a transformation of f that retains all the information contained in f while still being orthogonal to the all-ones vector $\mathbf{1}$, so that we can apply Cheeger's inequality. This orthogonalization is rather standard and can be found in [22].

Let $s = |S|/|V(G)|$. Note that $s \in [0,1]$, and without loss of generality we can assume that $s \leq 1/2$.

We observe that the vth coordinate of the vector f' corresponds to the mapping

$$f'(v) = \begin{cases} 1-s & v \in S \\ -s & v \notin S \end{cases} \tag{11.1}$$

This ensures that f' is orthogonal to $\mathbf{1}$, as

$$f'^{\top}\mathbf{1} = \sum_{i=1}^{n} f'(v_i) = |S|\left(1 - \frac{|S|}{|V|}\right) + (|V| - |S|)\left(-\frac{|S|}{|V|}\right) = |S| - |V|\left(\frac{|S|}{|V|}\right) = 0.$$

We then note that $\|f'\|_2^2 = \sum_{i=1}^{n} f(v)^2$ is equal to $s(1-s)|V|$, and we can infer $|S|/2 \leq \|f'\|_2^2 \leq |S|$; the first inequality holds since $s \leq 1/2$.

The number of edges $|E(S,\bar{S})|$ crossing the signal-partition is equal to $f'^{\top}Lf'$, as

$$f'^{\top}Lf' = \sum_{(u,v)\in E} ((f(u) - s) - (f(v) - s))^2 = |E(S,\bar{S})|$$

where L is the Laplacian matrix of $\mathcal{G}$.

Thus the quantity $2M\frac{|E(S,\bar{S})|}{|E(S)|} \leq 2M\frac{f'^{\top}Lf'}{|E(S)|} \leq 2M\frac{f'^{\top}Lf'}{(d/2)|S|}$. We are able to get the second inequality because we know $|E(S)| \geq (d/2)|S|$. Because we know that $|S| \geq \|f'\|^2$, we can then upper bound this further by $2M\frac{f'^{T}Lf'}{(d/2)\|f'\|_2^2}$. Substituting this quantity in the proof of Theorem 11.1, we achieve the desired bound for M. □

Corollary 11.1.2. *Similar to Corollary 11.1.1, if instead of probability* $1/2$ *we want to bound the probability by* δ*, we have* $Pr[Y_M \geq 1] \leq \delta$ *if* $M \leq (\delta d)/2^{3/2}\sqrt{\alpha}$*.*

11.3 Overall prediction error

Let F be the event that the graph generated by the M-random walk, G_M, does **not** lie completely within a single signal-partition. We will first analyze the probability of various methods giving a correct judgement, given that the event F does not take place.

If all of the nodes of M are within a single signal-partition, then all of their labels are the same. Therefore if ϕ is a model trained on N-random walks of G, then one can just sample a single connected N-subgraph G' from G_M via random walk (or any other method), feed it into ϕ, and return the output of ϕ as the answer for all of G_M.

The probability that ϕ returns a wrong answer is exactly that of ϕ returning a wrong answer on G'. In our PAC-Bayes framework, ϕ is actually analyzed as being a member of a distribution Q of models, so the probability of a wrong judgement is $\mathbb{E}_{\phi \sim Q}[L_{D_0}[\phi]] = L_{D_0}[Q]$.

With this quantity, we are then able to prove a bound on the overall prediction error of this procedure.

Theorem 11.2 (Overall prediction error)**.** *For any $\delta \in [0, 1)$, if we restrict M to the condition in Corollary 11.1.1, that $M \leq (\delta d)/2^{3/2}\sqrt{\alpha}$ then the overall probability of error of each of the above errors, denoted as $Pr[F]$, satisfies*

$$Pr[F] \leq \delta + L_{D_0}[Q] \tag{11.2}$$

Proof. We make use of the fact that, for any random events F and E:

$$\Pr[F] \leq \Pr[E] + \Pr\left[F|\bar{E}\right] \tag{11.3}$$

Let E be the event that the M-random walk that generates the graph G_M is crosses the sparsest-cut partition, and let F be the overall event that we make a wrong prediction. The theorem in the previous section is the second term, $\Pr\left[F|\bar{E}\right]$, because it examines the case where G_M doesn't cross the partition.

Substituting the values from the previous two theorems we find that the claimed inequality is satisfied instantly.

□

11.4 Discussion

This section has laid out a theoretical framework for understanding certain problems in size generalization. There are a number of future directions that can be pursued to expand this theory.

- Firstly, this framework for size generalization assumes the property that nodes that are connected to each other are likely to be labelled the same. This does not cover all of the cases in which GNNs might work well. As a trivial example, consider a case where for all of $\mathcal{G}$ users are necessarily connected only to users with the opposite labelling - a trained GNN would still work well in predicting the labels in this setting, as it is trivial to see that setting neighbor weights so that the model mimics the behavior of a NOT gate would be able to perform the task perfectly. In order to work towards the overall goal of seeing how GNNs are able to perform size generalization, these cases must be considered as well.

- The theorems in this section give probability bounds for size generalization under a setup that assumes certain properties about the underlying graph $\mathcal{G}$ as well as the ground-truth labelling function f. Difficulties occur in analysis when $\mathcal{G}$ is very large. If the investigator is fortunate enough to have access to the entire graph $\mathcal{G}$, while computing quantities like λ_2 or $f^\top Lf$ is generally costly one may apply algorithms such as the Lanczos iteration [23] for more efficient computation of eigenvalues. If the investigator does not have access to all of $\mathcal{G}$, these properties may not be computable, either because it would be costly or because the investigator simply cannot access all of the data. In particular, since $\mathcal{G}$ is so big (our canonical example is the graph of users of a billion+-user social network), it makes it infeasible to obtain exact values for the space of sparsest cuts because that would involve computing $\lambda_2(\mathcal{G})$; likewise for more general signals, it is also costly to calculate $f^\top Lf/\|f\|_2^2$ for the whole graph (because f has dimension equivalent to the number of nodes).

- For the sake of ease of analysis, we have made the assumption throughout the setup that the graph is d-regular. Generally for results in spectral graph theory, similar assumptions can be made to prove theorems and then later extended to the non-d-regular case. In particular, we have used the d-regularity assumption in the calculation of inequalities regarding Cheeger's inequality as well as our framework that assumes that the stationary distribution is uniform over the nodes. The stationary distribution assumption was integral in our calculation of the random walk-based quantities.

Chapter 12

Conclusion

Graph neural networks have been an important development in machine learning, particularly for applications related to social network analysis as well as computational chemistry, with still more to follow. In the context of this work, graph neural networks also represent a class of model where the speed of development of new types of model is much faster than how well we can theoretically analyze their efficacy. A canonical example of this is the original GCN paper by Kipf and Welling [24], which gave birth to one of the most popular forms of GNN without theoretical justification.

The developments in this work are an attempt to further the theoretical understanding of the efficacy of this model. Although the generalization bounds in Section **??** are an advancement over the state-of-the-art, the actual numerical values for the generalization gaps found on "canonical" graph learning tasks are still in exponential orders of magnitude, despite being meant to compare probabilities, which are necessarily less than 1. Despite the impracticality of using these bounds for practitioners at the present moment, furthering our understanding of what we can prove about neural networks *a priori* could be of fundamental importance in the future. If, with perhaps more specific analysis of datasets, mathematical ingenuity, or a combination of both, we are able to reduce the bounds enough to get them within probabilistic range (in $[0, 1]$) or even further, then it would be of immediate practical usefulness. Before starting an expensive training process, a practitioner could check the theoretical bounds or other provable quantities about the task to aid in selecting the class of model or its hyperparameters. Such a step would become particularly important in situations where computational resources are more scarce, but training new machine learning models is still necessary.

Understanding size generalization would also be an important step in expanding theoretical knowledge for the sake of reducing overall computational cost. In this analysis, what was analyzed was the generalization from small to large subgraphs of a large underlying graph $\mathcal{G}$. By identifying cases where it is practical to train on small subgraphs instead of large ones, we potentially ease the burden of expensive data collection. Expanding the cases to which theoretical analysis can apply is imperative to achieve this easing across different graph learning tasks.

It is the author's hope that this work could be a contribution, starting point, or even mere inspiration for expanding the theoretical analysis of graph neural networks and neural networks in general, as these models continue to maintain their presence in everyday life.

Bibliography

[1] R. Liao, R. Urtasun, and R. Zemel, "A pac-bayesian approach to generalization bounds for graph neural networks," *arXiv preprint arXiv:2012.07690*, 2020.

[2] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms.* Cambridge university press, 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[4] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, (New York, NY, USA), 2016.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[6] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159.

[7] G. V. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, pp. 303–314, 1989.

[8] L. G. Valiant, “A theory of the learnable,” *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.

[9] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, “Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks,” *Journal of Machine Learning Research*, vol. 20, no. 63, pp. 1–17, 2019.

[10] D. McAllester, “Simplified PAC-Bayesian margin bounds,” in *Learning theory and Kernel machines*, pp. 203–215, Springer, 2003.

[11] B. Guedj, “A primer on PAC-Bayesian learning,” *arXiv preprint arXiv:1901.05353*, 2019.

[12] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, “A PAC-Bayesian approach to spectrally-normalized margin bounds for neural networks,” *CoRR*, vol. abs/1707.09564, 2017.

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” *arXiv preprint arXiv:1810.00826*, 2018.

[14] Z. Chen, L. Chen, S. Villar, and J. Bruna, “Can graph neural networks count substructures?,” *Advances in neural information processing systems*, vol. 33, pp. 10383–10395, 2020.

[15] K. Xu, J. Li, M. Zhang, S. S. Du, K. ichi Kawarabayashi, and S. Jegelka, “What can neural networks reason about?,” 2020.

[16] P. Yanardag and S. Vishwanathan, “Deep graph kernels,” in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

[17] F. Scarselli, A. C. Tsoi, and M. Hagenbuchner, “The Vapnik–Chervonenkis dimension of graph and recursive neural networks,” *Neural Networks*, vol. 108, pp. 248–259, 2018.

[18] V. Garg, S. Jegelka, and T. Jaakkola, "Generalization and representational limits of graph neural networks," in *International Conference on Machine Learning*, pp. 3419–3430, PMLR, 2020.

[19] R. Vershynin, *High-dimensional probability: An introduction with applications in data science*, vol. 47. Cambridge University Press, 2018.

[20] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, pp. 415–444, 2001.

[21] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bulletin of the American Mathematical Society*, vol. 43, no. 4, pp. 439–561, 2006.

[22] D. Spielman, "Spectral graph theory course notes."
`https://www.cs.yale.edu/homes/spielman/561/`.

[23] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, vol. 50. Siam, 1997.

[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.