COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

HABILITATION À DIRIGER DES RECHERCHES

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *AST*

Par
Nicolas KERIVEN

Graph Machine Learning and Graph Neural Networks on Large (random) Graphs

Thèse présentée et soutenue à Rennes, le 2 juin 2026
Unité de recherche : IRISA

Composition du Jury :

Sophie	ACHARD	Directrice de Recherche, CNRS, Rapporteuse
Marc	LELARGE	Directeur de Recherche, INRIA, Rapporteur
Pierre	VANDERGHEYNST	Professeur, Ecole Polytechnique Fédérale de Lausanne, Rapporteur
Florence	d'ALCHÉ-BUC	Professeure, Télécom-Paristech, Examinatrice
Rémi	FLAMARY	Professeur, École polytechnique, Examinateur
Élisa	FROMONT	Professeure, Université de Rennes, Examinatrice
Aurélien	GARIVIER	Professeur, École Normale Supérieure de Lyon, Examinateur

I want AI to do my laundry and dishes so that I can do art and writing, not for AI to do my art and writing so that I can do my laundry and dishes.

Joanna Maciejewska

Not a word of this manuscript was written by a Large Language Model. Ethical considerations aside, if I like writing and hate proofreading, why would I automate the first to do only the second?

Acknowledgement

Since the very beginning of my research journey, I have been privileged and blessed in many ways: with great supervisors, great collaborators, great research environments where money was not scarce, and finally great students. I realize that in all those years my path has been, for all intents and purposes, rather straightforward and "easy", and I am all the more grateful for it.

I first want to thank my PhD and postdoc supervisors, R. Gribonval and G. Peyré. Although it's been a few years, I am continuously in debt to and eternally grateful for everything that they brought me.

I want to thank all my collaborators, past and present (and future): S. Vaiter (for being my most consistent collaborator and friend over the years, although diametrally opposed on the map of France), S. Barthelmé and N. Tremblay (for welcoming me in Grenoble and being incredible researchers and great sources of inspiration), T. Maugey and A. Roumy (for welcoming me in Rennes and instantaneously making me feel at home), Y. de Castro, Damien Garreau and Gilles Blanchard (for teaching me what actual rigorous statistics look like), A. Bietti and T. Vayer (for teaching me what actual good ideas look like), C. Poon (for being the most incredible mathematician that I have had the chance to work with), Iacopo Poli (since I am worthless as an engineer), A. Bourrier and Y. Traonmilin (for helping me in my very first years as a young PhD student) and A. Deleforge, A. Liutkus, K. O'Hanlon, M. Plumbley, from a lifetime ago where I was doing sound and music processing.

I want to thank all the students that I had the chance to co-supervise over the years: H. Ghanem, M. Cordonnier, M. Gjorgjevski, A. Joly, A. Jamadandi, C. Mazini-Rodrigues, H. Jaquard, Y. Malot, M. Theveneau, R. Léger, Y. Viegas, K. Rawat, A. Coulais, C. Fantasio. Supervising people is about the most formative, challenging, rewarding aspect of the job, and I've learned more from them than they know.

I want to thank the reviewers and jury of this HdR for taking the time and doing me the honor of reading and hearing about my work.

I want to thank all the administrative staff at Gipsa-lab and IRISA, they are the life and blood of the labs. Special thanks to C. Tanguy, who I'm gonna continue to challenge with administrative nightmares that she will solve out of thin air.

I want to thank the CNRS. Pestering host institutions and everything government-related is France's national sport, but I realize how lucky I am to have had a permanent position two years after my PhD, where I am fully free to conduct the research that I want, and how exceptional this is in the international research landscape. Such unique positions are a treasure to be cherished and *preserved*.

Finally, I want to thank all my friends and family. My parents, brothers and sister for nurturing my taste for science from the very beginning. Noémie for being my musical partner in crime and lifelong friend. Aude and Lowen for being my world, to which everything else pales in comparison after all.

NOTATIONS

$\mathcal{X}$	set of samples/latent variables x_i. Assumed included in $\mathbb{R}^{d_x}$ and generally compact.
$\mathcal{Y}$	Set of labels: discrete for classification, continuous for regression.
Y	set of predictions $\hat{y}$, that may be different from $\mathcal{Y}$ or not. Generally included in $\mathbb{R}^{d_y}$. Equipped with a metric d_{Y}.
$\mathcal{Z}$	set of node features z_i. Assumed included in $\mathbb{R}^{d_z}$ and generally compact.
$\mathfrak{G} = \mathfrak{G}(\mathcal{Z})$	set of all graphs $G = (A, Z)$, with weighted adjacency A and node features Z in $\mathcal{Z}$.
$\mathfrak{G}_n \subset \mathfrak{G}$	graphs of size n
$\left\|\cdot\right\|_q$	norm $\left\|v\right\|_q = (\sum_i v_i^q)^{1/q}$ for a vector, or $\left\|S\right\|_q = \sup_v \left\|Sv\right\|_q / \left\|v\right\|_q$ for a matrix. Sometimes shortened $\left\|\cdot\right\| = \left\|\cdot\right\|_2$
$\left\|\cdot\right\|_{q,2,n}$	applied to a matrix $X \in \mathbb{R}^{n\times d}$, equal to $\left\|X\right\|_{q,2,n} = \left(\frac{1}{n}\sum_i \left\|X_{i,:}\right\|^q\right)^{1/q}$. When $q = \infty$, $\left\|X\right\|_{\infty,2} = \sup_i \left\|X_{i,:}\right\|$.
$\mathcal{F}_{all}$	Space of all "classical" prediction functions $f : \mathcal{X} \to \mathsf{Y}$. Equipped with $d_{Y,\infty}(f, g) = \sup_x d_{\mathsf{Y}}(f(x), g(x))$.
$\mathcal{B}(\mathcal{X})$	bounded functions $f : \mathcal{X} \to \mathbb{R}$, equipped with the norm $\left\|f\right\|_\infty = \sup_x \left\|f(x)\right\|$. For multivariate functions $f \in \mathcal{B}(\mathcal{X})^p$, the norm is $\left\|f\right\|_\infty = \sup_x \left\|f(x)\right\|_2$
$\mathcal{F} \subset \mathcal{F}_{all}$	Set of function in practice, e.g. neural nets.
$\iota_X f$	sampling operator $\iota_X f = [f(x_i)]_{i=1}^n \in \mathbb{R}^{n\times d}$
$\mathcal{H}_{all}$	Space of all node-level prediction functions, applicable to all graphs and permutation-equivariant. Equipped with a family of metrics $d_{\mathsf{Y},n}(h, h') = \sup_{A,Z} \max_i d_{oYy}(h(A, Z)_i, h'(A, Z)_i)$.
$\mathcal{H}_{all,\infty} \subset \mathcal{H}_{all}$	functions whose expected risk over random graphs of size n converges. Depends on the random graph model.
$\mathcal{H} \subset \mathcal{H}_{all}$	node-tasks prediction functions in practice, e.g. GNNs. Ideally subset of $\mathcal{H}_{all,\infty}$, but must be proven.
ℓ	loss function $\ell : \mathsf{Y} \times \mathcal{Y} \to \mathbb{R}_+$
$\hat{\mathcal{R}}_{\mathrm{ML}}$	empirical risk in classical ML

$\mathcal{R}_{\mathrm{ML}}$	expected risk in classical ML
$f^{\star}_{\mathrm{ML}}$	Bayes predictor
$f_{\mathcal{F}}$	best function among $\mathcal{F}$
$\hat{\mathcal{R}}$	empirical risk in graph ML
$\mathcal{R}_n$	expected risk over graphs of size n
$\mathcal{R}_\infty$	limit risk for functions in $\mathcal{H}_{all,\infty}$
$h^\star$	best graph prediction function (minimizes $\mathcal{R}_\infty$ over $\mathcal{H}_{all,\infty}$)
$h_{\mathcal{H}}$	best graph prediction function among $\mathcal{H}$
$W_n(x,x')$	connectivity kernel for Latent Position Models (LPMs). Generally $W_n(\cdot) = \alpha_n w(\cdot)$.
$\mathfrak{P}$	a family of graph distributions $\mathfrak{P} = \{P_n\}_{n\geqslant 1}$ for each graph size n. Generally a LPM
$\mathcal{F}_{\mathcal{H}} \subset \mathcal{F}_{all}$	for "$\mathfrak{P}$-convergent" set of functions on graphs $\mathcal{H} \subset \mathcal{H}_{all}$, set of limit functions
S	graph representation matrix, generally $S = A/(n\alpha_n)$, $S = D^{-1/2}AD^{-1/2}$ or $S = D^{-1}A$.
$Z^{(0)} = Z^{(0)}(A,Z)$	Positional Encodings (PE) or node features
ρ_k	activation function at layer k
Φ_θ	GNN parametered by θ, generally a Message-Passing GNN
Φ^{c}_θ	"continuous" GNN, towards which Φ_θ is generally $\mathfrak{P}$-convergent.

Table of Contents

A WORD AND ORGANISATION OF THE MANUSCRIPT

For the reader not familiar with the French *Habilitation à diriger des recherches*, it is the "last" academic diploma in France. It is in particular required to supervise PhD students (although having *co*-supervised PhD students beforehand is a requirement to obtain it!). The associated manuscript – this document – is often thought to be a "second" PhD thesis, although somewhat more free in content and form. The candidate is more or less free to present whatever they want, from a succinct presentation of their past and current work to a quasi-book with complete introduction to their field.

This manuscript presents a selection of my works of the last few years in the field of **Graph Machine Learning** (Graph-ML, or GML). Compared to the original papers, it includes quite a bit of reformulation, harmonization, and consequently some new material (Chapter 2).

Although I deal with graphs routinely, I do not consider myself a specialist of "graph theory", or even of "network (data) science", as it is sometimes referred to in data science. Like many researchers in Graph-ML, I have a background in *Signal Processing and Machine Learning*: before 2019, when I started to take an interest in graphs, I worked on dimensionality reduction in machine learning and signal processing (I am a product of the "compressed sensing" school). I think that this background is significantly reflected in my work [1]. Over time, my research also shifted more and more towards *theoretical* contributions, although I collaborate with students and colleagues that are more on the practical side. As a consequence, this manuscript will be mostly theoretical in nature, and only includes a few numerical illustrations extracted from their original papers.

It can be said that in the current field of Graph-ML, there are roughly two main approaches. The first deals with graphs using *discrete* and *combinatorial* reasonings, and is more rooted in computer science and classical graph theory. It allows to prove "strong" properties that are exact and often deterministic: e.g., a particular Graph Neural Network [2] (GNN) is (theoretically) able

1. to an extent, this is also true for most of the Graph-ML community, in which researchers have in background in ML rather than graph theory.
2. a deep neural network on graphs, which we will extensively introduce in the rest of the manuscript

to emulate this or that classical graph algorithm, produce a particular graph coloring, and so on. The second approach, rooted in *statistical Machine Learning*, would rather treat graphs as random data, and derive probabilistic results: e.g., a GNN will succeed with high probability in identifying approximate communities in a Stochastic Block Model (SBM), and so on. Of course, these two points of view have always co-existed long before GML and are naturally porous, such that many results could be said to belong to both worlds: it is a continuum rather than a black-and-white situation.

Nevertheless, I am definitely part of the second realm. Like many researchers with my background, I love dealing with probabilities, statistics, concentration inequalities... as well as *continuous* objects like functions, which is a paradox when working with graphs, but also a source of originality that is a core part of my work. On the other hand, I find myself painfully aware of my limitations with regards to discrete mathematics. A large part of my research has therefore been to use the age-old trick of taking the limit "from-discrete-to-continuous", generally by making various quantities increase to infinity and having some kind of process to describe the growth (here, the number of nodes in a *random* graph). In the continuous world, one gets rid of pesky combinatorial difficulties and awful intricated sums for the familiar world of Lipschitz functions, integrals, and so on.

As it turned out, this approach – combining modern GML objects such as GNNs with random graphs – was quite absent from the GML landscape when I started studying it in 2020. All the required mathematical notions were lying around and were not new by any means, but GML and GNNs were young enough such that nobody had really taken a point of view grounded in *statistical* ML on it. We were a few authors (now colleague and friends) to introduce this approach approximately at the same time in different parts of the world (namely, Germany [84], the US [130], and my co-authors and I in France [NK14]). Now, the use of random graphs in GML is (slightly) more common, but is still riddled with fascinating open questions and remains at the core of my research [3].

This manuscript. In light of this, I elected to present a quite narrow batch of papers in this manuscript, focused on random graphs in machine learning, and assemble them into (what I hope to be) a coherent whole – except for the last chapter, which contains a standalone contribution. This is not the most common approach for an *Habilitation* manuscript, which usually presents a diverse selection of works. Indeed, it ignores several projects that I have worked on in Graph ML: on graph coarsening [NK6, NK7], community detection [NK12], optimal transport on graphs [NK8, NK16], determinantal sampling [NK5], kernel methods for random graphs [NK3, NK4], among others. Nevertheless, it is by far my "main" line of work, and gathering these papers into

3. 6 years after this first paper, random graphs in ML are still the main topic of my recent ERC StG grant (MALAGA), and should therefore remain my main object of interest for the next 5 years.

a coherent whole was an appealing project in itself. In particular, Chapter 2 presents a general framework tying many of my previous results together, new and still unpublished: it was only by attempting to give a unifying view in this manuscript that I ended up introducing new tools for this chapter. In details, the manuscript is organized as follows.

1. In Chapter 1, I start with preliminary material on graphs, Graph-ML, and Graph Neural Networks. Of course, each of these topic would deserve a whole book, and I necessarily give only a brief and narrow overview. I focus on material useful to understand this manuscript, strongly biased towards my personal preferences.
2. Chapter 2 contains material that is as of yet unpublished [4]. Starting from a brief introduction to statistical ML, I develop a generic framework to analyze ML algorithms that perform prediction on the *nodes* of graphs, with an emphasis on the large-graph limit. I then introduce a family of random graph models called *Latent Position Models*, which encompass most random graph models developed in the literature [5]. Combined with the new GML framework, they yield *novel connections* between GML and regular ML, and conditions to perform a full analysis of the generalization error. This was in fact the motivation to develop such a framework, and a consequence of writing this manuscript.
3. In Chapter 3, I implement all the results of the previous chapter for various Graph Neural Networks, thereby showing complete results on their generalization for node prediction. It is a gathering of results from several papers [NK14, NK15, NK13, NK1], re-casted in light of Chapter 2.
4. Chapter 4 presents an independent contribution, on the matter of GNN *oversmoothing*. It is a fascinating phenomenon specific to GNNs that severely limits their depth. Oversmoothing has remained an interesting "side-project" for me, and this chapter contains the results of two single-author publications on the topic. This chapter is presented as a "bonus" chapter, even though notions from Chapter 2 will be evoked and it is not entirely independent from the rest of the manuscript. It can still be largely read independently.

All along the manuscript, several technical proofs will be given, when they concern novel results, or when the given version of the result departs too far from its formulation in the original paper. They are situated in the Appendix of each chapter, which can be safely skipped by the reader if wanted. Otherwise, all technical proofs of results from my papers can be found in the original version. All my paper are available on open-access repositories, whose links can be found on my website `https://nkeriven.github.io`. In the manuscript, references to papers that I co-authored are indicated by [NK·]. Whenever appropriate, I try to use the pronoun "I" when

4. as such, it hasn't been peer-reviewed. In particular, I am not sure if it constitutes a "contribution" in the classical sense of the term, or just an interesting rewriting and gathering of results.

5. virtually all models except for *preferential attachment* random graphs [9] and their variants

referring to personal thoughts for this manuscript or (rare) single-author works, and default to the usual "we" for any collaborative work, in the majority of the case. A complete table of notations can be found at the beginning of the manuscript.

Chapter 1

Introduction

Summary

This is an introductory chapter, organized around three levels of specificity. In a nutshell:

— **Graphs** are both very real and purely mathematical objects representing networks of interconnected objects. *Graph theory* is strongly multi-faceted, depending on one's background and preferences.

— **Graph Machine Learning** (GML) comes in several flavors: performing predictions at the graph-level, node-level, edge-level... This manuscript will focus on the node-level. In particular, it deviates from the "independent data" paradigm at the foundation of ML theory. *Permutation-invariance* is the main constraint of GML algorithms.

— **Graph Neural Networks** (GNNs) are deep networks acting on graphs. They may *appear* to come in many flavors, but most of them perform message-passing on the edges of the graph. They are extremely flexible and permutation-invariant by construction, but have limitations: limited expressivity, difficulty in dealing with heterophilic graphs, difficulty with increasing depth...

Contents

In this first chapter, I try to give a brief introduction to graphs, graph Machine Learning, and Graph Neural Networks. As will often be the case in this manuscript, I will be quite straight-to-the-point, with no attempt to be exhaustive whatsoever – each of these topics deserves an entire book, and there are certainly many [35, 21].

1.1 Graph...

In this section, I introduce "graphs" and graph "data science", with a few real-world examples. I then describe some useful mathematical notions around graphs.

1.1.1 Graphs in the wild

Graphs are very general objects that appear in virtually every field of science. Each discipline has its own conventions in terms of vocabulary, mathematical definition or informal description. Loosely speaking, a graph is an object representing several elements (the *nodes*) with interconnections between them (the *edges*).

In the real world, graphs are often conflated with the notion of *network*. One interesting point of view [35] is to reserve the term "network" for the real-world, complex system under observation, and to call "graph" the *mathematical representation* of it, among many possible representations at different levels. In particular, a graph must often be *constructed* from a real-world network, which generally requires experts in the respective field.

Every graph everywhere all at once. Since they can be used to describe any complex system, the notion that "graphs are ubiquitous" has been repeated so often in introductory sections of various works that it has become somewhat of an inside joke between researchers in the field. That is not to say that it is not true! Interestingly, the (relatively) recent adoption of Machine Learning (ML) by many scientific fields has resulted in a rapid gathering of graph datasets from a large variety of domains, in open-access repositories such as the Open Graph Benchmark [70] or the Pytorch Geometric library [52]. Crucially, having easy access to many different graphs in one place offers novel insights into the *differences* and *similarities* of networks across scientific fields, an aspect that might significantly gain in importance in the future.

Some examples. Let us now quickly mention a few example of graphs, particularly of graphs that are common in modern data science, often due to data scientists' biases concerning the relative "importance" of their field of application, or simply due to the availability of popular open-access datasets. See Fig. 1.1.

Physical networks certainly represent the most immediate and early historical graphs: it is often said that graph theory was born from Euler's famous problem of the "Seven Bridges of Königsberg", which involve a small example of a *road network*. Nowadays, analyzing such networks is useful e.g. for traffic control and prediction [74], time-of-arrival estimation, etc. Our day-to-day life includes many other physical networks such as the electrical grid [64], computer networks [122], urban sewer systems [158]... among many others. In these networks, the edges are obvious since they represent physical connections.

Social networks [65, 71] modelling interactions between humans or other social entities have been a central object in social science and the humanities for a long time, dating back to the early 20th century [139]. In general, the edges in a social network may be less obvious to observe: it represents the "strength" of a connection or interaction, but interactions are limited in time, of very different nature, and so on.

Nowadays, social networks are of course associated with the *Internet*, which certainly has been the main source of popular graphs and datasets in recent data science [118]. The Internet is a physical network of interconnected computers, but also contains a myriad of other networks: connected webpages [107], forums [170]... by definition, connections between various entities on the Internet are numerous, well-defined, and potentially easily accessible, hence their popularity in data science.

Biological networks come from all fields of biology: gene-interaction networks [98, 106, 124] represent their co-expressivity or their regulatory roles, protein-protein interaction networks [138] represent the various ways in which proteins interact with each other, drug/drug interaction network [70] represent the interactions between different drugs, and so on. In neuroscience, the brain itself is often seen as a network [50], either between individual neurones or regions. Generally, in biology, edges are never directly observed, and must be *inferred* from (very) indirect and noisy measurements.

In *chemistry*, graphs have been extensively used to represent molecules [44, 58] as networks of connected atomes, with very popular applications in the ML community such as molecular properties prediction [58], drug discovery [25, 73], protein interface prediction [53] or various applications in material science [54].

Information networks have naturally become a central notion in the so-called information age. A very general concept, they may represent any kind of link between "elements of information". For instance, *knowledge graphs* [72] represent relations between semantic elements with application in linguistics and, naturally, artificial intelligence [105]. *Citation networks* [57], which represent academic papers with citations between them, have a historical importance: since they come from data that was readily available within the academic world itself, they have long been among the

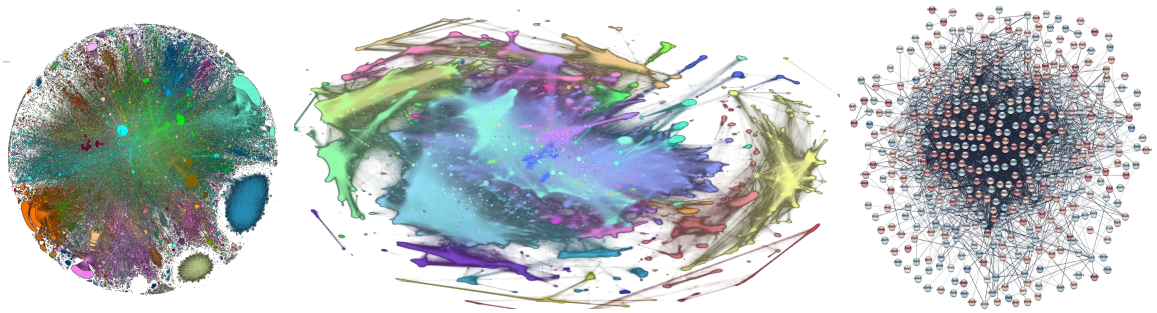


Figure 1.1 – Examples of visualization of graphs. From left to right: a network or Reddit posts from `https://github.com/wojciech-graj/reddit-graph`, a network of Wikipedia pages, from `https://www.youtube.com/@adumb_codes`, a network of protein associations, from [42].

most popular "large-scale" (at the time) open-access datasets. The famous triad "Cora, Citeseer, Pubmed" datasets [163] remain the most basic benchmark in graph data science.

Of course, graphs and networks can be found in many other applications. *Ecological networks* such as food web [24] are crucial to model complex relations in the natural world. *Epidemiological networks* [69, 120] serve to study and anticipate the spread of diseases in a given population. *Physics*, and particularly statistical physics, is rich with myriads of systems of interacting elements [132]. In *combinatorial optimization* [26], graphs help modelling complex, discrete relations between variables... and so on and so on.

All these graphs exhibit many different characteristics. Personally, I consider that a better understanding of the *common traits and differences* of graphs coming from many different fields will gain in importance in graph data science and graph ML. It goes without saying that molecules with a few dozen nodes have little in common with internet graphs with millions of nodes. Generally, the former can be highly "structured", with key discrete motifs formed by a few nodes, while the latter is strongly "stochastic" in nature, and might be better analyzed at a macro scale. In this manuscript, we will encounter various other characteristics of graphs (sparsity, homophily, community structure...) that are sometimes more specific to different fields of applications.

1.1.2 Graphs: a few mathematical tools

Let us now turn to a few mathematical notions related to graphs.

A **graph** G is an abstract mathematical object formed by a finite, discrete set of **nodes** V, with pairwise connections between them $E \subset V \times V$, the **edges**. Additionally, nodes and edges can be "decorated" with various quantities, features, labels, weights, and so on.

The most common representation of the structure of a graph with n nodes is through its *adjacency* matrix $A \in \mathbb{R}^{n \times n}$. **Given an ordering of the nodes from** 1 **to** n, in its most classical

version the matrix A contains a 1 at position (i, j) whenever the edge (i, j) is in the graph, or 0 otherwise:

$$A_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

More generally, A can be a matrix in $\mathbb{R}^{n \times n}$ that contains edge *weights* as scalars, in which case 0 (the absence of edge) is somewhat an arbitrarily chosen value. When A is symmetric, the graph is said to be *undirected*, and directed otherwise. In this manuscript, we will mostly consider undirected graphs, as symmetric matrices are infinitely more convenient to manipulate. The *degree* of a node is the number of edges attached to it: $d_i = \sum_{i=1}^{n} A_{ij}$. For directed graphs, one may distinguish the in-degrees and the out-degrees by summing either the ith row or the ith column of A, but these are equal for undirected graphs.

Graph signal and node features. A vector $z \in \mathbb{R}^n$ associating a scalar to each node of the graph will be called a *graph signal*[1]. More generally, a matrix $Z \in \mathbb{R}^{n \times d_z}$ contains *node features* of dimension d_z as rows. Both terms are important in their respective literature, namely Graph Signal Processing (GSP) [143, 41] and Graph ML [21], but node features are nothing else than multi-dimensional graph signals and graph signals are uni-dimensional node features. Both somewhat carry the same intuitions, if not the same set of terminologies (typically, graph signals are *filtered* while node features are *aggregated*).

Graph Laplacian and filtering. Besides A, several matrices play important roles in graph processing [32]. The first one is without doubt the *combinatorial Laplacian*:

$$L_{comb} = D - A \tag{1.2}$$

where $D = D(A) = \text{diag}(A1_n)$ is the diagonal matrix of the degrees. The Laplacian is positive semi-definite (p.s.d.), since for a graph signal z

$$z^\top L_{comb} z = \sum_{(i,j) \in E} (z_i - z_j)^2 \tag{1.3}$$

Interestingly, this means that this quantity measures the *smoothness* of a graph signal along the edges of the graph.

Graph Laplacians come in different flavors. One issue of L_{comb} is that its eigenvalues are not bounded with respect to the graph size n, hence most variants "normalize" the Laplacian to

1. we use z instead of the more classical x, which will be associated with latent variables in later chapters

respect this property. The two most common variants are certainly

$$L_{sym} = Id - D^{-1/2}AD^{-1/2}, \quad L_{rw} = Id - D^{-1}A \tag{1.4}$$

with the convention that $D_i^{-1} = 0$ when [2] $d_i = 0$. These are generally referred to as the symmetric normalized Laplacian and random walk Laplacian, respectively. The matrix L_{sym} is symmetric and has its eigenvalues in $[0, 2]$ for *all* graphs. It is the variant most commmonly studied in GSP [32]. The latter gets its name from the fact that $D^{-1}A$ is the transition matrix of a random walk on the graph with equiprobability on all neighbors at each node. It is diagonalizable with the same eigenvalues as L_{sym} when there is no isolated nodes, since $L_{rw} = D^{-1/2}L_{sym}D^{1/2}$.

In the field of GSP, the eigenvalues and eigenvectors of a graph Laplacian L (often L_{sym}) are generally interpreted as Fourier frequencies and modes. Decomposing $L_{sym} = U\Lambda U^\top$ the Fourier transform and inverse Fourier transform are respectively the operations $\tilde{z} = U^\top z$ and $z = U\tilde{z}$. Filtering a signal if then done by applying a function to the frequencies:

$$z_h = \varphi(L)z = U\varphi(\Lambda)U^\top z \tag{1.5}$$

where $h : \mathbb{R} \to \mathbb{R}$ is applied element-wise to each diagonal coefficient λ_i of Λ, and we denote $\varphi(L) = U\varphi(\Lambda)U^\top$ for simplicity. For instance, a low-pass filter can be obtained by $\varphi(\lambda) = 1_{\lambda \leqslant \tau}$ for some threshold τ. Note that U and Λ depend on the underlying graph G, but that a filter φ can be applied to any graph. This is crucial in ML, where one may want to *learn* some operations on a training graph and apply them to another graph. Of course, one can already see that normalized Laplacians are more appropriate for this, as the maximum range of their eigenvalues (among other properties) do not depend on the graph.

Graph isomorphism. It is important to remember that, for a given graph, all the objects above depend on a particular *ordering of the nodes*. For all intents and purposes, relabelling the nodes results in the "same" graph. More precisely, two graphs which are re-ordering of each other are said to be **isomorphic**: that is, two graphs represented by the adjacency matrices A and A' are isomorphic if there exists a permutation matrix [3] $\sigma \in \{0, 1\}^{n \times n}$ such that

$$A' = \sigma A \sigma^\top . \tag{1.6}$$

When the graphs are also equipped with node features Z, Z', they are isomorphic when both $A' = \sigma A \sigma^\top$ and $Z' = \sigma Z$ for some σ. In the rest of the manuscript, we will use a unique notation $\sigma \cdot A = \sigma A \sigma^\top$ and $\sigma \cdot Z = \sigma Z$, and for a graph $G = (A, Z)$, we define $\sigma \cdot G = (\sigma \cdot A, \sigma \cdot Z)$.

2. although we will generally consider that there are no isolated node for simplicity
3. that is, a square binary matrix with exactly one 1 on each line and column, and 0 elsewhere

Permutation-invariance is a central notion in graph theory, and *a fortiori* in GML. Indeed, algorithms applied to graphs must not depend on a particular representation of them, that is, they should be "consistent" with a re-labelling of the nodes. This is particularly crucial in machine learning, where predictions are expected to be consistent on new graphs that were not seen during training. We will discuss these notions in more details in the next section.

Notations. In the rest of the manuscript, a graph of size n will therefore be represented by $G = (A, Z)$ where $A \in \mathbb{R}_+^{n\times n}$ is its symmetric weighted adjacency matrix, and $Z \in \mathcal{Z} \subset \mathbb{R}^{n\times d_z}$ are node features of dimension d_z. We denote by $\mathfrak{G}_n = \mathfrak{G}_n(\mathcal{Z})$ the set of all graphs of size n with node features in $\mathcal{Z}$, where the dependency on the set $\mathcal{Z}$ will generally be omitted, and $\mathfrak{G} = \cup_{n\geqslant 1}\mathfrak{G}_n$ the set of all graphs. Remark that in this notation G and $\sigma \cdot G$ exist as two isomorphic representants of the "same" graph within $\mathfrak{G}_n$, that is, $\mathfrak{G}_n$ does not contains classes of equivalence or similar complex objects, but really only pairs of matrices (A, Z).

1.1.3 Random graphs and statistical graph models

In statistical learning, the data are generally assumed to be *random*, under various assumptions: independently and identically distributed (i.i.d.) for "classical" data, Markovian dependence for time series, and so on. We will refer to statistical models of graphs as *random graphs*[4]. They will be key objects in this manuscript.

Random graphs is a field that dates back to the famous Erdös-Rényi (ER) model [48], in which each edges appears independently with the same probability $p \in (0, 1)$. Although the ER model is exceedingly simple and serves more to illustrate various concepts rather than as an actual model for real-world data, it already showcases a quite stunning mathematical complexity, when one lets the number of nodes go to infinity $n \to \infty$ and the probability of connection depends on it $p = p_n$. The higher the p_n, the more "connected" the graph, and one can then distinguish several non-trivial regimes: from *dense* graphs $p_n = O(1)$ with $O(n^2)$ edges to *sparse* graphs $p_n = O(1/n)$ with $O(n)$ edges, and *relatively sparse* graphs in between. While these regimes can be found in many graph models, for the ER model the description can be pushed a lot further: for instance, for $p_n \leqslant (1-\varepsilon)/n$ for any $\varepsilon > 0$, all connected components in the graph have size $O(\log n)$ with high probability. For $p_n \geqslant (1+\varepsilon)/n$, there is a single "giant" connected component with size $O(n)$, all other components being of size $O(\log n)$, while for $p_n = 1/n$, there is a component with size $O(n^{2/3})$. For $p_n = O(\log n/n)$ and above, the graph is connected with high probability. The rate $\log n/n$ will be the limit above which our results will be valid.

While this mathematical richness has been instrumental in the developement of random graphs,

4. In some fields, the term "random graphs" only refers to the Erdös-Rényi model. Here we use it to indicate any statistical models of graphs.

the study of the ER model has remained somewhat theoretical in nature, being intuitively "too simple" to model real data. Early on, researchers identified characteristics of real-world graphs that could not be produced by ER graphs, such as the so-called power-law of the degrees (also referred to as *scale-free* property), which gave rise to the Babarasi-Albert [9] (or "preferential attachment" [5]) model. In modern graph data science, we often focus more on modelling other phenomena that are intuitively related to the *prediction* problems of interest: relationship between node features/graph structure and node labels, smoothness of various "signals" (labels or features) on the graph, and so on. For these purposes, a popular class of models is that of *Latent Position Models* (LPM) [65], in which various latent mechanisms are used to generate the graph structure. Typically, nodes are assumed to be equipped with unobserved *latent variables*, and for each pair of nodes, an edge is present with a probability that depends on the latent variables of the two nodes. This is a very general class of models that includes several models as particular cases: the ER model (where the probability of edges do not depend on the latent variables), but also the so-called Stochastic Block Models [66] (SBM, obtained with discrete latent variables that corresponds to community appartenance) that models communities of well-connected nodes, random geometric graphs [121], graphons [92]... Beyond this, LPMs are very convenient to incorporates additional observed quantities, e.g. node features or labels, and relate them to the observed graph structure through use of the same set of unobserved latent variables. In fact, as we will see in the next chapter it is possible to draw many connections between usual notions in Machine Learning and LPMs.

> *In my research and this manuscript, I work exclusively with Latent Position Models (LPM) [65]. They will be introduced in details in Chapter 2.*

1.2 Graph... Machine Learning

Graph *Machine Learning* (Graph ML, or GML) is a relatively recent field, even for the standard of ML. Historically, graph ML was very much influenced by the real-world *tasks* that data scientists were trying to solve on graphs. For instance, many approaches were developed for the problem of *community detection* [87]: that of grouping nodes in communities, generally in an unsupervised manner (without training labels). One of the most popular algorithm is *spectral clustering* [112].

It is only recently that graph ML has exploded, both in terms of number of applications, available datasets from many different field of science [70, 52], and variety of predictive tasks, in particular *supervised* tasks. Supervised graph ML can be broadly divided in two classes of tasks: *graph tasks*, in which one wants to predict a single quantity for the whole graph, and *node tasks*,

5. In this model, a graph is progressively grown by adding nodes, and a new node has a probability of connection proportional to the degrees of the nodes already in the graph.

	Graph task	**Node tasks**
Training data	i.i.d.	non-i.i.d.
Graph size n	small	high
Number of graphs N	high	small (even $N = 1$)
Constraint	$f(\sigma \cdot G) = f(G)$	$f(\sigma \cdot G) = \sigma \cdot f(G)$

Table 1.1 – Summary of the main differences between graph tasks and node/edge tasks. Edge tasks are for all intent and purposes similar to node tasks: non-iid data, a few large graphs, and permutation-equivariant functions. This manuscript will focus on node tasks.

in which one wants to predict a quantity for each node of one or several graphs. Other tasks such as edge prediction are close to node tasks in intuition. Although graph tasks and node tasks are often treated as two sides of the same coin in the literature, in my point of view *they are fundamentally different*, at almost every level, as I argue below. It is true that some mathematical tools from graph theory will inevitably be found in both, and GNNs can treat both with minimal structural adjustement (see next section), but otherwise they differ in *all intuitions, mathematical formalisms*, and properties of typical graph datasets.

Besides the relative novelty of supervised prediction problems, modern graph ML also exhibits other defining characteristic: for instance, the quasi-systematic presence of *additional observed data*. Typically, **node features** are available with the graph structure, that is, additional characteristics for each node of the graph. Historically, most methods focused on graph structure, and in particular node features are not often found in *theoretical analyses* of graph algorithms.

In the next two subsections, we dive a bit deeper into the difference between graph tasks and node tasks, and introduce several useful concepts along the way.

1.2.1 Graph tasks

In graph tasks, one wants to predict a single quantity y for an entire graph [6] G: for instance, predicting various properties of molecules from their structure [58], classifying protein-association networks in taxonomic groups [70], and so on. Typically, one has access to a *training set* of labelled graphs $(G_1, y_1), \ldots, (G_N, y_N)$, and aim to learn a prediction function $f(G) = y$.

Typical datasets for graph tasks contain rather *small* graphs of size n_i, with a few hundreds nodes at most, but potentially a very large *number* of graphs N. Intuitively, the higher the n_i, the more difficult the task, since the data is more "high-dimensional", and the higher the N, the easier the task, since it is the number of observed samples, as in "regular" ML.

Graph tasks are i.i.d. In many aspects, graph ML is very similar to regular supervised ML: one has access to a training set of "independent" labelled examples, and wants to learn

6. Recall that here G refers to all quantities observed for one graph, typically graph structure *and* node features.

a prediction function. Mathematically, it can even operates under the same hypotheses: the samples (G_i, y_i) can be assumed to be drawn i.i.d. from a probability distribution over the set of all graphs and labels. That is why the vast majority of (early) theoretical studies of graph ML focused exclusively on graph tasks [56, 148]: many notions and statistical tools of regular ML can be used and "just" need to be adapted to graphs.

Permutation invariance. That is not to say that graph tasks are "easy". Indeed, traditional notions of ML must be adapted to the highly irregular, discrete objects that are graphs. As we outlined in the previous section, a central difficulty is linked to *graph isomorphism*, or the fact that two graphs that are a re-ordering of each other are the "same" graph. For graph tasks, prediction functions need to be **permutation-invariant**: for a permutation σ and graph G, one must have

$$h(\sigma \cdot G) = h(G) \tag{1.7}$$

Generally this permutation invariance will be hard-coded into the function under consideration: as we will see, GNNs are by construction permutation-invariant. However the theoretical analysis of such functions may be difficult: for instance, in ML a central notion is that of function *regularity*: intuitively, a problem is easy if the ideal (unknown) function transforming the input into the output is "smooth". Here expressing such "smoothness" of the prediction problem brings forth the notion of permutation-invariant *metrics* over graphs, which are known to be a difficult topic linked to graph isomorphism [14]. The salient properties of graph prediction tasks are summarized in Tab. 1.1.

1.2.2 Node tasks

Node tasks are prediction problems over the *nodes* of one or several graphs. Arguably, for real-world applications they are far more numerous than graph tasks [70, 125]: almost all examples of graphs from the previous section are generally used to predict properties at the *node* level, from inferring the preferences of users of a social network [156], to predicting the role of proteins from their association network [181], predicting traffic in a road network [74], assessing the roles of brain regions [50]...

Unlike graphs tasks, which are similar to traditional supervised ML with i.i.d. labelled data, node tasks exist in several flavors: one may have access to training nodes within one *or* several graphs, these graphs may not have their full set of nodes labelled, we may want to perform prediction over *several* test nodes at once, these test nodes may be within the *same* partially labelled graphs as the training nodes, or in new graphs... and so on.

For this reason, node tasks are often likened to *Semi-Supervised Learning* (SSL) [27, 147], a paradigm in ML in which unlabelled data is available in addition to labelled data during training.

In SSL, the unlabelled data is ideally used to help the training when compared to using only the labelled data, for instance by helping estimating the overall distribution [127]. In graph ML, the term "SSL" just refers to the fact that unlabelled nodes may be found in the same graph as the training ones – it is not clear whether increasing their quantity (having a large graph) helps at all. In fact, one generally distinguishes two types of SSL in graph ML:

1. In **Transductive SSL** [2], the unlabelled nodes are already available at training time, within the same graphs as the training nodes. The goal is then to "extend" the labels from the training nodes to the unlabelled ones. In particular, it is not necessary to be able to generalize to new graphs not seen during training.
2. In **Inductive SSL** [61], it is possible that new graphs with no labelled nodes are encountered at test time. The model must be able to generalize to these new graphs. In this manuscript, we will mostly focus on the more general case of inductive SSL.

Datasets for node tasks are typically the opposite of that for graph tasks (see Tab. 1.1): the graphs are generally large (high n), with thousands to millions of nodes, and few of them will be considered (low N). In fact, one usually considers the case $N = 1$ of *one* large graph at training or testing. More importantly, from a conceptual and theoretical point of view, the main difference of node tasks compared to graph tasks is that nodes **are not independent anymore**, since they are found in the same graph. So, one generally cannot rely on the classical ML i.i.d. framework, upon which the majority of mathematical tools are built. New, specific notions are needed, and this is still an active area of research, with many potential candidate solutions but no consensus yet.

Permutation equivariance. Similarly to graph tasks, prediction functions for node tasks must respect graph isomorphism. However here one does not talk of *invariance*: when the input graph is permuted, the *output must be permuted in the same way*. Functions with this property are called **permutation-equivariant**:

$$h(\sigma \cdot G) = \sigma \cdot h(G) \tag{1.8}$$

where here the output returns a label per node $h(G)$ and $\sigma \cdot h(G)$ refers to permutation of the output. Again, such equivariance is almost always hard-coded into the models. In theoretical analyses, depending on the case, equivariance may be "much" harder to handle than invariance [NK11], or not [164, 133].

The main differences between graph tasks and node tasks are summarized in Tab. 1.1. I reiterate that in my point of view, *they are in a completely different realm*, even if the mathematical tools of graph theory are of course useful for both. This is also valid for the main properties of typical datasets, and one must strive to perform analysis in *relevant regimes*: for instance, some works

[104] study graph tasks in the large-graph regime ($n \to \infty$) and derive convergence rates when $N \to \infty$ as well: this is not realistic, as typical graphs for graph tasks are small, and there is virtually no datasets where both n and N are large.

> *In my work and this manuscript, I generally focus on inductive node tasks on a single large graph ($n \gg 1, N = 1$).*

1.3 Graph Machine Learning... with Graph Neural Networks

Graph Neural Networks (GNNs) broadly refer to deep neural networks on graphs. They are generally traced back to the work of Scarselli in 2009 [137, 136], while the work of Bruna and co-authors [22] is concurrent to the first deep learning explosion in 2012/13. Another seminal work is that of Kipf and Welling in 2017 [77], who coined the term "Graph Convolutional Networks [7]" (GCN), which still as a baseline model in most experiments, and finally Gilmer et al. in 2017 [58], who coined the term "Message-Passing Neural Networks".

Interestingly, if we loosely describe the most popular models of GNNs in chronological order, they somewhat became *simpler* as time went on: from *Spectral* GNNs with any filters [22], to the use of polynomial filters as in ChebNet [38], to Message-Passing GNNs [58], which is how GNNs are generally described nowadays. We use this order to describe GNNs below. Of course, in parallel, infinitely many variants of these models were proposed in the literature, with various levels of complexity.

GNNs: main properties. At each layer, a GNN update *node representations* $Z^{(k)} \in \mathbb{R}^{n \times d_k}$ using the graph structure, usually starting with the node features [8] $Z^{(0)} = Z$. These updates must be **permutation-equivariant**: if the graph G becomes $\sigma \cdot G$, then the representation $Z^{(k)}$ at each layer k must become $\sigma \cdot Z^{(k)}$. Then, after L layers, the output of the GNN is:

$$\begin{cases} \Phi(G) = Z^{(L)} \in \mathbb{R}^{n \times d_L} & \text{for node tasks} \\ \bar{\Phi}(G) = F(Z^{(L)}) \in \mathbb{R}^{d_L} & \text{for graph tasks, where } F \text{ is permutation-invariant} \end{cases} \tag{1.9}$$

hence, $\Phi(G)$ is indeed permutation-equivariant and $\bar{\Phi}(G)$ is permutation-invariant. The function F, sometimes referred to as *pooling*, is generally a simple global average of the node representations or a coordinate-wise maximum. As mentioned before, in this manuscript we will focus on node tasks, and therefore Φ.

7. although we will see that this term is discutable: GCNs do not really perform convolutions, but rather, in the modern terminology, Message-Passing

8. or handcrafted Positional Encodings, see Sec. 1.3.3.3.

1.3.1 Spectral GNNs

The first models of GNNs [22] were based on the idea of adapting Convolutional Neural Networks (CNNs) [82] from traditional signals like images and sounds to the irregular geometry of graphs. A deep CNN iteratively modifies a signal by stacking layers that use three main ingredients:

1. a **convolution** operation, the main ingredient of CNNs from which they get their name. This is a linear, localized transform. This is generally done through the use of small *convolution kernels* whose values are learned during training.
2. a non-linear **activation function** applied element-wise, necessary to obtain a non-linear model. This is usually a very simple function, such as ReLU $\rho(x) = \max(0, x)$.
3. finally, a **pooling** [9] operation, which is akin to a downsampling of the signal (e.g. with local averages or maximum), and intuitively allows to obtain "hierarchical" representations of the signal.

Bruna and co-authors [22] aimed to adapt these operations on signals on graphs. The activation function is not a problem and left as is. The *pooling* operation is difficult on graphs: it roughly corresponds to a hierarchical "coarsening" of the graph. This is a difficult problem, for which there is no real consensus on the best way to proceed, and is still an active area of research [108, 60]. In modern introductions to GNNs, pooling is generally skipped, and we will not talk about it in this manuscript.

The main ingredient of "Spectral" GNNs is therefore the convolution. Unlike images, it can be quite difficult to define a purely-spatial convolution operation on graphs (although there has been work on this [44]), and Bruna et al. [22] use graph Fourier filtering, which we briefly introduced in the previous section.

Given a graph-matrix $S = S(A)$ (such that $S(\sigma \cdot A) = \sigma \cdot s(A)$, such as a graph Laplacian) that can be diagonalized, at each layer the node representations are updated with a collection of filters [10] $h_{i,j,k}(S)$: for $1 \leqslant i \leqslant d_k$, the columns of $Z^{(k)}$ are updated as

$$Z^{(k)}_{:,i} = \rho_k \left(\sum_{j=1}^{d_{k-1}} \varphi_{i,j,k}(S) Z^{(k-1)}_{:,j} + 1_n b^{(k)}_i \right) \in \mathbb{R}^n \tag{1.10}$$

where ρ_k is the activation function at layer k, and $b^{(k)} \in \mathbb{R}^{d_k}$ is a bias term. Note that to respect permutation-equivariance, the same bias b must be added to each node. The filters $\varphi_{i,j,k}$ are generally parametric functions [11], whose parameters are learned along with the biases. One can

9. that uses the same name as the final pooling F to go from Φ to $\bar{\Phi}$, but refers to a very different concept
10. where we recall that in $h(S)$, $h : \mathbb{R} \to \mathbb{R}$ is a function applied to the eigenvalues of S
11. In the original paper [22], the authors even propose to directly learn n diagonal coefficients $U diag(\theta) U^\top$ instead of a function φ. However, this type of filter cannot be transferred from graph to graph, they are *specific* to a particular graph. On the contrary, a filtering *function* can be applied to any set of frequencies, and therefore

easily check with a simple recursion that these node representations are indeed permutation-equivariant: since $\varphi(S(\sigma \cdot A)) = \sigma \cdot \varphi(S(A))$, we have

$$\varphi(S(\sigma \cdot A))\sigma \cdot Z = \sigma\varphi(S(A))\sigma^\top \sigma Z = \sigma\varphi(S(A))Z = \sigma \cdot [\varphi(S(A))Z]$$

so the filtering operation is permutation-equivariant.

Spectral GNNs with polynomial filters. In GSP, it is known that general filters are costly to compute, since they require to diagonalize the matrix S. This is in general in $O(n^3)$, and therefore unreachable for medium-to-large graphs. A solution was proposed slightly before the first introduction of GNNs by Hammond et al. [62]: using **polynomial filters** $\varphi(\lambda) = \sum_{\ell=0}^{M} \beta_\ell \lambda^\ell$ for some order M. Since polynomes can uniformly approximate any function on a compact interval, if the eigenvalues of S are bounded – typically, for the normalized Laplacian L_{sym} where $\lambda \in [0, 2]$ for any graph; any filter can be approximated by a polynomial filter with M high enough.

The key property of polynomial filters is that we have

$$\varphi(S) = \sum_{\ell=0}^{M} \beta_\ell S^\ell \,.$$

That is, *there is no need to diagonalize* S, but only to compute its powers. Even more efficiently, the filtering of a signal z can be done by only performing recursive matrix-vector multiplications $z \leftarrow Sz$ to compute $S^\ell z$, that is, there is no need to explicitely compute and store S^ℓ. Indeed, for sparse matrices S (i.e. that have a lot of 0's), matrix-vector multiplication is *very* efficient, as it costs $O(nnz(S))$ where $nnz(S)$ is the number of non-zero entries in S. Graph matrices are usually sparse, with $nnz(S) \approx |E|$ the number of edges in the graph, and $|E| = O(n)$ for so-called *sparse* graphs. Computing the whole filtering operation is therefore in $O(M|E|) = O(Mn)$, which is generally insignificant compared to the original $O(n^3)$! Historically, the authors in [62] propose to use *Chebyshev approximation* to efficiently compute recursively the desired polynomial approximations, hence the first architecture to use polynomial filters was coined ChebNet in [38].

Interestingly, polynomial filters are also *localized*: if we take S as the graph adjacency or Laplacian, the filtering of a signal z at node i only depends on the values of z at nodes that are at most M-hops away from i (we say that each power of S extends the *receptive field* by one hop). Thus, we find again a property of CNNs – where convolution kernels are localized – that was lost for general Spectral GNNs.

any graph.

1.3.2 Message-Passing GNNs

In parallel to these GSP-inspired approaches, researchers have built GNNs by iteratively passing *messages* along the edges of the graph [58], similar to **message-passing** (MP) algorithms such as the classical Belief Propagation algorithm, used in many fields including chemistry and statistical physics. Within such a *Message-Passing GNN* (MPGNN [12]), at each layer, for each node, the representation is updated by using that of its neighbors. In its most general form, for $1 \leqslant i \leqslant n$ the *rows* of $Z^{(k)}$ are updated as

$$Z^{(k)}_{i,:} = \mathrm{AGG}_k \left(Z^{(k-1)}_{i,:}, \left\{ \left(s_{ij}, Z^{(k-1)}_{j,:} \right) \right\}_{j \in \mathcal{N}(i)} \right) \tag{1.11}$$

where $\mathcal{N}(i)$ is the neighborhood of node i, and AGG_k is an **aggregation function** that takes as input the representation of the node at the previous layer, and the *set* of representations of the neighbors along with potential values s_{ij} of a graph representation matrix S. The key property of this function is that it considers the neighbors' representations $\{(s_{ij}, Z^{(k-1)}_{j,:})\}_{j \in \mathcal{N}(i)}$ *as a set* [13], such that its value is *invariant by reordering of the neighbors*. Consequently, it is easy to see that the whole MP process is globally permutation-equivariant.

Remark 1.1. *Note that Spectral GNNs* (1.10) *update graph signals at each dimension (the columns of Z), while MPGNN* (1.11) *update each node representation (the rows of Z).*

There are many examples of aggregation functions and message-passing schemes. We will describe several of them in detail in Chapter 3. Most of them belong to the general class of "sum" aggregations, that is, they perform a weighted sum over the neighbors. This is generally followed by linear weights and a non-linear activation function.

$$Z^{(k)}_{i,:} = \rho_k \left(\theta^{(k)}_0 Z^{(k-1)}_{i,:} + \theta^{(k)}_1 \left(\sum_{j \in \mathcal{N}(i)} s_{ij} Z^{(k-1)}_{j,:} \right) + b^{(k)} \right) \in \mathbb{R}^{d_k} \tag{1.12}$$

for weights $\theta^{(k)}_t$ and bias $b^{(k)}$, and coefficients s_{ij}. In this case, the message-passing can be written in a matrix form as:

$$Z^{(k)} = \rho_k \left(Z^{(k-1)} \theta^{(k)}_0 + S Z^{(k-1)} \theta^{(k)}_1 + 1_n b^{(k)\top} \right) \in \mathbb{R}^{n \times d_k} \tag{1.13}$$

We will call such models **S-MPGNNs**, or simply **S-GNNs**. They form the majority of the popular architectures: GCN [77], GIN [160], etc.

Some aggregation schemes depart from S-GNNs. The most common is probably attention-based

12. as mentioned before, nowadays the term "GNN" implicitely refer to MPGNN, and in the rest of the manuscript when we do not explicitely write otherwise this will be the case

13. technically, this is a *multiset*, as we authorize doublons. The important part is the permutation-invariance.

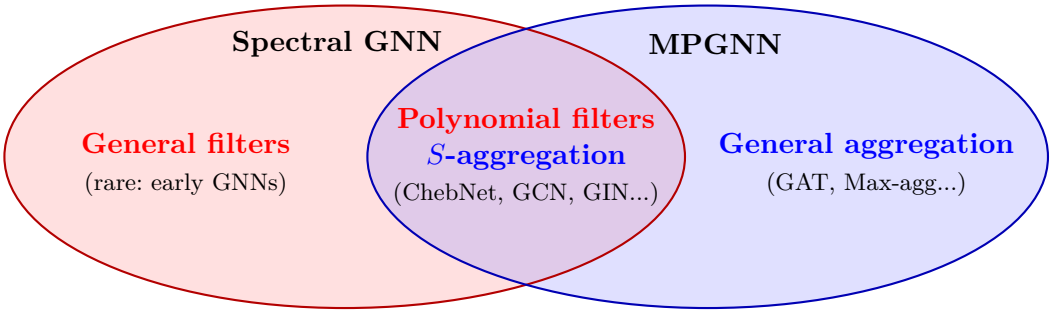


Figure 1.2

models such as GAT [151], where the s_{ij} are replaced by coefficients $c_{ij}^{(k)}$ that are allowed to depend on the previous layer's node representations at each end of the edge, that represent attentional weights along the edges. We will give several examples in Chapter 3.

Polynomial filters and S-MPGNNs. While Spectral GNNs and MPGNNs are often presented separately as different architectures (mostly due to their historical origines), interestingly, the most popular models of these two architectures *are actually the same models*! Namely: S-MPGNNs (1.13) are actually Spectral GNNs with polynomial filters (in S) of order $M = 1$; and conversely, polynomial filters of higher (finite) order can be implemented as S-MPGNNs with layers that have $\rho_k = Id$ as activation function for some layers (more precisely, stacking M layers with $\rho_k = Id$ can implement any polynomial of order M, followed by one layer with general ρ_k to exactly obtain the desired Spectral GNN). This overlap is visualized in Fig. 1.2. While this architecture (1.13) is *by far the most common*, both for MPGNNs and Spectral GNNs, there are still models that fall outside of it: for Spectral GNNs, that would be general (non-polynomial) filters, which are hardly used in practice, as being too costly, but may retain a theoretical interest. For MPGNNs however, other sum-aggregation schemes such as attention-based GAT are indeed used in practice.

> *In the rest of the manuscript, I will focus on S-GNNs, except briefly at the end of Chapter 3 where I will mention some other examples of MPGNNs. Unless otherwise mentioned, the term GNN will be used for MPGNN.*

1.3.3 Discussion

Of course there are many more aspects to GNNs, but the underlying, fundamental landscape is not far from the simple picture that I have drawn in the section above. Here I discuss a few topics that will help the interested reader get a better grasp of the literature around GNNs, and point out to a few references. Of course, this is strongly biased towards my personal interest.

1.3.3.1 Pros and cons of GNNs

(Some) advantages of GNNs. Compared to other models, and earlier methods on graphs, the main advantage of GNNs is without doubt their **flexibility**. While earlier methods were taylored for specific problems or specific graphs, GNNs can be adapted to virtually any task with little architecture change: going from node task to graph task only requires a final aggregation F, edge prediction can be performed by using pairwise node representations [135], and so on. More importantly, the architecture of the GNN does not depend on the underlying graph structure, such that GNNs trained on one graph can be directly applied to another graph (whose node features have the same dimension) *with the same set of parameters*. Whether this "transfer" is successful is of course another matter, and the topic of this entire manuscript.

Regarding MPGNNs, one of their main advantages, and a major reason why they dominate the current landscape despite the existence of more sophisticated variants, is their very light computational cost. Indeed, one round of message-passing scales linearly in the number of edges $O(|E|)$, which tends to be of the order of $O(n)$ since real-world graphs are often sparse. Moreover, message-passing can be implemented efficiently by representing the graph as a "list of edges", and avoiding cumbersome matrix representations or sparse matrix representations, which always incur a complex gestion of efficient matrix-vector multiplication. Unsurprinsingly, lists of edges are the default representation of graphs in the popular PyTorch Geometric library [52].

(Some) limitations of GNNs. While MPGNNs remain the most popular class of GNNs by far, their relative simplicity do lead to limitations, among which we can non-exhaustively cite:

— **Homophily vs heterophily**, which respectively refers to the fact that connected nodes tend to have similar (resp. dissimilar) node labels or features. MPGNNs are generally thought to be more appropriate for homophilic graphs [96, 93, 177], as message-passing tends to make connected nodes have similar representations. Dealing with heterophilic graphs is an active area of research, with many proposed solutions [180, 176] but no clear consensus.

— **Oversmoothing**, which refers to an intuitive phenomenon that was observed quite early [85, 131]. Loosely speaking, each rounds of message-passing tends to "smooth" the node representations, e.g. by performing some kind of sum or averaging (1.12). Hence, for *very deep GNNs*, the node representations may become indistinguishable, which makes node tasks difficult to perform. This prevents vanilla MPGNNs from being too deep. On the other hand, shallow GNNs are subject to **underreaching**, where nodes that are distant from more than the number of layers do not interact with each other at all in the computation of their respective representations. Common fixes to oversmoothing includes various levels of skip connections [86] or normalization [175].

> *I made several contributions on oversmoothing [NK9, NK10], see Chapter 4.*

— **Oversquashing and information bottleneck**. Different terms are used to describe several phenomena related to the representation and propagation of "information" in MPGNNs. *Oversquashing* [1] generally refers to the fact that, as the number of layers grows, the amount of information that needs to be contained in a node representation "grows exponentially", since this is generally the case of the *receptive field*, i.e. the number of nodes affecting the representation (the neighbors, then the neighbors' neighbors, and so on). Researchers argue that this is generally problematic, as node representations have a fixed dimensionality, generally independent of the underlying graph: this large amount of information is "squashed" into a single vector.

Related to this, various "information bottlenecks" have been discussed in the literature, mostly related to the propagation of information between two distant nodes, which can be problematic if the graph has a "bottleneck" of low-connectedness area. All these phenomena have been characterized using various notions to describe the information flow in the graph, such as curvature, effective resistance, and so on [145, 16].

All the phenomena above roughly underline the facts that MPGNNs have trouble with dealing with **high frequency signals**, since they tend to smooth out node representations, and with **long-range dependencies**, since they only communicate locally. Many layers are required for long-range communications, which results in a host of different problems. To summarize, GNNs need to be deep to handle underreaching and long-range communication, but they cannot be too deep due to oversmoothing and oversquashing. Many MPGNN variants have been proposed in the literature to address these issues, but there is no clear consensus on a single best architecture.

Expressivity of MPGNNs. In addition to these rather informal observations, a very large number of early theoretical studies of MPGNNs have seeked to formally characterize the "expressivity" of MPGNNs, that is, roughly "what they can compute". It has been known since the 80s that simple NNs such as the fully-connected Multi-Layer Perceptrons (MLP) are *universal* [68, 36, 123], in the sense that they can approximate any (continuous) function. For GNNs, it was quickly realized that the permutation-invariance/equivariance constraint would likely prevent such universality for GNNs: indeed, universality would imply solving the *graph isomorphism* (GI) problem (and is in fact more or less equivalent to it [30]), that is, the problem of deciding if two graphs are permutation of each other, a major combinatorial problem in graph theory [14]. Since algorithms such as the so-called *Weisfeiler-Lehman* (WL) algorithm [157] are well-known approximate GI solvers in graph theory, and use some message-passing mechanism, the seminal study by Xu et al. [161] showed that MPGNNs were *at most as powerful as the WL algorithm* for

14. Interestingly, no polynomial algorithm is known for the GI problem, but it is also not proven to be NP-complete. Some authors argue that GI might represent a complexity classe of its own strictly in-between P and NP (if they are different!). The best known algorithm is quasi-polynomial, in $O(2^{O(\log n)^3})$ [5, 63].

the GI problem (and could be made to be exactly as powerful). This was a first characterization of their "expressivity": it is known that the WL algorithm sometimes returns false positives, but that these failure cases can be studied [43].

Of course, the GI problem is a discrete, true-or-false problem that may have little to do with "real" prediction problems. Many works have since introduced "softer" notions of expressivity for GNNs [111, 23, 171, 126]. This is still an active area of research.

> *As described in Chapter 3, in my work I characterize some approximation power of GNNs in the large-graph limit instead [NK15, NK13, NK2], which allows to circumvent a lot of the combinatorial difficulties.*

1.3.3.2 A few other variants

Some GNN architectures do depart from traditional MPGNNs or Spectral GNNs, but they are far less common in practice.

For instance, some models process (bags of) *subgraphs* [15, 8], mimicking the historical "graphlet kernel" [142]. They lead to flexible, efficient models but add a layer of dependency on the manner in which the subgraphs are extracted, and may have limited expressivity for large graphs.

Several models employ variants of "high-order" message-passing scheme or representation of the graph. It can be projected to *high-order tensors* in an equivariant way [100], which leads to more powerful models [101, 99]. It is also possible to augment the message-passing scheme in various manners: for instance by considering higher-order structures in the graph [110], or by adding an extra "node-dimension" at various stages and pooling it later on in a permutation-invariant way [154]. However, all these models are generally (far) more costly than traditional MPGNNs since they are not linear in the number of edges, and are quite rare in practice.

> *In [NK11] we show that high-order tensors lead to true universality for equivariant models. This contribution is not described in this manuscript.*

Another class of models seeks to *modify the underlying graph* to promote desirable implicit biases: counteract oversmoothing or oversquashing [51], reduce heterophily, adjust the sparsity of the graph one way or another, and so on. This is a simplified instance of *graph learning*, a difficult problem: when done naively it is quadratic in the number of nodes, and moreover there is a trade-off to how much the graph must be modified, since it is supposed to contain useful information. This can be done by graph *rewiring* [10] or sparsifying [128], often to optimize some notions such as curvature [51, 145] or effective resistance [16].

Finally, some architectures reintroduce the use of *pooling* [165, 108, 55, 60], as was done in CNNs. This bears links with *graph coarsening* [91]. As we mentioned, this is a difficult problem

for graphs that requires delicate hyperparameter tuning, algorithmic choices, may incur stability issues, lead to degenerate graphs, and so on. However the hierarchical decomposition brought by pooling may have several advantages: it allows for long-range message-passing while avoiding oversmoothing and oversquashing, as well as lead to computational efficiency both in time and memory footprint.

> *Graph coarsening and its guarantees in Machine Learning are the topic of the thesis of A. Joly [**NK6**, **NK7**], co-supervised with A. Roumy. These contributions are not described in this manuscript.*

1.3.3.3 On node features

One important feature of GNN is that they *require* node features $Z^{(0)}$ as input. In the absence of such features, with only the graph structure available, various strategies can be deployed, but it is generally admitted to be a rather problematic situation, as node features can serve to "disentangle" a lot of the symmetries [15] of the graph.

The simplest, and most often used, strategy is to simply input constant node features [NK15]. After a few layers, the node representations contains information about the graph structure: for instance, after one layer of S-MPGNN where $S = A$ the adjacency matrix, the node representations are the degrees. This strategy can however significantly worsen the oversmoothing phenomenon. To address this, another simple strategy is to input *random* node features [134], to the price of adding randomness to the output and *technically* not respect permutation-invariance/equivariance (but only "in expectation"). Some theoretical works do entirely break the permutation-invariance/equivariance by using unique node identifiers [90], but they cannot be implemented to generalize to new graphs, and are not used in practice.

In the absence of a perfect solution, a general methodology is to *handcraft* node features, often referred to as *Positional Encodings* (PE) [45, 17] (inspired by the vocabulary of Transformers [149]), since they are supposed to identify the "position" of the node in the graph and disentangle some symmetries. Note that: a) they must still be permutation-equivariant, and thus technically bear little ressemblance to the PEs in Transformers, and b) they are often inspired by well-tested, historical algorithms on graphs (whose results are then fed as inputs to GNNs). For instance, some PEs in the literature are based on eigenvectors of the adjacency matrix or Laplacian of the graph [45, 46] (with recent variants to handle the sign/basis indeterminancy [89]), random-walks [46], node metrics [166, 88], or subgraphs [19]. Interestingly, some of these have been shown to lead to overall architectures with an expressive power beyond WL [88, 19, 89], without changing the subsequent MP process. As it is, the community somewhat admits that PEs are not an ideal

15. so-called *auto-morphisms*, that is, permutations that leave the graph unchanged.

solution, as it is a "hybridization" of deep and non-deep methods that may be discutable. For instance, MPGNNs are precisely used for computational efficiency compared to spectral methods, and computing the eigenvectors of the Laplacian as PEs negates this. Moreover, it is clear that eigenvectors already contain a lot of information, which is probably redundant with a lot of operations performed within the relatively simple MP scheme.

> *I have worked on PEs in the context of random graphs [NK13], see Chapter 3.*

1.4 Conclusion and discussion

I finish this chapter with a few remarks on GNNs and GML in general.

One architecture to rule them all? In the last few decades, graph ML and GNNs have gone from niche subfields to primary citizens in ML and deep learning. GNNs are applied to everything everywhere [16], thanks to their limitless flexibility and hard-coded invariances. However, that is also their curse: by being "too general", they are sometimes not adaptive to the specificity of the data for a particular application. Balancing between generality and specialization is without doubt one of the great challenges to advance the field.

Is it even feasible? It is possible that treating all graphs as a unified type of data might be an exercise in futility: one might argue that, from one scientific field to another, graphs are as profoundly different as images are from, say, financial data, despite both being vectorial data. For instance, some graphs are highly structured (e.g. the vanilla Ising model in statistical physics is a lattice), while others admit no kind of structure at all (e.g. a social network). By default, a unified GNN applicable to all cases will *not* exploit the additional structure of the former.

That being said, it is true that the recent history of deep learning mitigates this impression: DL has seen the birth of seemingly all-powerful tools, that can more or less process any kind of data with enough training samples and time. With minimal adaptation, Convolutional Neural Networks (CNN) and other NNs have become the *de facto* state-of-the-art models for many kind of data, even where they were not expected to. This has been partly true for GNNs, but arguably in a less stellar manner. There is still much debate and research around the "universality" of current GNNs [11] (in the informal sense of the term), and whether we should return to more specialized models for different graphs. Nevertheless, until now "blindly" applying GNNs to all graphs, for good or for worse, has been rich in insights.

Do we need graphs? To finish, I should mention that researchers in GML, and graph data science in general, may tend to "have a hammer" and "consider that everything is a proverbial

16. all at once

nail". In particular, the *graph representation* itself is rarely questioned: it is virtually *always* possible to build a graph to represent some data, but is it relevant? On a related note, the processus of *building* the graph is crucial, but rarely integrated within data science pipelines. For instance, in neuroscience, researchers typically use correlations between EEG signals to build a graph of connected brain regions [81], but depending on the downstream task to solve, might it not be better to "directly" process the EEG data? Is the intermediate graph really useful?

> *As a theoretician, my own contributions and considerations tend to remain quite far from these discussions: I am given a graph that experts in the field worked really hard to produce, I process it without questioning its provenance.*
>
> *But I am still conscious that such questions are crucial, especially since my own work include the use of statistical models. Ideally, such models should reflect the interesting aspects of real data, and therefore include mechanisms flexible enough to accommodate for graphs coming from many different fields. As of now, I still consider that the* adaptivity *of random graph models remains an outstanding open question, that will be discussed more in the Conclusion of this manuscript.*

Chapter 2

Statistical approach to Graph Machine Learning: a generic framework

Contents

Summary

This chapter introduces a general **mathematical framework** for node prediction tasks, staying as close as possible to **classical ML**.

— Following a large-graph approach, we introduce the class of prediction function whose expected risk *converges* when the graph grows. We then decompose the *excess risk* in three terms, instead of two as in classical ML, with an additional "transferability error".

— On Latent Position Models (LPMs), functions on graph may converge to *functions on the space of latent variables*. We call this notion $\mathfrak{P}$-convergence.

— Using $\mathfrak{P}$-convergence, we draw *formal links* between Graph ML and regular ML. We describe general methodologies to bound all terms in the excess risk decomposition, and bound the generalization error of node-prediction models on large random graphs.

As the time of the writing of this manuscript, the material in this chapter is mostly new and unpublished. Sec. 2.4.4 was done in collaboration with Y. Viegas during his internship.

In the previous chapter, we have informally described how *node prediction tasks* escaped the classical "i.i.d." framework of machine learning. In this chapter, we introduce the mathematical tools to understand this notion, and see how to adapt classical tools for GML *on random graphs*. In particular, Latent Position Models (LPMs) will allow to draw strong links between ML and node-level GML. The chapter is decomposed as follows:

— In Sec. 2.1, I introduce some basic definitions of "regular" Machine Learning: Empirical Risk Minimization, excess risk decomposition, generalization error, and so on.

— In Sec. 2.2, I adapt this framework to node prediction tasks on random graphs, with a focus on the *large graph limit*. I thus revisit several definitions and introduce a novel excess risk decomposition.

— In Sec. 2.3, I introduce in more details Latent Position Models.

— In Sec. 2.4, I revisit the framework of Sec. 2.2 for LPMs. Through the novel notion of "$\mathfrak{P}$-convergence", I show how a large class of GML models can be related to traditional ML, through functions on the latent space of the LPM. I give some general methodologies to bound the generalization error, including a generic covering numbers approach.

Note that this chapter will involve a fair amount of mathematical notations. The most important are gathered in the Table at the beginning of the manuscript. All technical proofs are gathered in the Appendix of the chapter.

Notations. The norm $\|\cdot\| = \|\cdot\|_2$ is the Euclidean norm for vectors, and more generally $\|v\|_q = (\sum_i v_i^q)^{1/q}$.

2.1 Statistical Machine Learning: a crash course

There are uncountably many good books on ML [140, 109, 173, 7]. Here, following the spirit of this manuscript, I will give a straight-to-the-point, *simple* crash course, that will voluntarily ignore a fair amount of important material. In particular, I will focus on a simple, mathematical presentation of *supervised* ML focusing on its statistical aspect, leaving asides most of the *algorithmic* aspect that constitutes the day-to-day life for ML practitioners. I am mostly inspired by F. Bach's excellent book [7], which has become my go-to reference on the topic.

In supervised ML, the goal is to predict a value $y \in \mathcal{Y}$ from an observed quantity $x \in \mathcal{X}$. One often distinguishes two main classes of problems: *classification*, where y is categorical, and *regression*, where y is continuous. Classification is generally modelled by adopting $\mathcal{Y} = \{1, \dots, C\}$, with a large part of the literature studying $C = 2$ as a representative case. Regression is simply modelled with $\mathcal{Y} \subset \mathbb{R}^p$, again emphasizing $p = 1$ as a representative case. In this manuscript, the observation x will always be a vector in $\mathcal{X} \subset \mathbb{R}^d$, and $\mathcal{X}$ will generally be compact. We will call x a *sample*, while y will always be called a *label* (even in the regression case).

The vast majority of ML models can be modelled as **prediction functions** $f : \mathcal{X} \to \mathcal{Y}$; although this view, especially in graph ML, can be challenged. It can sometimes be convenient to consider functions f that yield an output in a different set Y instead of directly $\mathcal{Y}$. The typical example is multiclass classification, for which one may output a set of *probabilities* of belonging to each class instead, that is, $\mathsf{Y} \subset \mathbb{R}_+^C$. A simple maximization over the coordinates then yields a single class in $\mathcal{Y}$ if desired. We denote by $\mathcal{F}_{all} = \mathsf{Y}^{\mathcal{X}}$ the set of all functions from $\mathcal{X}$ to Y.

Generalization and iid assumption. To train ML models, one observes pairs of labelled examples $(x_1, y_1), \dots, (x_n, y_n)$ called the *training set*. The goal is then to exploit this information such that, when a new sample x is observed, one can predict the corresponding y. This notion is loosely called *generalization*, that is, the capacity to "generalize" from training data to new, unseen data. Generalization can only happen if some sort relationship exists between the training set and new samples observed in the future. In statistical ML, one therefore assumes that (x, y) are drawn from a *joint probability distribution* over $\mathcal{X} \times \mathcal{Y}$:

$$(x, y) \sim P_{xy} \in \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \tag{2.1}$$

where $\mathcal{P}(\mathcal{Z})$ represents the set of all probability distributions over some set $\mathcal{Z}$. In terms of intuition, the important object is naturally the conditional distribution $P(y|x)$, which if known allows us to perform optimal prediction [1]. Then, the training set is assumed to be formed of

1. namely, it allows to compute the so-called Bayes predictor, see next section

independently and identically distributed (i.i.d.) examples:

$$(x_i, y_i) \overset{\text{iid}}{\sim} P_{xy} \tag{2.2}$$

The i.i.d. assumption (especially the "independent" part) is absolutely crucial to all of ML mathematics, and often represent the most basic of assumptions, except in specific applications (e.g., times series). In graph ML, **for node-tasks**, this modelling may be challenged, as nodes are naturally linked. This explains why a large part of the graph ML literature focus on graph tasks [56, 103].

Loss function. To measure the quality of a prediction $\hat{y} \in \mathsf{Y}$ against a label $y \in \mathcal{Y}$, one uses a *loss function* $\ell : \mathsf{Y} \times \mathcal{Y} \to \mathbb{R}_+$: the higher the loss, the worse the prediction. The loss function is not necessarily symmetric, and by convention we will adopt $\ell(f(x), y)$, i.e. the prediction as the first argument and label as second. In general, it will often be convenient to have a loss that is Lipschitz with respect to the first coordinate. Our two running examples will be the following.

Example l1 (Classification)**.** *For classification, the labels are discrete* $\mathcal{Y} = \{1, \dots, C\}$*. Generally, it is more convenient to allow prediction models to have continuous outputs in* $\mathsf{Y} = \mathbb{R}^C$*, and perform a maximization afterward. One popular loss in deep learning is then the so-called* Categorical Cross-Entropy *(CCE):*

$$\ell(\hat{y}, y) = -\log\left(\frac{e^{\hat{y}_y}}{\sum_{c=1}^{C} e^{\hat{y}_c}}\right) \tag{2.3}$$

That is, the vector $e^{\hat{y}}$ *is normalized to summing to one to represent probabilities of belonging to each class, then its* y*th coordinate is passed through a logarithm to follow the so-called log-likelihood methodology. By an easy computation, this loss is* 1*-Lipschitz in its first argument with respect to* $\|\cdot\|_\infty$ *on* $\mathbb{R}^C$*.*

Example l2 (Regression)**.** *For regression, the labels* y *can be any vector in a domain* $\mathcal{Y} \subset \mathbb{R}^{d_y}$ *that is typically bounded. The outputs are in* $\mathsf{Y} = \mathbb{R}^{d_y}$ *(we may remove the boundedness assumption for convenience, even if most prediction functions will naturally be bounded). The loss is the* square loss*:*

$$\ell(\hat{y}, y) = \|\hat{y} - y\|^2 \tag{2.4}$$

Technically, this loss is not Lipschitz unless we assume $\hat{y}$ *to live in a bounded domain as well. But we will often treat it as a particular case, since many computations simplify in this case.*

Overall, we will assume the following on the loss, treating the square loss as a special case since it is not Lipschitz.

Assumption 2.1. *The loss function* ℓ *is either:*

1. *L_ℓ-Lipschitz in its first argument w.r.t. some metric d_{Y}*

$$\forall \hat{y}, \hat{y}' \in \mathsf{Y}, y \in \mathcal{Y}, \quad \left|\ell(\hat{y}, y) - \ell(\hat{y}', y)\right| \leqslant L_\ell d_{\mathsf{Y}}(\hat{y}, \hat{y}')$$

or

2. *it is the square loss* (2.4) *and $\mathcal{Y}$ is bounded by $\|y\| \leqslant D_{\mathcal{Y}}$. In this case we take d_{Y} as the Euclidean norm on $\mathbb{R}^{d_y}$.*

Expected risk. To measure the quality of a prediction function f, one uses the *expected risk*[2]

$$\mathcal{R}(f) = \mathbb{E}_{(x,y)\sim P_{xy}} \ell(f(x), y) \tag{2.5}$$

The ultimate goal of ML is therefore to find a prediction function $f^\star$ that minimizes the expected risk

$$f^\star \in \operatorname*{argmin}_{f \in \mathcal{F}_{all}} \mathcal{R}(f) \tag{2.6}$$

Remark 2.1 (Bayes predictor)**.** *An optimal function $f^\star$ is known as a* Bayes predictor*. Note that it may not always exist: e.g. for classification with the CCE loss, it is well-known that optimal solutions may be unbounded. However we may always compute the* optimal risk*:*

$$\mathcal{R}^\star = \inf_{f \in \mathcal{F}_{all}} \mathcal{R}(f) = \mathbb{E}_x \inf_{\hat{y} \in \mathsf{Y}} \mathbb{E}_{y|x} \ell(\hat{y}, y) \tag{2.7}$$

where $\mathbb{E}_{y|x}$ is the conditional expectation of y w.r.t. x, see [7, Prop 2.1]. When $f^\star$ exists, it is equal to

$$f^\star(x) \in \operatorname*{argmin}_{\hat{y} \in \mathsf{Y}} \mathbb{E}_{y|x} \ell(\hat{y}, y) \tag{2.8}$$

For simplicity, we will generally implicitely assume the existence of all the argmins in this manuscript.

Naturally, it is generally not possible to exactly compute $f^\star$ in practice. Given an *estimator* $\hat{f}$, the difference

$$\mathcal{R}(\hat{f}) - \mathcal{R}^\star \geqslant 0 \tag{2.9}$$

is called the *excess risk*. Deriving estimators from the training set with minimal excess risk is the main goal of ML. In computing $\hat{f}$ instead of $f^\star$, there are two (main) levels of approximation: first, it is not possible to computationally manipulate "all" functions in $\mathcal{F}_{all}$, but rather a convenient class $\mathcal{F} \subset \mathcal{F}_{all}$, and second, one has only access to the training set and not the true P_{xy}.

2. or population risk... among other names

Function space. Concerning the first level of approximation, in practice, the user must choose a set $\mathcal{F} \subset \mathcal{F}_{all}$ of potential prediction functions that can be implemented on a computer, which can be *parametric* $\mathcal{F} = \mathcal{F}_\Theta = \{f_\theta \mid \theta \in \Theta\}$, in which case the learning corresponds to selecting a good parameter $\theta \in \Theta$, or non-parametric. For instance, neural networks are typical parametric models, while the so-called Nadaraya-Watson estimator, which predicts a label by averaging the labels of the closest samples in the training set, is non-parametric. The choice of $\mathcal{F}$ is a core aspect of ML practice and theory, taking into account many trade-offs and expert knowledge, about the nature of the data, computational complexity, the quantity of training data, and so on. Thus, a first level of approximation of $f^\star$ is to define $f_\mathcal{F}$ as the best function [3] *within* the class $\mathcal{F}$:

$$f_\mathcal{F} \in \operatorname*{argmin}_{f \in \mathcal{F}} \mathcal{R}(f)\,. \tag{2.10}$$

Empirical Risk Minimization. The second level of approximation comes from dealing with a finite quantity of data. The vast majority of ML methods follow an approach known as *Empirical Risk Minimization* (ERM), in which the expected risk (2.5) is naturally replaced by the so-called *empirical risk*:

$$\hat{\mathcal{R}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{2.11}$$

Empirical Risk Minimization then seeks to solve

$$\hat{f} \in \operatorname*{argmin}_{f \in \mathcal{F}} \hat{\mathcal{R}}(f) \tag{2.12}$$

Intuitively, the excess risk of $\hat{f}$ will be small if the empirical risk is a good approximation of the true risk. One thus begins to see the importance of the i.i.d. assumption: the empirical risk is an average of i.i.d. random variables, which will naturally "converge" toward its expectation – the true risk – when n grows, a fact that has been studied for centuries [4]. Many statistical tools are available to characterize this convergence, from Central Limit Theorems to advanced concentration inequalities. An subtle but crucial feature of the desired convergence here is that it must be *uniform* over the set $\mathcal{F}$, since we are *minimizing* over it. See more details below.

Error decomposition. Given the successive levels of approximation above, one can decompose the excess risk into an *estimation error* and an *approximation error*: :

$$\mathcal{R}(\hat{f}) - \mathcal{R}^\star = \underbrace{\mathcal{R}(\hat{f}) - \mathcal{R}(f_\mathcal{F})}_{\text{estimation error}} + \underbrace{\mathcal{R}(f_\mathcal{F}) - \mathcal{R}^\star}_{\text{approximation error}} \tag{2.13}$$

3. again, this assumes the existence of such a minimum
4. Bernoulli's first proved the Law of Large Numbers in 1713.

The estimation error materializes the fact that the true risk is unknown, and one has to learn over the training set instead. The approximation error evaluates how much functions in $\mathcal{F}$ can approximate the ideal function $f^\star$. As $\hat{f} \in \mathcal{F}$, it is immediate to see that both terms are nonnegative. Sometimes, one may add an *optimization error*, which expresses the fact that the optimization problem (2.12) may be only approximately solved up to some precision. Although it is of course crucial, we will generally not consider it in this manuscript.

Remark 2.2 (Bias variance)**.** *The decomposition* (2.13) *loosely corresponds to a* bias-variance *decomposition: the approximation error is a "bias" from the true target $f^\star$ to some deterministic function $f_\mathcal{F}$ that only depends on $\mathcal{F}$, while the estimation error is a "variance" that encompasses the randomness of the training set. Informally, the "simpler" the set $\mathcal{F}$ is (e.g., smoother functions, smaller parametric space, and so on), the higher the bias and smaller the variance: $f_\mathcal{F}$ is a worse approximation of $f^\star$, but it is easier to estimate from the training set. The variance also decreases when more training examples n are available, as the true risk is better estimated.*

Bounding the approximation error. Handling the approximation error is more *case-specific* than the estimation error (see below). It highly depends on the problem at hand, the choice of ML models $\mathcal{F}$, and various assumptions on the target function $f^\star$. Some models are said to be **universal**, in the sense that they can approximate any function in some class: for instance, it is well-known that neural networks are universal among continuous functions [36, 68]. In this case the approximation error may vanish, generally when we let some quantity grow unbounded (the dimensions of the neural net, the norm of the weights, etc.). For instance, one may define models with bounded parameters $\mathcal{F}^R = \{f_\theta \mid \Omega(\theta) \leqslant R\}$, such that Ω controls various quantities related to the parameters θ. Then, universality means that the approximation error vanishes when $R \to \infty$. Ideally, one would obtain approximation *rates*, that is, at which speed the approximation error decreases with respect to R, however this a complex topic in which only partial results are known and many open questions remain.

Bounding the estimation error. Unlike the approximation error, the estimation error may be studied in a more systematic way. Most analyses start with the following bound:

$$\begin{aligned}\mathcal{R}(\hat{f}) - \mathcal{R}(f_\mathcal{F}) &= \mathcal{R}(\hat{f}) - \hat{\mathcal{R}}(\hat{f}) + \underbrace{\hat{\mathcal{R}}(\hat{f}) - \hat{\mathcal{R}}(f_\mathcal{F})}_{\leqslant 0} + \hat{\mathcal{R}}(f_\mathcal{F}) - \mathcal{R}(f_\mathcal{F}) \\ &\leqslant 2 \sup_{f \in \mathcal{F}} \left| \hat{\mathcal{R}}(f) - \mathcal{R}(f) \right| \end{aligned} \tag{2.14}$$

Thus, one must characterize the proximity of the empirical risk $\hat{\mathcal{R}}$ to the true risk $\mathcal{R}$ *uniformly over the whole set* $\mathcal{F}$. One notices that the upper bound (2.14) increases with the size of $\mathcal{F}$.

It is relatively easy to bound $\left| \hat{\mathcal{R}}(f) - \mathcal{R}(f) \right|$ with high probability over the training set for

one function f, for instance with concentration inequalities for sum of i.i.d. variables such as Hoeffding's or Bernstein's inequality.

Lemma 2.1 (Application of Hoeffding's inequality)**.** *If the loss function ℓ is bounded, for any function f, with probability at least $1-\delta$,*

$$\left|\hat{\mathcal{R}}(f) - \mathcal{R}(f)\right| \leqslant C\sqrt{\log(1/\delta)/n} \tag{2.15}$$

for some constant C.

Then, most of the technical work must handle the "uniform" part, that is, the supremum over the function class $\mathcal{F}$. Several classical techniques have been developped for this: the so-called covering numbers method, Rademacher complexities... We will skip the details, but, to fix ideas, an example of result is the following:

Lemma 2.2 ([7, Chap. 4.4])**.** *Assume the loss ℓ is bounded. With probability $1-\delta$ over the (x_i, y_i),*

$$\mathcal{R}(\hat{f}) - \mathcal{R}(f_{\mathcal{F}}) \leqslant \mathrm{Rad}_n(\mathcal{F}) + C\sqrt{\log(1/\delta)/n} \tag{2.16}$$

where $Rad_n(\mathcal{F})$ is the Rademacher complexity *of $\mathcal{F}$:*

$$\mathrm{Rad}_n(\mathcal{F}) = \frac{1}{n}\mathbb{E}_{\varepsilon,X}\left(\sup_{f\in\mathcal{F}}\sum_{i=1}^{n}\varepsilon_i f(x_i)\right) \tag{2.17}$$

where $\varepsilon_1, \ldots, \varepsilon_n$ are independant Rademacher variables (i.e. uniform on $\{-1, 1\}$).

Rademacher complexities are often themselves in $1/\sqrt{n}$, and Lemma 2.2 leads to the well-known $1/\sqrt{n}$ for the estimation error. Remark that the *multiplicative constants* appearing in the Rademacher computations are then particularly important to distinguish various models.

> *As we will see, for Graph ML we will be able to implement a "simple" covering numbers approach, which results in the suboptimal rate $O(\sqrt{\log n/n})$. A complete Rademacher approach is still open. Partial results were obtained during the internship of Y. Viegas.*

Notations. For this point on, all risks in "regular" ML will be denoted by $\mathcal{R}_{\mathrm{ML}}$ (such as $\mathcal{R}^\star_{\mathrm{ML}}$, etc.), to distinguish them from the novel definitions of risk that we will make in graph ML. The Bayes predictor will also be denoted by $f^\star_{\mathrm{ML}}$.

2.2 A statistical framework for node tasks on graphs

As mentioned in the introduction, in graph ML, there is a fundamental difference between *graph tasks* and *node tasks*. We first briefly revisit this discussion under the light of the ML framework above. Then, we define a proper statistical framework for node tasks.

2.2.1 A reminder on graph tasks vs. node tasks

Graph tasks are classical ML. Recall that in graph tasks, the goal is produce labels for entire graphs: the training set is formed of a collection of labelled graphs (G_i, y_i). A key observation is that **graph tasks fit into the i.i.d. framework of classical Machine Learning**. Indeed, the methodology of the previous section entirely applies here, defining $\mathcal{X}$ as (a subset of) the set of all graphs $\mathfrak{G}$. One can then define the expected ans empirical risks, learn with ERM, define the estimation and approximation errors, and so on.

As mentioned in the previous chapter, graph tasks still present unique difficulties, usually linked to the highly "irregular" nature of the space of graphs $\mathcal{X}$ and its inherent permutation-invariance. One must generally define appropriate *metrics* over $\mathcal{X}$ – an arduous task that is a research domain in and of itself; derive from it a notion of "regularity" or smoothness of prediction functions, from which one can compute Rademacher complexities [56] or other similar quantities, and so on. Approximation errors are then often linked to various notions of "expressivity", which as mentioned before bear similarities with the *graph isomorphism problem* [30]. Nevertheless, due to its similarities with the fundamentals of ML, many studies of generalization in graph ML focus exclusively on graph tasks [56, 104].

Node tasks and the need for statistical models. Unlike graph tasks, node tasks do **not** directly fit into the classical i.i.d. framework. In all generality, it is not even clear what quantity is "random": it might drastically change depending on the framework and the task considered. As mentioned in the introduction, node-tasks are often cast as Semi-supervised learning (SSL), where one observes both labelled and unlabelled data during training, with a distinction between transductive and inductive SSL.

Transductive SSL can be particularly simple for Graph ML: labelled nodes and unlabelled nodes are part of the same fixed graph, and one wants to "extend" the labels to the unlabelled nodes. In this case, the statistical modelling of the problem may be kept to a minimum, and the main "randomness" of the problem comes from the selection of labelled nodes (often sampled uniformly among all nodes).

For inductive SSL however, unlabelled nodes may be added to the graph in the future, or even be part of entirely new graphs. In this case, deriving prediction guarantees requires more detailed *statistical models* for the observed data, in order to actually get a meaningful link between training and test data: how does the graph grows? What are the links between training and test nodes, potentially across different graphs? Hence the use of statistical models for (large) graphs, that we coin under the general term *random graphs*.

> *As mentioned in the introduction, in my work (and this manuscript) I focus on node-tasks, for two main reasons: a) I am mostly interested in* large *graphs, for which virtually all tasks in the real world are node-tasks, and b) I find the departure from the i.i.d. framework to be fascinating and riddled with open questions.*

2.2.2 Risk for node tasks: a large graph approach

We now go back to graph ML, and (attempt to) take inspiration from regular ML to define a statistical framework to analyze node tasks. Recall that $\mathfrak{G} = \mathfrak{G}(\mathcal{Z})$ is the set of all graphs $G = (A, Z)$ with node features in $\mathcal{Z}$, and $\mathfrak{G}_n$ the set of all graphs of size n.

Functions Unlike in classical ML, the notion of *functions* is somewhat loose in graph ML on node tasks. Many studies will still attempt to manipulate "functions" from node features to labels and their corresponding function spaces, to try to use classical ML tools, but often in an unrigorous manner. For instance, [95] define a "function space" on nodes, computes its Rademacher complexity, then use an equivalent of Lemma 2.2 to link it to generalization, however their proof is incorrect in several ways: a) it is not a real function space, and b) Lemma 2.2 does not apply in this case (more precisely, the so-called "symmetrization lemma" [7, Chap 4] does not apply anymore). Hence, a good rule of thumb is that, rather than borrowing "shortcuts" to try to use classical ML results, one generally has to rigorously re-define all objects.

This error in the proof of [95] was identified by Y. Viegas during his internship.

In its most general from, a function $h(A, Z)$ for node tasks takes as input a graph $G = (A, Z)$, and returns an output $Y \in \mathsf{Y}^n$, that is, a label $y_i \in \mathsf{Y}$ per nodes. When $\mathsf{Y} \subset \mathbb{R}^{d_y}$ (as is generally the case), we have $Y \in \mathbb{R}^{n \times d_y}$. Note that we use the notation h here to distinguish from "regular" functions f in the previous section. As mentioned in the introduction, functions in graph ML satisfy two important conditions:

1. They can take input graphs *of any size* n, potentially not known in advance: this is for instance the case for GNNs whose number of parameters do not depend on the graph size. The output must of course be of the same size n.
2. They are permutation equivariant:

$$h(\sigma \cdot A, \sigma \cdot Z) = \sigma \cdot h(A, Z) \tag{2.18}$$

So formally speaking, the functions we consider belong to

$$\mathcal{H}_{all} := \Big\{ h : \mathfrak{G} \to \cup_{n=1}^{\infty} \mathsf{Y}^n, \\ \text{s.t. } \forall n, G \in \mathfrak{G}_n, \sigma \in \Sigma_n : \begin{cases} h(A,Z) \in \mathsf{Y}^n \\ h(\sigma \cdot A, \sigma \cdot Z) = \sigma \cdot h(A,Z) \end{cases} \Big\} \tag{2.19}$$

where we recall that Σ_n is the set of all permutation matrices of size n.

Remark 2.3. *In the definition above, to be technically correct, we must indeed state the constraint that inputs of size n produce outputs of size n. Note that this is different from the set $h \in \cup_n \{\mathfrak{G}_n \to \mathsf{Y}^n\}$, which contains functions that applies to only one particular size n. Here, functions must be able to take* any size n *as input, with the constraint that the output is of the same size.*

Empirical risk. Unlike classical ML, it is easier to start by defining the empirical risk that is actually minimized in practice, and see what notions of expected risk would be appropriate from there. Let us assume for simplicity that the training dataset is comprised of *one* graph $G = (A, Z)$ of size n with all its nodes labelled (see Remark 2.4 below). For $h \in \mathcal{H}_{all}$, one can then define the empirical risk as a sum of prediction errors over the nodes:

$$\hat{\mathcal{R}}(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(h(A,Z)_i, y_i) \tag{2.20}$$

Unlike classical ML, this is *not* a sum of i.i.d. variables, so its interpretation might be very different. However, this is the risk that is actually minimized in practice, following the ERM method.

Remark 2.4 (SSL)**.** *Following the SSL framework, if only a subset $V_{train} \subset [n]$ of nodes are labelled, the empirical risk is typically defined as*

$$\hat{\mathcal{R}}_{SSL}(h) = \frac{1}{|V_{train}|} \sum_{i \in V_{train}} \ell(h(A,Z)_i, y_i) \tag{2.21}$$

However, assuming that the indices in V_{train} are uniformly and independently from $[n]$, then it is easy to check that one has

$$\mathbb{E}_{V_{train}} \hat{\mathcal{R}}_{SSL}(h) = \hat{\mathcal{R}}_n(h) \tag{2.22}$$

Hence, for most of the mathematical properties that we are going to examine in this manuscript, the difference between the two is somewhat slightly irrelevant, and we will consider $\hat{\mathcal{R}}$ for simplicity. Of course, in practice it can make a big difference, and there is a whole field dedicated to transductive SSL, but we will mostly ignore it here.

Expected risk. One immediately notices that defining a notion of "true" risk is far from obvious for graph ML, as the empirical risk $\hat{\mathcal{R}}$ is not a sum of independent variables. In the absence of additional modellization, we may only consider a joint probability distribution $P_{G,Y}$ *over the set of all graphs* $\mathfrak{G}$ and labels, and take the expectation of the empirical risk:

$$\mathcal{R}(h) = \mathbb{E}_{(G,Y)\sim P_{G,Y}} \frac{1}{|G|} \sum_{i=1}^{|G|} \ell(h(A,Z)_i, y_i) \tag{2.23}$$

This is a very generic definition, that we will need to progressively refine. Assuming that the (unique) training graph (G,Y) is indeed distributed according to $P_{G,Y}$, the empirical risk (2.20) is an estimator of the expected risk (2.23) over *one* example: there is no reason why it should concentrate! We need to exploit some kind of "averaging" over the nodes, even though it is not clear how yet. For this, we will make progressively more assumptions on the graph distribution $P_{G,Y}$, up until LPMs in the next section.

First of all, a natural assumption over $P_{G,Y}$ is that it is itself permutation-invariant [5]: all permutations of a graph are equiprobable. In which case, we already have the following property.

Proposition 2.1. *If for all* n*, for all* (G,Y) *of size* n*, and for all* $\sigma \in \Sigma_n$*, it holds that* $(\sigma{\cdot}G, \sigma{\cdot}Y)$ *has the same distribution as* (G,Y)*, then for any* $h \in \mathcal{H}_{all}$

$$\mathcal{R}(h) = \mathbb{E}_{(G,Y)\sim P_{G,Y}} \ell(h(A,Z)_1, y_1) \tag{2.24}$$

In other words, for exchangeable $P_{G,Y}$, the node index i does not play any particular role in expectation, and the expected risk has a particularly simple form: given the whole graph, we measure the performance in expectation for a prediction on *one* node (arbitrarily, the first one). This is already a first step towards a more classical ML framework: the expected risk measures the prediction over *one* node in expectation, and the empirical risk is an empirical average over many nodes. Of course, the crucial differences here are that nodes are not i.i.d., and more importantly the function h does not act on a single node, but on the entire graph.

Sample size and sequence of risks. There is a crucial ingredient missing from this framework: we have no mechanisms to *study large dataset sizes* $n \to \infty$ and describe sample complexity, rates of convergence, and so on. As of yet, we only have a distribution over all graphs of all sizes, and one training graph drawn from this distribution, but no action lever to "let" the training graph grow. To handle this, we need a probabilistic framework to describe (large) graphs of any size, which coincidently is one of the main purposes of random graph models (see Chapter 1).

5. One will say that the random graph model is *exchangeable*.

In full generality, we thus work with a **collection of graph distributions**

$$\mathfrak{P} = \{P_n\}_{n\geqslant 1} \tag{2.25}$$

where $P_n = P_{G,Y|n}$ is a shorthand notation for a distribution over graphs G *of size* n and their labels. Given such a family, we denote by $\mathcal{R}_n$ the expected risk (2.24) with respect to P_n: for any $h \in \mathcal{H}_{all}$,

$$\mathcal{R}_n(h) = \mathbb{E}_{(G,Y)\sim P_n}\ell(h(A,Z)_1, y_1) \tag{2.26}$$

Note that our definition of the base space $\mathcal{H}_{all}$, where a single function can take graphs *of any size* as input, is crucial in this framework. By applying Prop. 2.1, if P_n is exchangeable this is equal to $\frac{1}{n}\mathbb{E}_{P_n}\sum_{i=1}^n \ell(h(A,Z)_i, y_i)$.

Assuming that the training graph is drawn from P_n, we may now study the approximation $\hat{\mathcal{R}} \approx \mathcal{R}_n$ when we let $n \to \infty$. However, this would technically only guarantee that training on a graph of size n generalizes to graphs of the same size, also drawn from P_n. We need an additional hypothesis to generalize *across graph sizes*, or, intuitively, we need to "relate" the distributions P_n between them.

Transferability and limit risk. The final ingredient to our framework is what researchers in graph ML have dubbed **transferability** [130, 102]: the capacity to train on one graph and generalize to graphs of different size (*a fortiori*, if both graphs are large enough). Here, we naturally materialize transferability by having $\mathcal{R}_n \approx \mathcal{R}_m$ for all n, m large enough, that is, the risks functions are close for all large graphs drawn from members of $\mathfrak{P}$. Thus, if we train on a graph from P_n and have $\hat{\mathcal{R}} \approx \mathcal{R}_n \approx \mathcal{R}_m$, we will generalize to graphs of size m.

By properties of Cauchy sequences, having $\mathcal{R}_n(h) \approx \mathcal{R}_m(h)$ is equivalent to having the risk *converge* (pointwise) when $n \to \infty$ to some limit quantity that we will denote by $\mathcal{R}_\infty(h)$. Since this convergence is not *a priori* guaranteed for any function $h \in \mathcal{H}_{all}$, we will thus *restrict the function space* to add this constraint:

$$\mathcal{H}_{all,\infty} = \mathcal{H}_{all,\infty}(\mathfrak{P}) := \{h \in \mathcal{H}_{all} \mid \exists \mathcal{R}_\infty(h) \in \mathbb{R}_+ \text{ s.t. } \mathcal{R}_n(h) \to \mathcal{R}_\infty(h)\} \tag{2.27}$$

In other words, this considers the functions h in $\mathcal{H}_{all}$ *such that* $\mathcal{R}_n(h)$ converges in $\mathbb{R}_+$. Note that this function space *depends on* the family of graph distributions [6] $\mathfrak{P} = \{P_n\}_n$, and might well be empty! For now, it is difficult to describe $\mathcal{H}_{all,\infty}$ in more details [7]. In particular, observe that the limit quantity $\mathcal{R}_\infty(h)$ has no reason to be a "simple", or even easily interpretable, mapping

6. although we will often omit the notation for simplicity

7. I haven't even be able to prove that it is a vector space... although I suspect that this is true and that the argument is probably simple, but highly dependent on the loss function

of the function h. It is just a *scalar* in $\mathbb{R}_+$ of which we suppose the existence, nothing else. Of course, we will see later examples of random graphs where $\mathcal{R}_\infty(h)$ has a friendly closed-form expression, and where the description of the overall space $\mathcal{H}_{all,\infty}$ is considerably simplified.

Excess risk decomposition. Henceforth, we will now consider that $\mathcal{R}_\infty$ plays our role of "true" risk, since it will approximate all expected risks over large graphs, and $\mathcal{H}_{all,\infty}$ to be our "base" space to seek out the optimal prediction function $h^\star$.

$$h^\star = \operatorname*{argmin}_{h \in \mathcal{H}_{all,\infty}} \mathcal{R}_\infty(h)\,. \tag{2.28}$$

Similar to the practical function space $\mathcal{F} \subset \mathcal{F}_{all}$ is ML, we consider prediction models [8] $\mathcal{H} \subset \mathcal{H}_{all,\infty}$, and derive a new excess risk decomposition: denoting as before $\hat{h} = \operatorname*{argmin}_{h \in \mathcal{H}} \hat{\mathcal{R}}(h)$, $h_{\mathcal{H}} = \operatorname*{argmin}_{h \in \mathcal{H}} \mathcal{R}_\infty(h)$, assuming that the training graph is drawn from P_n,

$$\begin{aligned}
&\mathcal{R}_\infty(\hat{h}) - \mathcal{R}_\infty(h^\star) \\
&\quad = \mathcal{R}_\infty(\hat{h}) - \mathcal{R}_n(\hat{h}) + \mathcal{R}_n(\hat{h}) - \hat{\mathcal{R}}(\hat{h}) + \underbrace{\hat{\mathcal{R}}(\hat{h}) - \hat{\mathcal{R}}(h_{\mathcal{H}})}_{\leqslant 0} \\
&\qquad + \hat{\mathcal{R}}(h_{\mathcal{H}}) - \mathcal{R}_n(h_{\mathcal{H}}) + \mathcal{R}_n(h_{\mathcal{H}}) - \mathcal{R}_\infty(h_{\mathcal{H}}) + \mathcal{R}_\infty(h_{\mathcal{H}}) - \mathcal{R}_\infty(h^\star) \\
&\quad \leqslant \underbrace{2 \sup_{h \in \mathcal{H}} |\mathcal{R}_n(h) - \mathcal{R}_\infty(h)|}_{\text{transferability error}} + \underbrace{2 \sup_{h \in \mathcal{H}} \left|\hat{\mathcal{R}}(h) - \mathcal{R}_n(h)\right|}_{\text{estimation error}} + \underbrace{\mathcal{R}_\infty(h_{\mathcal{H}}) - \mathcal{R}_\infty(h^\star)}_{\text{approximation error}}
\end{aligned} \tag{2.29}$$

The risk decomposition now involves three errors.

1. The estimation error, which is now between the empirical risk computed on a graph G and its expectation with respect to P_n.
2. The approximation error between $h_{\mathcal{H}} \in \mathcal{H}$ and $h^\star \in \mathcal{H}_{all,\infty}$, w.r.t. the limit risk $\mathcal{R}_\infty$.
3. And the novel term that we refer to as *transferability error*, which expresses the convergence from $\mathcal{R}_n(h)$ to $\mathcal{R}_\infty(h)$ *uniformly* over the function set $\mathcal{H}$.

At this point, we cannot really go further in the analysis without specifying further a distribution of graphs $\mathfrak{P}$. In the rest of the chapter, we introduce *Latent Position Models* (LPMs), a well-known general class of random graph models. We then revisit this framework for LPMs, which allows us to push the analysis considerably further and draw more intuitive connections between graph ML and classical ML.

8. again, in full generality $\mathcal{H}_{all,\infty}$ is defined by a somewhat cryptic notion of convergence, and it might not be easy to guarantee that actual prediction models belong to it. Fortunately, we will see that situation considerably simplifies for LPM random graphs, and that in particular GNNs are in $\mathcal{H}_{all,\infty}$

2.3 Latent Position Random Graphs

As we described in the introduction, Latent Positions Models (LPMs) are a general family of random graphs that include several well-known examples: Erdös-Rényi random graphs, Stochastic Block Models, graphons [9], random geometric graphs, and others. As we will see in the next sections, in graph ML, they also offer striking similarities to classical ML, with many non-trivial interesting intuitions. In this section, we introduce more in details LPMs and their main properties.

2.3.1 Basic definition

In (vanilla) LPMs, a graph with n nodes is generated as follows. One assumes that each node is equipped with an unobserved *latent variable* $x_i \in \mathcal{X} \subset \mathbb{R}^{d_x}$. They represent some embedding of the nodes in a high-dimensional space characterizing the underlying phenomenon responsible for the graph structure, for instance, unknown user preferences in a social network [65]. As we will see, the choice of notation x is done on purpose.

Then, for each pairs of nodes, *random edges* are drawn **independently** according to a distribution that is conditioned on the latent variables of the corresponding nodes, and potentially on the graph size n:

$$a_{ij} \sim P(\cdot \mid x_i, x_j, n) \tag{2.30}$$

This very generic notations allows for weighted or unweighted edges, random or deterministic edges, directed or undirected graphs... Recall that in this manuscript, we will deal exclusively with undirected graphs: random edges are generated for $i < j$, then $a_{ji} = a_{ij}$. The dependency on n allows for a control of the *sparsity* of the graph, that is, the expected number of edges with respect to n. We will come back to this notion later.

The distribution of edges is often represented through a symmetric function $W_n : \mathcal{X} \times \mathcal{X} \to [0, 1]$ referred to as *connectivity kernel*, whose expression may also depend on the graph size. In this manuscript, we will focus on two main cases: deterministic weighted edges, or random binary edges drawn as Bernoulli variables. That is,

$$a_{ij} = W_n(x_i, x_j) \qquad \text{(deterministic edges)} \tag{2.31}$$

$$a_{ij} \sim \text{Ber}(W_n(x_i, x_j)) \qquad \text{(random edges)} \tag{2.32}$$

where $\text{Ber}(p)$ are Bernoulli variables with parameter p. In the latter case, two nodes with latent variables x_i, x_j are connected with probability $W_n(x_i, x_j)$. Intuitively, the connectivity kernel W_n decreases when the distance between x_i and x_j increases, such that two nodes with similar

9. graphons actually refer to a specific, slightly different theory, see Remark 2.5

latent variables have a higher chance of being connected. Remark that the expectation of the random model (with respect to the edges) is the deterministic model.

Although some works consider that the latent variables x_i may be fixed (akin to fixed design in statistics), most of the time they will be considered i.i.d. according to some distribution, and this is the choice we adopt:

$$x_i \overset{\text{iid}}{\sim} P_x \tag{2.33}$$

2.3.2 Sparsity

As we have seen in the introduction, the sparsity (or, conversely, the density) of a graph refers to the ratio between the number of edges and number of nodes n. While sparsity is mostly an intuitive notion when talking about a fixed (real-world) graph, this can be made a precise definition when n grows to infinity and the graph is generated according to some model. In this context, a graph with $O(n^2)$ edges is said to be *dense*[10], while a graph with $O(n)$ edges is said to be *sparse*. All rates in-between are generally referred to as *relatively sparse*. The relatively sparse rate $O(n \log n)$ plays a particular role, as many phenomena can only be characterized above this rate (such as the emergence of the *giant component* in Erdös-Rényi models [48]).

Consider LPMs with random edges (2.32) and i.i.d. random variables. When the kernel $W_n = W$ does not depend on n, the generated graphs are dense in expectation: the expected number of edges is in $O(n^2)$. When W_n depends on n, sparsity is usually handled in two main ways. One possibility is to introduce a multiplicative factor called the *sparsity level* $\alpha_n \in [0, 1]$:

$$W_n(x_i, x_j) = \alpha_n w(x_i, x_j) \tag{2.34}$$

where w is a fixed kernel. Here, the expected number of edges is in $O(\alpha_n n^2)$, such that $\alpha_n = 1/n$ generates sparse graph, and $\alpha_n n \to \infty$ with $\alpha_n = o(1)$ generates relatively sparse graphs. The special rate $n \log n$ is obtained with $\alpha_n \sim (\log n)/n$. This is a model often used when representing sparse social networks or networks with community structures, such as SBMs [83].

The second main way of introducing sparsity in LPMs is through a varying *bandwidth* $\sigma_n > 0$: assuming $\mathcal{X} \subset \mathbb{R}^{d_x}$,

$$W_n(x_i, x_j) = w(x_i/\sigma_n, x_j/\sigma_n) = \tilde{w}\left(\frac{x_i - x_j}{\sigma_n}\right) \tag{2.35}$$

where $w : \mathbb{R}^{d_x} \to [0, 1]$ is a function that typically decreases at infinity. The smaller the bandwidth, the more "narrow" the kernel W_n is, such that nodes must be increasingly close to each other to be connected. Under some mild hypotheses (e.g. when $\tilde{w}$ is smooth), the expected num-

10. remember that the number of all possible edges is $n(n-1)/2 = O(n^2)$

ber of edges is then [11] in $O(n^2\sigma_n^{d_x})$, such that sparse graphs are obtained with $\sigma_n = (1/n)^{1/d_x}$. This is a model often used in applications involving physical networks with spatial locations, such as electrical or road networks, or meshes [121]. These two strategies yield very different mathematical properties when n goes to infinity.

> *In the rest of the manuscript, I use the multiplicative factor sparsity level* (2.34)*, which is the choice adopted in the majority of my works [NK12, NK14, NK15, NK16, NK13]. Occasionally, I obtained results for kernels with varying bandwidth [NK8, NK3], but they will not be presented here.*

2.3.3 Examples of LPMs

Classical examples of LPMs include:

1. **Erdös-Rényi** [48]: when the kernel $W_n = p_n$ is constant, we obtain the classical Erdös-Rényi model, where each edge is drawn with probability p_n.
2. **Stochastic Block Model** (SBM) [66]: SBMs model communities, where the probabilities of connection between two nodes only depend on their communities. SBMs can be obtained is two ways: either the set $\mathcal{X}$ is finite, in which case the latent variables are directly the community labels, or the kernel W_n is constant-by-part, and each constant represents one community. Sparsity is typically modelled with a multiplicative factor (2.34).
3. **Gaussian kernel** [NK9]: a typical kernel is the Gaussian kernel $W(x, x') = e^{-\frac{1}{2}\|x-x'\|^2_{\Sigma^{-1}}}$ for some "covariance" matrix Σ. Sparsity can be of both types (2.34), (2.35).
4. **random geometric graphs** [121]: They are also called *ε-graphs*. Node are connected if some metric $\|x - x'\|$ is less then a threshold: $W(x, x') = 1_{\|x-x'\|\leqslant\varepsilon}$. Sparsity is generally obtained with varying bandwidth (2.35), which corresponds to a decreasing $\varepsilon_n = \sigma_n\varepsilon$.
5. **Dot-product kernel** [3]: corresponds to some $W(x, x') = f(\varphi(x)^\top\varphi(x'))$. A typical example is to take $W(x, x') = f(x^\top x')$ and $\mathcal{X}$ the d_x-dimensional sphere: if P_x is the uniform distribution on the sphere [12], one can then deploy the powerful theory of *spherical harmonics* to analyze the model. Sparsity is not often considered in this literature, but by default it would be obtained with a multiplicative factor (2.34).
6. **Graphons** [92]: obtained when P_x is the uniform distribution over $[0, 1]$. See remark 2.5 below.

Remark 2.5 (Graphon and Machine Learning)**.** *Under this form, dense LPMs bear connections with graph limits called* graphons *[92]. The full theory of graphons is however somewhat more general: they can be defined using only notions of convergence in a well-chosen metric (the so-called* cut metric*) when n goes to infinity. However, the typical random graph example of graphon*

11. this is easily shown with a change of variables
12. which is however not very realistic for ML applications

adopted in this community is indeed dense LPMs with $P_x = \mathcal{U}([0,1])$ *and* $W_n = W$ *independent of* n*.*

With such a simple distribution P_x*, graphon theory thus delegates all the complexity of the model on* W*. On the contrary, in ML the distribution* P_x *will typically be complicated and (very)* high-dimensional*, as it represents real quantities, while the kernel* W *will often be quite simple, typically the Gaussian kernel. While some mathematical equivalencies between the two models exist, the points of view and intuitions are quite different. In particular, the* dimension *of the latent variables* d_x *often plays an important role in our theorems.*

2.3.4 Node features and node labels

Up until this point, we have seen how LPMs can generate *graph structures*. However, as described in the introduction, *node features* are equally important in graph Machine Learning. Unfortunately, traditional LPMs (and random graph models overall) do not include node features. Hence, the current ML literature adopt a variety of choices to obtain node features, but there is not real consensus yet and it remains an active literature. A popular choice, which I often adopt in my work, is to draw node features independently according to a distribution conditioned on individual latent variables for each node:

$$z_i \sim P_z(\cdot \mid x_i) \tag{2.36}$$

This is for instance the case of the popular Contextual SBM (CSBM) [39] model for community graphs, where node features are drawn according to different Gaussian distributions depending on their community, and the graph structure is an SBM. A simplified choice that we will often make in later chapters is to take a deterministic function $z_i = f_z(x_i)$. Typically, such a function is not injective and **does lose information** between x_i and z_i, such that it is not possible to entirely deduce the graph structure from the node features, and the two indeed represent complementary sources of information.

Similarly, the node labels y_i will be drawn independently:

$$y_i \sim P_y(\cdot \mid x_i) \tag{2.37}$$

Equivalently, the triplets (x_i, z_i, y_i) are generated i.i.d. according to a joint distribution P_{xyz}, and y and z are conditionally independent given x. The graph structure A, in turn, is generated using the collection of x_i's. This is illustrated in Fig. 2.1.

2.3.5 Learning on LPMs: intuitions

At the end of the day, a LPM is a pair $(P_{xyz}, \{W_n\}_{n\geqslant 1})$, such that y and z are conditionally independent w.r.t. x, and $\{W_n\}_{n\geqslant 1}$ is a collection of kernels that generates either deterministic or random edges (this will be precised for each result, or if not, both cases are covered) for graphs of size n. For convenience, we will now let $P_n = P_{X,A,Z,Y|n}$ denote the corresponding LPM distribution over graphs of size n and their labels, but *also their latent variables*, keeping in mind that they are unobserved. Naturally, a LPM corresponds to a family of graph distribution $\mathfrak{P} = \{P_n\}_{n\geqslant 1}$ as defined in the previous section. We will sometimes abusively denote $\mathfrak{P} = (P_{xyz}, W_n)$ by the parameters of the LPM instead.

All along this section, we have purposely used notations reminiscent of classical ML. Indeed, LPMs allow for very interesting connections between the two worlds. To summarize (see Fig. 2.1):

1. In ML, one observes pairs of variables

$$(x_i, y_i) \overset{\text{iid}}{\sim} P_{xy}$$

   and aim to derive rules to predict y from x.

2. In node-tasks GML on LPMs, there are

$$(x_i, y_i, z_i) \overset{\text{iid}}{\sim} P_{xyz}$$

   where y and z are conditionally independent given x. However **the x_i are not directly observed**, and instead one observes:

   (a) the z_i, which typically contains only *partial* information about the x_i

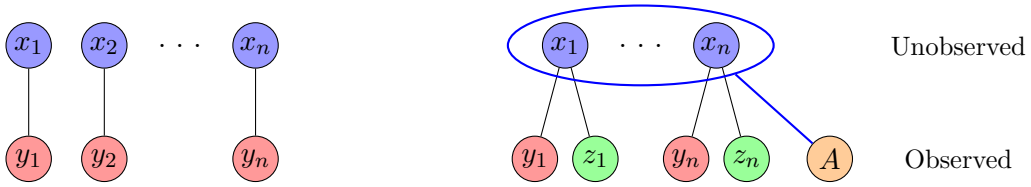
   (b) the graph structure A, which roughly contains notions of "similarities" between the x_i, albeit through an unknown kernel and with additional randomness on the edges

   From these **two different sources of information**, one aim to derive rules to predict the y_i. The question then becomes: how much does the information contained in the graph allow for "recovering" some of the information lost between the x_i and the z_i?

Graph ML is harder than regular ML. The intuition above suggest that graph ML on LPMs is "more difficult" than regular ML: the crucial quantities are hidden. Can this be made rigorous in our mathematical framework, with our definition of the risks $\mathcal{R}_n$ and limit risk $\mathcal{R}_\infty$?

Recall from the previous section (Remark 2.1) that the optimal risk in ML is the Bayes risk:

$$\mathcal{R}^\star_{\text{ML}} = \inf_{f\in\mathcal{F}_{all}} \mathcal{R}_{\text{ML}}(f) = \mathbb{E}_x \inf_{\hat{y}\in\mathsf{Y}} \mathbb{E}_{y|x}\ell(\hat{y}, y)$$



(a) In ML, one observes a collection of i.i.d. training data (x_i, y_i), and aim to derive a rule to predict y from x.

(b) In graph ML on LPMs, the x_i are not observed, but only partially known through the node features z_i's and the graph structure A. The latter depends on the *whole* collection of x_i's. The goal is to exploit both the z_i and A to derive prediction rules for the y_i, potentially in a new graph at test time.

Figure 2.1 – Statistical representation of ML (left) and graph ML on LPMs (right).

and that when it exists, one Bayes estimator $f^\star_{\mathrm{ML}}$ is

$$f^\star_{\mathrm{ML}}(x) \in \operatorname*{argmin}_{\hat{y}\in\mathsf{Y}} \mathbb{E}_{y|x}\ell(\hat{y}, y)\,.$$

Then we have the following proposition.

Proposition 2.2. *Let $\mathfrak{P}$ be a LPM. For all $h \in \mathcal{H}_{all}$ and all n, we have*

$$\mathcal{R}_n(h) \geqslant \mathcal{R}^\star_{\mathrm{ML}} \tag{2.38}$$

In particular, for $h \in \mathcal{H}_{all,\infty}$ we have $\mathcal{R}_\infty(h) \geqslant \mathcal{R}^\star_{\mathrm{ML}}$, and

$$\mathcal{R}^\star_\infty = \inf_{h\in\mathcal{H}_{all,\infty}} \mathcal{R}_\infty(h) \geqslant \mathcal{R}^\star_{\mathrm{ML}} \tag{2.39}$$

Hence, since the x_i's are unknown, graph ML is indeed "more difficult" than regular ML, in the sense that the optimal risk will always be higher. Of course, here we are only talking about intrinsic difficulty of the problem as materialized by the optimal risk and Bayes estimator, not all the different rates that will actually come into play in the risk decomposition (2.29).

An immediate but interesting corollary of the proposition above is the following.

Corollary 2.1. *Let $\mathfrak{P}$ be a LPM* ***for which the latent variables are observed****: $Z = X$, and assume that the Bayes predictor $f^\star_{\mathrm{ML}}$ exists. Then, the optimal fonction $h^\star \in \mathcal{H}_{all}$ is*

$$h^\star(A, X) = [f^\star_{\mathrm{ML}}(x_i)]_{i=1}^n$$

The proof is immediate since $\mathcal{R}_n(h^\star) = \mathcal{R}^\star_{\mathrm{ML}}$ reaches the lower bound for any n.

In other words, the optimal $h^\star$ returns $f^\star_{\mathrm{ML}}(x_i)$ independently for each node and *completely ignores the graph structure*! Which intuitively makes sense, as it contains no additional informa-

tion when the x_i's are known. So the core difficulty in graph ML is indeed the *discrepancy of information* between the observed node features and graph structure, and the unobserved latent variables that produced them.

2.4 Learning on LPMs

As we will now see, the framework of LPMs allows to push significantly further the analysis of Section 2.2. In particular: we will obtain more explicit expression for $\mathcal{R}_\infty$ and $\mathcal{H}_{all,\infty}$, derive a generic framework to analyze the transferability and estimation error, and relate the approximation error to more classical notions of functional approximations. As the reader might have guessed in light of the previous section, this mostly rely upon the presence of the latent variables and the latent space $\mathcal{X}$. This allows us to draw connections with more natural notions of *functions from $\mathcal{X}$ to* Y, the central objects in regular ML.

In the rest of this section, $\mathfrak{P} = (P_{xyz}, W_n)$ is a LPM.

2.4.1 $\mathfrak{P}$-convergence

As we have seen, a central tool of our framework is the somewhat cryptic notion of convergence (2.27) defining $\mathcal{H}_{all,\infty}$ (that is, it is the set of functions such that $\mathcal{R}_n(h)$ converges to *some* scalar $\mathcal{R}_\infty(h)$), which naturally leads to the risk decomposition (2.29). While $\mathcal{H}_{all,\infty}$ is hard to express in general, for LPMs our analysis will rely on a substantial simplification brought by a more natural notion of convergence towards functions *on the latent space*. We call this notion **$\mathfrak{P}$-convergence**. We recall that we have defined a metric d_{Y} on Y.

Definition 2.1 ($\mathfrak{P}$-convergent set of functions)**.** *Let $h \in \mathcal{H}_{all}$. If there exists $f \in \mathcal{F}_{all}$ such that*

$$\mathbb{E}_{P_n} d_{\mathsf{Y}}(h(A,Z)_1, f(x_1)) \leqslant r_n \xrightarrow[n\to\infty]{} 0 \tag{2.40}$$

*we say that h is **$\mathfrak{P}$-convergent** towards f with rate r_n.*

For an entire set of $\mathfrak{P}$-convergent functions $\mathcal{H} \subset \mathcal{H}_{all}$, we denote by $\mathcal{F}_{\mathcal{H}}$ the set of limit functions:

$$\mathcal{F}_{\mathcal{H}} = \mathcal{F}_{\mathcal{H}}(\mathfrak{P}) := \{f \mid \exists h \in \mathcal{H} \text{ which is } \mathfrak{P}\text{-convergent towards } f\} \tag{2.41}$$

*We say that $\mathcal{H}$ **is $\mathfrak{P}$-convergent towards** $\mathcal{F}_{\mathcal{H}}$. If all functions in $\mathcal{H}$ are $\mathfrak{P}$-convergent with a (maximum) rate of $r_n^{(\mathcal{H})}$, we say that $\mathcal{H}$ is $\mathfrak{P}$-convergent with rate $r_n^{(\mathcal{H})}$.*

That is, h is $\mathfrak{P}$-convergent if in expectation the prediction $h(A,Z)_1$ at the node 1 (or, by exchangeability, at any node i) converges to a function *of the latent variable* at this node (which we recall is unobserved). For instance, if $z_i = f_0(x_i)$, then the identity function $h(A,Z) = Z$

is immediately $\mathfrak{P}$-convergent towards f_0 (with rate 0). But of course the interest lies in more complex functions such as GNNs (see Chapter 3).

Remark that $\mathcal{F}_{\mathcal{H}} \subset \mathcal{F}_{all}$ are functions on the latent space whose definition *depends on* $\mathfrak{P}$, which is not the case of $\mathcal{H}$ itself. Intuitively, a function $h \in \mathcal{H}$ exists outside of all context (e.g. a GNN that can be applied to any graph), while $f \in \mathcal{F}_{\mathcal{H}}$ depends on a particular LPM $\mathfrak{P}$ (as we will see in the next Chapter, a "continuous GNN" applied to $\mathfrak{P}$). Also note that limit functions of $\mathfrak{P}$-convergence are unique only P_x-a.e., but we will not dwell on such technicalities here.

In later chapters, we may need notions of $\mathfrak{P}$-convergence with different metrics, mostly for technical convenience.

Definition 2.2 ($\mathfrak{P}$-convergence of order q)**.** *Let $h \in \mathcal{H}_{all}$ and $q \geqslant 1$. If there exists $f \in \mathcal{F}_{all}$ such that*

$$\mathbb{E}_{P_n}\left(\frac{1}{n}\sum_{i=1}^{n} d_{\mathsf{Y}}(h(A,Z)_i, f(x_i))^q\right)^{1/q} \leqslant r_n \xrightarrow[n\to\infty]{} 0 \tag{2.42}$$

*we say that h is $\mathfrak{P}$-convergent **of order** q towards f with rate r_n.*

By Prop. 2.1, regular $\mathfrak{P}$-convergence (Def. 2.1) is indeed $\mathfrak{P}$-convergence of order 1. A simple application of Hölder's inequality shows that for any $q \geqslant 1$, $\mathfrak{P}$-convergence of order q implies $\mathfrak{P}$-convergence of order 1. In the rest of this chapter, we will stick with $\mathfrak{P}$-convergence of order 1, and will only use order q in the next chapter.

$\mathfrak{P}$-convergence and limit risk. The next theorem is simple but key: it shows that $\mathfrak{P}$-convergence implies convergence of the risk, and therefore provides a wealth of functions that are in $\mathcal{H}_{all,\infty}$. Moreover, it provides a closed-form expression for the limit risk $\mathcal{R}_\infty$.

Theorem 2.1 ($\mathfrak{P}$-convergence and risk)**.** *Suppose that the loss satisfies Assumption 2.1. Let $h \in \mathcal{H}_{all}$ be $\mathfrak{P}$-convergent towards $f \in \mathcal{F}_{all}$. Then $h \in \mathcal{H}_{all,\infty}$, and*

$$\mathcal{R}_\infty(h) = \mathcal{R}_{\mathrm{ML}}(f)\,,$$

where $\mathcal{R}_{\mathrm{ML}}$ is the classical ML risk (2.5)*.*

In other words, a $\mathfrak{P}$-convergent function is *automatically* in $\mathcal{H}_{all,\infty}$, and **the limit risk $\mathcal{R}_\infty(h)$ is the ML risk $\mathcal{R}_{\mathbf{ML}}(f)$** of the $\mathfrak{P}$-convergent limit.

Thus $\mathfrak{P}$-convergence becomes a central notion when analyzing LPMs. It provides a practical way to prove that entire sets of functions $\mathcal{H}$ do indeed belong to $\mathcal{H}_{all,\infty}$, and it provides an actual, simple expression for the limit risk $\mathcal{R}_\infty$ as the regular ML risk for functions *on the latent space*. This is further confirmation that, conceptually, the latent variables do indeed play the role of x in regular ML. The limit function $f \in \mathcal{F}_{all}$ may then be analyzed with the usual tools of regular

ML: analyze their smoothness, bound the capacity of the set $\mathcal{F}_{\mathcal{H}}$, and so on. As we will see in the next subsection, we can entirely revisit the risk decomposition (2.29) under the lens of $\mathfrak{P}$-convergence.

Remark that $\mathcal{H}_{all,\infty}$ might be **larger** than the set of all $\mathfrak{P}$-convergent functions. In particular, an optimal function $h^\star \in \operatorname*{argmin}_{h\in\mathcal{H}_{all,\infty}} \mathcal{R}_\infty(h)$ may *not* be $\mathfrak{P}$-convergent, and we might not enjoy a simple expression of the optimal risk $\mathcal{R}^\star_\infty = \mathcal{R}_\infty(h^\star)$. Recall however that this optimal risk $\mathcal{R}_\infty(h^\star)$ is still always higher than $\mathcal{R}^\star_{\mathrm{ML}}$ by Prop. 2.2, so we have the immediate corollary:

Corollary 2.2. *Suppose that the loss satisfies Assumption 2.1 and the Bayes predictor $f^\star_{\mathrm{ML}}$ exists. If there is $h \in \mathcal{H}_{all}$ that is $\mathfrak{P}$-convergent towards $f^\star_{\mathrm{ML}}$, then $h = h^\star$ is the optimal prediction function on graphs.*

Of course, it is not realistic to expect to compute h to converge towards $f^\star_{\mathrm{ML}}$. However, this intuition will naturally be behind our handling of the *approximation error* in the next section, through notions of universality.

2.4.2 Generalization error on LPMs

We can now revisit the excess risk decomposition (2.29) with $\mathfrak{P}$-convergence. In this chapter we will still stay fairly general in our considerations: detailed specifications for GNNs will come in the next chapter.

2.4.2.1 Transferability error

The transferability error in (2.29) can be easily deduced from $\mathfrak{P}$-convergence with the following simple proposition.

Proposition 2.3 (Transferability and $\mathfrak{P}$-convergence)**.** *Assume that $\mathcal{H}$ is $\mathfrak{P}$-convergent towards $\mathcal{F}_{\mathcal{H}}$ with rate $r_n^{(\mathcal{H})}$, and that Assumption 2.1 holds on the loss. Then the transferability error in* (2.29) *satisfies*

$$\sup_{h\in\mathcal{H}} |\mathcal{R}_n(h) - \mathcal{R}_\infty(h)| \leqslant L r_n^{(\mathcal{H})} \tag{2.43}$$

where $L = L_\ell$ when the loss is Lipschitz and $L \lesssim \sup_h \mathbb{E}_{P_n} \|h(A,Z)_1\| + D_{\mathcal{Y}}$ when it is the square loss over a bounded domain.

Here we *do* need a rate $r_n^{(\mathcal{H})}$ for the whole set $\mathcal{H}$, since we are dealing with uniform convergence. Note that, for the square loss, the multiplicative constant involves $\sup_h \mathbb{E}_{P_n} \|h(A,Z)_1\|$ which might depend on n. In practice, we will see that it is generally uniformly bounded.

2.4.3 Approximation error

As mentioned before, bounding the approximation error is a complex topic, generally done on a case-by-case basis. With $\mathfrak{P}$-convergence, we will be able to relate the approximation error (2.29) to more classical notions of approximation for functions on $\mathcal{X}$. As mentioned before, for deep neural networks the basic property is *universality*, which we formally define below. We first define the following metric, for $f, g \in \mathcal{F}_{all}$:

$$d_{\mathsf{Y},\infty}(f,g) = \sup_x d_{\mathsf{Y}}(f(x), g(x)) \tag{2.44}$$

Definition 2.3 (Universality)**.** *Let $\mathfrak{F} = \{\mathcal{F}^R\}_{R>0}$ be a collection of function sets $\mathcal{F}^R \subset \mathcal{F}_{all}$ parameterized by some R, let $\mathcal{C} \subset \mathcal{F}_{all}$ be another set of functions. We say that $\mathfrak{F}$ is **universal** for $\mathcal{C}$ if: for any $f^\star \in \mathcal{C}$ and any $\varepsilon > 0$, there exists R_ε and $f_\varepsilon \in \mathcal{F}^{R_\varepsilon}$ such that*

$$d_{\mathsf{Y},\infty}(f_\varepsilon, f^\star) \leqslant \varepsilon \tag{2.45}$$

Typically, $\mathcal{F}^R = \{f_\theta \mid \Omega(\theta) \leqslant R\}$ is a set of parametric functions, where Ω is some measure of capacity of the parameter θ, generally its dimensionality and maximum amplitude. In classical universality theorems, for instance for deep neural nets, the set $\mathcal{C}$ are continuous functions. Also note that here we make the choice to use the classical supremum over $\mathcal{X}$ as the norm, which will be enough for our purposes, but various universality theorems can be obtained for different norms [NK13]. The following theorem relates universality for functions on $\mathcal{X}$ with the approximation error (2.29).

Theorem 2.2 (Approximation error of universal models)**.** *Let $\mathfrak{P}$ be a LPM, and $\mathcal{H}^R \subset \mathcal{H}_{all,\infty}$ be a family of models. Assume that for each R, the set $\mathcal{H}^R$ is $\mathfrak{P}$-convergent towards $\mathcal{F}_{\mathcal{H}^R}$, and that there is a set of functions $\mathcal{C}$ such that the collection $\mathfrak{F} = \{\mathcal{F}_{\mathcal{H}^R}\}_{R>0}$ is **universal** for $\mathcal{C}$. Suppose that the loss satisfies Assumption 2.1.*

Consider $h^\star \in \operatorname*{argmin}_{h \in \mathcal{H}_{all,\infty}} \mathcal{R}_\infty(h)$. If either:

1. *$h^\star$ is $\mathfrak{P}$-convergent towards f, and $f \in \mathcal{C}$, or*
2. *a Bayes estimator $f^\star_{\mathrm{ML}} \in \operatorname*{argmin}_{f \in \mathcal{F}_{all}} \mathcal{R}_{\mathrm{ML}}(f)$ exists and $f^\star_{\mathrm{ML}} \in \mathcal{C}$.*

Then for all $\varepsilon > 0$, there is R_ε such that, taking $h_{\mathcal{H}^{R_\varepsilon}} \in \operatorname*{argmin}_{h \in \mathcal{H}^{R_\varepsilon}} \mathcal{R}_\infty(h)$, the approximation error (2.29) *satisfies:*

$$\mathcal{R}_\infty(h_{\mathcal{H}^{R_\varepsilon}}) - \mathcal{R}_\infty(h^\star) \leqslant L\varepsilon\,, \tag{2.46}$$

where $L = L_\ell$ if the loss is L_ℓ-Lipschitz, or $L \lesssim \sqrt{\mathcal{R}_\infty(h^\star)} + D_{\mathcal{Y}}$ if the loss is the square loss and $\mathcal{Y}$ is bounded.

In other words, if either the optimal prediction function is $\mathfrak{P}$-convergent towards a function in $\mathcal{C}$, or the Bayes predictor belong to $\mathcal{C}$, then the approximation error can be bounded by leveraging the universality of $\mathcal{F}_{\mathcal{H}^R}$ in $\mathcal{C}$. As in classical ML, we will be able to show universality for some models and LPMs in the next chapter, but estimating the R_ε to obtain explicit rates will remain largely open.

2.4.4 Estimation error: a covering number approach

The material in this section has been developed during the internship of Y. Viegas.

Finally, we are left to deal with the estimation error, that is

$$\sup_{h\in\mathcal{H}} \left|\hat{\mathcal{R}}(h) - \mathcal{R}_n(h)\right|$$

As mentioned earlier, in ML there usually are systematic approaches to bound this quantity. In graph ML, adapting these methods to our framework is still quite open. We discuss below various potential routes, before introducing a simple covering numbers-based method that this quite generic but probably suboptimal.

1. **Covering numbers** [7, Sec. 4.4.4]: For a metric set (X, d), the *covering number* $\mathcal{N}(X, \varepsilon, d)$ is the minimum number of balls of radius ε required to cover it. In particular, a compact set has *finite* covering numbers (the converse is not entirely true up to some technicality).

   Using covering numbers is a very generic method to handle uniform concentration over a (generally compact) set, used in many domains besides ML. One first proves a pointwise concentration, then uniformly extend it using finite coverings of the set, stability of the various quantities, and a union bound. In ML, this is known to be applicable in broad situations, but to be suboptimal: one gets an additional factor $\log(n)$, and the dependence in the number of parameters is generally too high.

   As we will see below, for LPMs it is indeed possible to prove the required pointwise concentration with a simple bounded differences approach, and thus to apply this approach by covering numbers.

2. **Rademacher complexity** [7, Sec. 4.5]: Rademacher complexities work using a *symmetrization* approach, by remarking that, given $x_1, \dots, x_n, x'_1, \dots, x'_n$ iid random variables, then

   $$\mathbb{E}_{X,X'} \sup_f \left|\sum_i f(x_i) - f(x'_i)\right| = \mathbb{E}_{X,\varepsilon} \sup_f \left|\sum_i \varepsilon_i f(x_i)\right|$$

   where ε_i are independent Rademacher random variables (uniform on $\{-1, 1\}$). The quantity on the r.h.s. is called the *Rademacher complexity* of the function set, and it can then

be bounded. This approach no longer works for graph ML, since the terms of the empirical risk are no longer independent: the so-called "symmetrization lemma" is no longer valid [13]. One potential approach is to model the dependence between the terms and compute a so-called *fractional Rademacher complexity* [172], however this only works if the graph is fixed. In LPMs the graph structure vary with the node features, so this approach is also no longer valid. As of now Rademacher complexities for random graphs are still an open question.

3. **Detour through limit functions**: given the reasoning above, if h is $\mathfrak{P}$-convergent towards f, the expected risk converges. Then one may hope to express convergence *for the empirical risk* as well:
$$\left|\hat{\mathcal{R}}(h) - \hat{\mathcal{R}}_{\mathrm{ML}}(f)\right| \to 0$$
in the same manner that the transferability error expresses this convergence for the expected risk. Then, it would be possible to resort to more classical proofs in the world of $\hat{\mathcal{R}}_{\mathrm{ML}}$ and $\mathcal{R}_{\mathrm{ML}}$, e.g. with Rademacher complexities.

   It is indeed possible to prove such a convergence result with high probability, however *the concentration bound must also hold uniformly over* $h \in \mathcal{H}$. One may resort to covering numbers, which negates the gain and brings us back to the first approach. For GNNs, one hope would be to obtain this uniform concentration *via* the concentration of *graph operators* [129] without resorting to covering numbers, but this is still open.

Let us now detail a generic covering number approach for LPMs, based on bounded differences. The core of this approach is to assume that the empirical risk *does not vary too much* with each independent variable that constitute the random graph, then prove concentration with the classical Mc Diarmid's inequality. Then, covering numbers allow us to obtain a uniform bound.

Pointwise concentration. For all the considerations below, we consider $\mathcal{H} \subset \mathcal{H}_{all}$ and $\mathfrak{P}$ a LPM. Note that here we do not technically need that $\mathcal{H} \subset \mathcal{H}_{all,\infty}$, or that functions in $\mathcal{H}$ are $\mathfrak{P}$-convergent, but as we have seen this is our main approach for the other two terms in the risk decomposition (2.29). Our main bounded difference assumption is the following. It deviates slightly from the usual application of McDiarmid's inequality, in the sense that we assume that it is valid only for a set of graphs $\mathcal{G}_n \subset \mathfrak{G}_n$ that is generated with high probability: this will be useful in some examples in the next chapter. The version of McDiarmid's inequality that we use is therefore a non-uniform version (Thm 2.4 in Appendix).

Assumption 2.2. *Assume that, for all* n*, there is a set of graphs* $\mathcal{G}_n \subset \mathfrak{G}_n$ *of size* n *such that, for all* $h \in \mathcal{H}$ *and* $(A, Z), (A', Z) \in \mathcal{G}_n$*:*

13. despite what is wrongly claimed in some papers such as [95]

— *if A, A' differs on only one coordinate A_{ij},*

$$\sum_{k=1}^{n} d_{\mathcal{Y}}(h(A,Z)_k, h(A',Z)_k) \leqslant \Delta_n^a \tag{2.47}$$

— *if A, A' differs on one line $A_{i,:}$ and the corresponding column $A_{:,i}$ (since they are symmetric),*

$$\sum_{j=1}^{n} d_{\mathcal{Y}}(h(A,Z)_j, h(A',Z)_j) \leqslant \Delta_n^A \tag{2.48}$$

— *if Z, Z' differs on one sample $z_i \neq z_i'$,*

$$\sum_{j=1}^{n} d_{\mathcal{Y}}(h(A,Z)_j, h(A,Z')_j) \leqslant \Delta_n^z \tag{2.49}$$

Moreover, we will need the following global bounds on the loss, which can be quite loose. It is particularly required for the square loss, and corresponds to a high probability bound on $h(A,Z)$ itself.

Assumption 2.3. *Considering the same set $\mathcal{G}_n$ as in Ass. 2.2, assume that*

$$\sup_{y\in\mathcal{Y}} \max_i |\ell(h(A,Z)_i, y)| \leqslant \begin{cases} c_n^{(\ell)} & \text{if } (A,Z) \in \mathcal{G}_n \\ C_n^{(\ell)} & \text{otherwise} \end{cases} \tag{2.50}$$

Then we have the following pointwise concentration result.

Lemma 2.3. *Let $\mathcal{H}$ satisfying Assumption 2.2 and 2.3. Suppose that the loss satisfies Assumption 2.1. Denote by $p_n = P_n(\mathcal{G}_n^c)$ the probability of generating a graph of size n that is not in $\mathcal{G}_n$. For any $h \in \mathcal{H}$, with probability at least $1-\delta-p_n$, we have*

$$\left|\hat{\mathcal{R}}(h) - \mathcal{R}_n(h)\right| \lesssim (D_n + nL\tilde{\Delta}_n^a + C_n^{(\ell)})p_n + (D_n + \sqrt{n}L\tilde{\Delta}_n^a)\sqrt{\frac{\log(1/\delta)}{n}} \tag{2.51}$$

where $D_n = L_\ell(\Delta_n^A + \Delta_n^z) + c_n^{(\ell)}$ and $L = L_\ell$ for Lipschitz bounded loss, or $D_n = (c_n^{(\ell)} + D_{\mathcal{Y}})(\Delta_n^A + \Delta_n^z + D_{\mathcal{Y}})$ and $L = c_n^{(\ell)} + D_{\mathcal{Y}}$ for the square loss; and $\tilde{\Delta}_n^a = \Delta_n^a$ for random Bernoulli edges, or $\tilde{\Delta}_n^a = 0$ for deterministic edges.

As we will see, we can often design $\mathcal{G}_n$ to have p_n negligible in front of the other terms, and for many models we will even have $p_n = 0$. Moreover, we will generally have $D_n = O(1)$, and $\Delta_n^a = o(1/\sqrt{n})$. Thus, we will generally obtain a global rate in $O(1/\sqrt{n})$ for dense LPMs. We may also obtain a slower rate for sparser graphs, see next chapter for GNNs.

Uniform bound. To obtain a uniform bound, we proceed by covering numbers. Define the metric over $\mathcal{H}_{all}$:

$$d_{\mathsf{Y},n}(h,h') = \sup_{(A,Z)\in\mathfrak{G}_n} \max_{i=1,\dots,n} d_{\mathsf{Y}}(h(A,Z)_i, h'(A,Z)_i) \tag{2.52}$$

Assume that $\mathcal{H}$ has finite covering numbers $\mathcal{N}(\mathcal{H},\varepsilon,d_{\mathsf{Y},n})$, which is often the case for parametric functions with bounded parameters. Note that they may depend on n. Then we have the following.

Theorem 2.3 (Estimation error through covering numbers.)**.** *Suppose that Assumption 2.2 and 2.3 hold. Suppose that the loss satisfies Assumption 2.1. Denote by $p_n = P_n(\mathcal{G}_n^c)$, and $\mathcal{N} = \mathcal{N}(\mathcal{H}, E_n/\sqrt{n}, d_{\mathsf{Y},n})$, where E_n is defined below. With probability at least $1-\delta-\mathcal{N}p_n$, we have*

$$\sup_{f\in\mathcal{F}} \left|\hat{\mathcal{R}}(f) - \mathcal{R}_n(f)\right| \lesssim (D_n + nL\tilde{\Delta}_n^a + C_n^{(\ell)})p_n + (D_n + \sqrt{n}L\tilde{\Delta}_n^a)\sqrt{\frac{\log(\mathcal{N}/\delta)}{n}} \tag{2.53}$$

where $D_n = L_\ell(\Delta_n^A + \Delta_n^b) + c_n^{(\ell)}$, $L = L_\ell$ and $E_n = D_n/L_\ell$ for Lipschitz bounded loss, or $D_n = (c_n^{(\ell)} + D_{\mathcal{Y}})(\Delta_n^A + \Delta_n^z + D_{\mathcal{Y}})$, $L = c_n^{(\ell)} + D_{\mathcal{Y}}$ and $E_n = D_n/(D_{\mathcal{Y}} + C_n^{(\ell)})$ for the square loss; and $\tilde{\Delta}_n^a = \Delta_n^a$ for random Bernoulli edges, or $\tilde{\Delta}_n^a = 0$ for deterministic edges.

Hence, the uniform bound looses a factor $\log(\mathcal{N})$, compared to the pointwise one. For parametric functions, such covering numbers generally yields $\log(\mathcal{N}) \approx d_\Theta \log(n)$, where d_Θ is the dimensionality of the parameters. In ML, the $\log(n)$ term is generally seen as suboptimal, while d_Θ is a rather crude dependency on the parameters, which only express their dimensionality and ignores any structure they might exhibit. Note that in this version the probability of failure includes a term $\mathcal{N}p_n$, which asks for a very small $p_n \leqslant \delta/\mathcal{N}$ to be negligible. Again, this may be done for most models we will see later, and even $p_n = 0$ in many cases.

2.4.5 Summary

To summarize, given a family of prediction functions $\mathcal{H}^R \subset \mathcal{H}_{all}$ and an LPM $\mathfrak{P}$, one may bound the entire excess risk (2.29) as such:

1. For each R, show that $\mathcal{H}^R$ is $\mathfrak{P}$-convergent towards some $\mathcal{F}_{\mathcal{H}^R}$. This shows that $\mathcal{H}^R \subset \mathcal{H}_{all,\infty}$ so the limit risk $\mathcal{R}_\infty$ always exists, and directly yields a bound on the transferability error;
2. Evaluate the approximation power of $\mathcal{F}_{\mathcal{H}^R}$. If it is universal w.r.t. some $\mathcal{C}$ and the Bayes predictor is in $\mathcal{C}$, then it will vanish when R increases. Ideally, compute this rate w.r.t. R;
3. For each R, bound the estimation error by showing stability of the functions in $\mathcal{H}^R$ and applying a covering numbers approach.

In the next chapter, we will apply this method to GNNs, and particularly to S-GNNs.

2.5 Conclusion and discussion

I am well aware that the material in this chapter is quite rough around the edges: it is a bit fresh, unpolished, and not peer reviewed yet! Nevertheless, I do believe that it draws rather elegant connections between ML and GML on LPMs, both at a (very) technical level and at an intuitive level. While most of the properties "proven" in this chapter are quite tautologic, and consists in appropriate reformulations of classic ML definitions, it provides a general framework for node tasks that is both rigorous and flexible. We apply it to GNNs in the next chapter, but I hope that it can serve as a basis for future studies. Naturally, besides a general polishing of the framework that may happen over time, there are many questions left, of which I outline a few below.

Open questions

1. **Limit risk**: we have seen that $\mathfrak{P}$-convergence considerably simplifies the expression of $\mathcal{R}_\infty$ and $\mathcal{H}_{all,\infty}$, nevertheless many basic properties of these objects are still open: is $\mathcal{H}_{all,\infty}$ a vector space? Could they be related to more generic notions of convergence of graphs with appropriate metrics, such as graphons with the *cut-metric* [92]?
2. **Random graph models**: LPMs provide a powerful framework, and most of the properties and intuitions that I have described entirely rely on their latent variables mechanism. Nevertheless, they could probably be improved, while keeping the same basics: in particular, the generation of node features and labels is quite simplistic for now, as well as the independency of all the edges. At longer term, the use of completely different graph models (e.g. preferential attachement [9]) is still completely open, and the absence of latent variables would drastically change the intuitions behind our large-graph framework.
3. **Rademacher and other approaches**: we have described a covering numbers-based proof, as most of the ingredients for other approaches such as Rademacher complexities (aka the symmetrization lemma) do not work anymore. Fixing this is an interesting, open question.

Appendix

2.A Proofs

Proof of Prop. 2.1. Considering that the graph size is random, we can decompose $\mathbb{E}_{G,Y} = \mathbb{E}_n \mathbb{E}_{G,Y|n}$, where the second expectation is over the distribution of graphs of size n. Taking σ_i the permutation that exchange i and 1, since $(\sigma_i A \sigma_i^\top, \sigma_i Z, \sigma_i Y)$ has the same distribution as (A, Z, Y) and h is equivariant,

$$\begin{aligned}
\mathcal{R}(h) &= \mathbb{E}_{G,Y} \frac{1}{|G|} \sum_{i=1}^{|G|} \ell(h(A,Z)_i, y_i) = \mathbb{E}_n \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{A,Z,Y|n} \ell(h(A,Z)_i, y_i) \\
&= \mathbb{E}_n \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{A,Z,Y|n} \ell(h(\sigma_i A \sigma_i^\top, \sigma_i Z)_i, [\sigma_i Y]_i) = \mathbb{E}_n \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{A,Z,Y|n} \ell([\sigma_i h(A,Z)]_i, [\sigma_i Y]_i) \\
&= \mathbb{E}_n \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{A,Z,Y|n} \ell(h(A,Z)_1, y_1) = \mathbb{E}_{G,Y} \ell(h(A,Z)_1, y_1)
\end{aligned}$$

□

Proof of Prop. 2.2. This is a simple computation: since y_1 only depends on x_1,

$$\begin{aligned}
\mathcal{R}_n(h) &= \mathbb{E}_{P_n} \ell(h(A,Z)_1, y_1) = \mathbb{E}_X \mathbb{E}_{A,Z|X} [\mathbb{E}_{y_1|x_1} \ell(h(A,Z)_1, y_1)] \\
&\geqslant \mathbb{E}_X \mathbb{E}_{A,Z|X} \inf_{\hat{y} \in \mathsf{Y}} [\mathbb{E}_{y_1|x_1} \ell(\hat{y}, y_1)] \\
&= \mathbb{E}_{x_1} \inf_{\hat{y} \in \mathsf{Y}} [\mathbb{E}_{y_1|x_1} \ell(\hat{y}, y_1)] = \mathcal{R}^\star_{\mathrm{ML}}
\end{aligned}$$

□

Proof of Thm. 2.1. When the loss is Lipschitz, the proof is immediate by remarking that for any n, $\mathbb{E}_{x,y} \ell(f(x), y) = \mathbb{E}_{P_n} \ell(f(x_1), y_1)$ and therefore

$$\begin{aligned}
|\mathcal{R}_n(h) - \mathbb{E}\ell(f(x), y)| &= |\mathbb{E}_{P_n} \left(\ell(h(A,Z)_1, y_1) - \ell(f(x_1), y_1)\right)| \\
&\leqslant L_\ell \mathbb{E}_{P_n} d_{\mathsf{Y}}(h(A,Z)_1, f(x_1)) \leqslant L_\ell r_n
\end{aligned}$$

When the loss is the square loss, remark that

$$|\mathbb{E}_{P_n} \left(\ell(h(A,Z)_1, y_1) - \ell(f(x_1), y_1)\right)| \leqslant \left(\mathbb{E} \left\|h(A,Z)_1\right\| + \mathbb{E} \left\|f(x)\right\| + 2D_{\mathcal{Y}}\right) r_n$$

The only quantity that depends on n in the multiplicative constant is $\mathbb{E} \left\|h(A,Z)_1\right\|$, and

$$\mathbb{E} \left\|h(A,Z)_1\right\| \leqslant \mathbb{E} \left\|h(A,Z)_1 - y_1\right\| + D_{\mathcal{Y}} \leqslant \sqrt{\mathcal{R}_n(h)} + D_{\mathcal{Y}}$$

Since $\mathcal{R}_n(h)$ converges, it is uniformly bounded in n, which concludes the proof. □

Prof of Prop. 2.3. The Lipschitz case has already been proved in the proof of Thm. 2.1. For the square loss, we recall that we have

$$|\mathbb{E}_{P_n}\left(\ell(h(A,Z)_1, y_1) - \ell(f(x_1), y_1)\right)| \leqslant \left(\mathbb{E}\,\|h(A,Z)_1\| + \mathbb{E}\,\|f(x)\| + 2D_{\mathcal{Y}}\right) r_n$$

and $\mathbb{E}\,\|f(x)\| \leqslant \mathbb{E}\,\|h(A,Z)_1\| + r_n^{(\mathcal{H})} \lesssim \mathbb{E}\,\|h(A,Z)_1\|$. □

Proof of Thm. 2.2. Let $\tilde{f} \in \mathcal{C}$ be either $f^\star$ or $f^\star_{\mathrm{ML}}$ depending on hypothesis 1) or 2) above. In both cases, we have $\mathcal{R}_\infty(h^\star) \geqslant \mathcal{R}_{\mathrm{ML}}(\tilde{f})$: in the first case, this is an equality by Thm. 2.1, and in the second, this is shown by Prop. 2.2.

Take $\varepsilon > 0$. Since $\mathfrak{F} = \{\mathcal{F}_{\mathcal{H}^R}\}_{R>0}$ is universal, there exists R_ε and $f_\varepsilon \in \mathcal{F}_{\mathcal{H}^{R_\varepsilon}}$ such that

$$d_{\mathsf{Y},\infty}(f_\varepsilon, \tilde{f}) \leqslant \varepsilon$$

Since $\mathcal{H}^{R_\varepsilon}$ is $\mathfrak{P}$-convergent towards $\mathcal{F}_{\mathcal{H}^{R_\varepsilon}}$, there exists $h_\varepsilon \in \mathcal{H}^{R_\varepsilon}$ that is $\mathfrak{P}$-convergent towards f_ε. By minimality of $h_{\mathcal{H}^{R_\varepsilon}}$ and Proposition 2.1, we have

$$\mathcal{R}_\infty(h_{\mathcal{H}^{R_\varepsilon}}) \leqslant \mathcal{R}_\infty(h_\varepsilon) = \mathcal{R}_{\mathrm{ML}}(f_\varepsilon) \tag{2.54}$$

Hence the approximation error satisfies

$$\mathcal{R}_\infty(h_{\mathcal{H}^{R_\varepsilon}}) - \mathcal{R}_\infty(h^\star) \leqslant \mathcal{R}_{\mathrm{ML}}(f_\varepsilon) - \mathcal{R}_{\mathrm{ML}}(\tilde{f})$$

If the loss is L_ℓ Lipschitz, we immediately have $\mathcal{R}_{\mathrm{ML}}(f_\varepsilon) - \mathcal{R}_{\mathrm{ML}}(\tilde{f}) \leqslant L_\ell d_{\mathsf{Y},\infty}(f_\varepsilon, \tilde{f}) \leqslant L_\ell \varepsilon$.

If ℓ is the square loss and $d_{\mathcal{Y}}$ is the Euclidean norm, we have

$$\mathcal{R}_{\mathrm{ML}}(f_\varepsilon) - \mathcal{R}_{\mathrm{ML}}(\tilde{f}) \leqslant \left(\mathbb{E}\,\|f_\varepsilon(x)\| + \mathbb{E}\left\|\tilde{f}(x)\right\| + 2D_{\mathcal{Y}}\right) d_{\mathsf{Y},\infty}(f_\varepsilon, \tilde{f})$$

and

$$\mathbb{E}\left\|\tilde{f}(x)\right\| \leqslant \mathbb{E}\left\|\tilde{f}(x) - y\right\| + D_{\mathcal{Y}} \leqslant \sqrt{\mathcal{R}_{\mathrm{ML}}(\tilde{f})} + D_{\mathcal{Y}} \leqslant \sqrt{\mathcal{R}_\infty(h^\star)} + D_{\mathcal{Y}}$$

and $\mathbb{E}\,\|f_\varepsilon(x)\| \leqslant \varepsilon + \mathbb{E}\left\|\tilde{f}(x)\right\|$, which allows us to conclude.

□

Proof of Lemma 2.3. Recall that the collection of independent variables involved in P_n are

(x_i, z_i, y_i) for $1 \leqslant i \leqslant n$, and U_{ij}, $i < j$ in case of random edges. Our goal is to apply a generalized version of McDiarmid's inequality [33] (Thm. 2.4).

Under the event that (A, Z) and (A', Z') both belong to $\mathcal{G}_n$, when there is an index j such that (x_j, y_j, z_j) varies: the node features change by one component – ie, Z and Z' are different only on index j, same thing for the labels Y, Y', while the adjacency matrix A' varies from A by column and line j. Hence

$$\begin{aligned}
&\frac{1}{n}\left|\sum_i \ell(h(A,Z)_i, y_i) - \sum_i \ell(h(A',Z')_i, y'_i)\right| \\
&\leqslant \frac{1}{n}\Big(\left|\sum_i \ell(h(A,Z)_i, y_i) - \ell(h(A',Z)_i, y_i)\right| + \left|\sum_i \ell(h(A',Z)_i, y_i) - \ell(h(A',Z')_i, y_i)\right| \\
&\quad + \left|\ell(h(A',Z')_j, y_j) - \ell(h(A',Z')_j, y'_j)\right| \Big)
\end{aligned}$$

When the loss is L_ℓ-Lipschitz, it results in

$$\begin{aligned}
&\frac{1}{n}\left|\sum_i \ell(h(A,Z)_i, y_i) - \sum_i \ell(h(A',Z')_i, y'_i)\right| \\
&\leqslant \frac{1}{n}\left(L_\ell \sum_i d_{\mathcal{Y}}(h(A,Z)_i, h(A',Z)_i) + L_\ell \sum_i d_{\mathcal{Y}}(h(A',Z)_i, h(A',Z')_i) + 2c_n \right) \\
&\leqslant c_i := \frac{1}{n}\left(L_\ell(\Delta_n^A + \Delta_n^z) + 2c_n^{(\ell)} \right)
\end{aligned}$$

In case of random edges, a variation in U_{ij} results in the same manner to a variation of $c'_{ij} := \frac{L_\ell \Delta_n^a}{n}$. We compute $\sum_i c_i + \sum_{ij} c'_{ij}$ and $\sqrt{\sum_i c_i^2 + \sum_{ij}(c'_{ij})^2}$ and applying Thm. 2.4 we obtain the result.

When the loss is the square loss and $d_{\mathcal{Y}}$ is the Euclidean norm, since $\max_i \|h(A,Z)_i\| \leqslant c_n^{(\ell)} + D_{\mathcal{Y}}$ for graphs in $\mathcal{G}_n$ by Assumption 2.2,

$$\begin{aligned}
&\frac{1}{n}\left|\sum_i \ell(h(A,Z)_i, y_i) - \sum_i \ell(h(A',Z')_i, y'_i)\right| \\
&\leqslant \frac{1}{n}\Big(2(c_n^{(\ell)} + D_{\mathcal{Y}}) \sum_i \left\|h(A,Z)_i - h(A',Z)_i\right\| \\
&+ 2(c_n^{(\ell)} + D_{\mathcal{Y}}) \sum_i \left\|h(A',Z)_i - h(A',Z')_i\right\| + 4(c_n^{(\ell)} + D_{\mathcal{Y}}) D_{\mathcal{Y}} \Big) \\
&\leqslant c_i := \frac{2(c_n^{(\ell)} + D_{\mathcal{Y}})}{n}\left(\Delta_n^A + \Delta_n^z + 2D_{\mathcal{Y}} \right)
\end{aligned}$$

In case of random edges, a variation in U_{ij} results in the same manner to a variation of $c'_{ij} := \frac{2(c_n^{(\ell)}+D_Y y)\Delta_n^a}{n}$. The end of the proof is the same.

□

Proof of Thm. 2.3. The proof is very classical, we provide it here for completeness. Let $\varepsilon = 1/\sqrt{n}$, and let $h_1, \dots, h_{\mathcal{N}}$ be an ε-covering of $\mathcal{H}$. Using a union bound and Lemma 2.3, with probability $1 - \delta - \mathcal{N}p_n$, for all h_i we have

$$\left|\hat{\mathcal{R}}(h_i) - \mathcal{R}_n(h_i)\right| \lesssim (D_n + nL\tilde{\Delta}_n^a + C_n)p_n + (D_n + \sqrt{n}L\tilde{\Delta}_n^a)\sqrt{\frac{\log(\mathcal{N}/\delta)}{n}}$$

When the loss is Lipschitz, we have $\left|\hat{\mathcal{R}}(h) - \hat{\mathcal{R}}(h')\right| \leqslant L_\ell d_{\mathsf{Y},n}(h, h')$ and similarly for $\mathcal{R}_n$. Thus, for any $h \in \mathcal{H}$, considering h_i such that $d_{\mathsf{Y},n}(h, h_i) \leqslant \varepsilon$, we have

$$\begin{aligned}\left|\hat{\mathcal{R}}(h) - \mathcal{R}_n(h)\right| &\leqslant \left|\hat{\mathcal{R}}(h) - \hat{\mathcal{R}}(h_i)\right| + \left|\hat{\mathcal{R}}(h_i) - \mathcal{R}_n(h_i)\right| + |\mathcal{R}_n(h_i) - \mathcal{R}_n(h)| \\ &\lesssim 2L_\ell\varepsilon + (D_n + nL\tilde{\Delta}_n^a + C_n^{(\ell)})p_n + (D_n + \sqrt{n}L\tilde{\Delta}_n^a)\sqrt{\frac{\log(\mathcal{N}/\delta)}{n}}\end{aligned}$$

which completes the proof.

When the loss is the square loss, we have $\left|\hat{\mathcal{R}}(h) - \hat{\mathcal{R}}(h')\right| \lesssim (C_n^{(\ell)} + D_{\mathcal{Y}})d_{\mathsf{Y},n}(h, h')$ and similarly for $\mathcal{R}_n$, the end of the proof is the same.

□

2.B Technical Material

Theorem 2.4 (Generalized McDiarmid's inequality [33])**.** *Let $(X_1, \dots, X_n) \in \mathcal{X} = \prod_{i=1}^n \mathcal{X}_i$ be independent random variables, and $f : \mathcal{X} \to \mathbb{R}$. Assume that there is $\mathcal{Y} \in \mathcal{X}$ such that: for all $X, X' \in \mathcal{Y}$ such that $X_j = X'_j$ except for some index i,*

$$|f(X) - f(X')| \leqslant c_i$$

and denote $p = \mathbb{P}(X \in \mathcal{Y})$. Moreover, assume that $|f(X)| \leqslant C$ with probability 1. Then with probability at least $1 - \delta - p$,

$$|f(X) - \mathbb{E}f(X)| \lesssim p\left(\|c\|_1 + C\right) + \|c\|_2 \sqrt{\log(1/\delta)} \tag{2.55}$$

where $c = (c_1, \dots, c_n)$.

Chapter 3

Generalization of Graph Neural Networks on Latent Position Models

---

Contents

Summary

In this chapter, we apply the framework of the previous chapter to Graph Neural Networks and bound their generalization error. In more details:

— For S-**GNNs**, we will study the three error terms in the excess risk decomposition. We will derive generic convergence conditions on S and instantiate two examples of S and two examples of PEs. Universality will be obtained in some specific cases.

— For other examples of **Message-Passing**, including attention-based ones, we will study the transferability error, based on a stability-based concentration bound. We will leave the other error terms still open.

For S-GNNs, the transferability error was studied in [NK14, NK13], the approximation error was studied in [NK15], and the estimation error was studied during the internship of Y. Viegas and is still unpublished. For general models, the transferability error was studied during the PhD thesis of M. Cordonnier [NK1], co-supervised with N. Tremblay and S. Vaiter.

In the previous chapter, we have introduced a general framework for characterizing the generalization error of node tasks on LPMs, with an excess risk decomposition in three terms: transferability error, estimation error, and approximation error. Most of the properties that we introduced relied on the notion of $\mathfrak{P}$-convergence on LPMs, that is, the notion that node prediction models converges toward *function on the latent space*, that can then be analyzed with regular ML tools. We then left the question: to what models do these notions actually apply? Unsurprisingly, in this chapter, we study Graph Neural Networks (GNNs), that we have introduced in Chapter 1. We will in particular focus on S-GNNs, for which all conditions can be deported to convergence conditions directly on S, which has a long history [12, 94, 129, 92, 146]. The chapter is organized as follows:

— In Sec. 3.1, we give a brief reminder on GNNs from Chapter 1 and introduce our main examples of message-passing.

— In Sec. 3.2, we examine the case of S-GNNs. We will be able to bound the transferability and estimation errors by deriving directly convergence conditions on S, and derive universality for the approximation error in some specific cases.

— In Sec. 3.3, we examine several examples of *general message-passing* schemes and bound their transferability error. We obtain different rates depending on the case.

As before, all proofs (and sometimes more detailed versions of the theorems) are available in Appendix.

Notations. We will need several notations. We recall that for a vector z, we have $\|z\|_q = (\sum_i |z_i|^q)^{1/q}$, and for a graph matrix S, the norm $\|S\|_q = \max_z \|Sz\|_q / \|z\|_q$. When omitted, we have $\|z\| = \|z\|_2$ the Euclidean norm and $\|S\| = \|S\|_2$ the largest singular value of S. Note that we will also use $\|\theta\|$ the operator norm for the weight matrices.

All along the chapter, $\mathfrak{P}$ is an LPM, with a compact latent space $\mathcal{X} \subset \mathbb{R}^{d_x}$, and kernel

$$W_n(x, x') = \alpha_n w(x, x')$$

with sparsity level $\alpha_n \gtrsim \log n/n$. Using the notations of the last chapter, the output space will be $\mathsf{Y} = \mathbb{R}^{d_y}$, and its metric d_{Y} the Euclidean norm. Regarding the results of the last chapter, this allows for any Lipschitz loss or the square loss. For a node features matrix $Z \in \mathbb{R}^{n \times d}$ and $q > 1$, it will be convenient to define the norm

$$\|Z\|_{q,2,n} = \left(\frac{1}{n} \sum_{i=1}^{n} \|Z_{i,:}\|_2^q \right)^{1/q} \tag{3.1}$$

such that, e.g., $\left(\frac{1}{n} \sum_{i=1}^n d_{\mathsf{Y}}(z_i, z_i')^q \right)^{1/q} = \|Z - Z'\|_{q,2,n}$. This is convenient to express $\mathfrak{P}$-convergence of order q (Def. 2.2). When we let $q \to \infty$, we obtain $\|Z\|_{\infty,2} = \sup_i \|Z_{i,:}\|$ (which no longer depends on n).

Finally, we denote by $\mathcal{B}(\mathcal{X})$ the set of bounded functions $f : \mathcal{X} \to \mathbb{R}$, and by $\mathcal{B}(\mathcal{X})^d$ if the output is d-dimensional. For instance, $\mathcal{B}(\mathcal{X})^{d_y} \subset \mathcal{F}_{all}$ in the notations of the previous chapter. For $f \in \mathcal{B}(\mathcal{X})^d$, we define $\iota_X f = [f(x_i)]_{i=1}^n \in \mathbb{R}^{n \times d}$ the sampling operator that returns the function f evaluated at the latent variables as rows of a matrix.

3.1 Reminder on GNNs

In this section, we recall some basic definitions on Graph Neural Networks (see Chap. 1) and give important examples on which we will illustrate our theory.

3.1.1 GNNs

As we mentioned in the introduction, the overwhelming majority of GNNs follow the *message-passing* paradigm: at each layer, *messages* are transmitted along the edges of the graph, and *aggregated* to form new node representations $Z^{(k)} \in \mathbb{R}^{n \times d_k}$, as

$$Z_{i,:}^{(k)} = \mathrm{AGG}_k \left(Z_{i,:}^{(k-1)}, \left\{ \left(s_{ij}, Z_{j,:}^{(k-1)} \right) \right\}_{j \in \mathcal{N}(i)} \right)$$

where $\mathrm{AGG}^{(k)}$ is an *aggregation function* that treats $\left\{\left(s_{ij}, Z_{j,:}^{(k-1)}\right)\right\}_{j\in\mathcal{N}(i)}$ as an unordered *set*. Often, node representations at the initial layer are directly the node features $Z^{(0)} = Z$, or *Positional Encodings*, as we describe below.

S-GNN. Very often, the GNN is a so-called S-GNN, or S-MPGNN. Given a representation of the graph under the form of a matrix $S = S(A)$, the GNN is written as

$$Z^{(k)} = \rho_k\left(Z^{(k-1)}\theta_0^{(k)} + SZ^{(k-1)}\theta_1^{(k)} + 1_n {b^{(k)}}^\top\right) \in \mathbb{R}^{n\times d_k}, \quad \Phi_\theta(A,Z) = Z^{(L)} \tag{3.2}$$

where $S = S(A), Z^{(0)} = Z^{(0)}(A,Z)$ are PEs, $\theta_i^{(k)} \in \mathbb{R}^{d_k\times d_{k+1}}$ are learnable weight matrices, $b^{(k)} \in \mathbb{R}^{d_k}$ is a learnable bias vector, and $\rho_k : \mathbb{R}\to\mathbb{R}$ are activation function applied elementwise. There are many examples of such graph matrix S: normalized adjacency, normalized Laplacian, with added self loop... The only property that $S = S(A)$ must satisfy is equivariance, since it implies equivariance of the GNN. We will see two particular examples below.

3.1.2 Augmenting the node features: Positional Encodings

As we have seen in the introduction (Sec. 1.3.3.3), in the overwhelming majority of situations, one will just consider the node features Z as input to the GNN:

$$Z^{(0)} = Z \tag{3.3}$$

Nevertheless, in some situations, or one may want to *augment* them with other graph-related quantities that are generally called *Positional Encodings* (PE) [45, 17], by analogy with Transformers [1]. We will use the generic notation

$$Z^{(0)} = Z^{(0)}(A,Z) \tag{3.4}$$

where $Z^{(0)}(\cdot)$ can be any permutation-equivariant function, trainable or not. Here we leave the definition of PEs voluntarily general (they can be anything), although in practice they are usually relatively simple and not trainable: the eigenvectors of the Laplacian, etc. We will see two examples in the following section.

3.1.3 Examples

We now present several running examples: of S-GNNs, of more general message-passing schemes, and of PEs.

1. even if, as we mentioned in the introduction, they bear little similarity to PEs in Transformers.

3.1.3.1 Examples of S

We introduce two classical examples of graph matrix S.

Example S1 (Adjacency)**.** *The first example is the adjacency $S = A/(n\alpha_n)$ normalized by $n\alpha_n$, where α_n is the sparsity level of the kernel W_n. For simplicity, we assume that α_n is known.*

Example S2 (Normalized Laplacian)**.** *The second example is a degree-normalized adjacency related to the symmetric normalized Laplacian $S = D^{-1/2}AD^{-1/2} = Id - L_{sym}$ where $D = D(A) = diag(A1_n)$ is the diagonal matrix of degrees. We will sometimes refer to it as "Laplacian" although it is actually $Id - L$, since the "identity" factor can be emulated within the GNN.*

Note that Ex. S1 involves a careful normalization by the sparsity level, which is required for our convergence results later on. We assumed that it is known for simplicity, but of course it would have to be estimated in practice, which is not a well-defined endeavor when one observes only one single training graph! On the contrary, the degree-normalized adjacency does not require to know α_n (the degree is an "automatic" normalization), which is why it is often preferred in practice. Nevertheless, some of our results will only be proven for the adjacency.

Note that another classical example is the random walk matrix $S = D^{-1}A$, however we do not use it in this chapter since it is not symmetrical and therefore quite complexify the computations. It is however a particular case of Ex. MPGNN2 below, quite popular in practice, and key in some studies on oversmoothing (Chapter 4).

3.1.3.2 Examples of other message-passing scheme

Here we introduce other examples of message-passing schemes, which are not S-GNNs. These examples are from our paper [NK1]. We denote $z_i^{(k)} = Z_{i,:}^{(k)}$ for short.

Example MPGNN1 (Graph Attention Network (GAT))**.** *Graph Attention Network (GAT[2]) were introduced by [151]. Inspired by the attention mechanism in Transformers,* attention coefficients *are computed along each edge of the graph at each layer, using the node representations. Then normalized sum-aggregation is performed with this coefficients:*

$$z_i^{(k)} = \frac{\sum_{j=1}^n c^{(k)}\left(z_i^{(k-1)}, z_j^{(k-1)}, a_{ij}\right) m^{(k)}(z_j^{(k-1)})}{\sum_{j=1}^n c^{(k)}\left(z_i^{(k-1)}, z_j^{(k-1)}, a_{ij}\right)} \tag{3.5}$$

where $m^{(k)} : \mathbb{R}^{d_{k-1}} \to \mathbb{R}^{d_k}$ is some mapping (often an MLP), and $c^{(k)}(z_i^{(k)}, z_j^{(k)}, a_{ij}) \in \mathbb{R}_+$ are attention coefficients that depend on the node representations at each end of the considered edge, and on the edge weight a_{ij}. Often, these coefficients are computed only along existing edges, such that $c(\cdot, \cdot, 0) = 0$. A common choice is to choose exponential coefficients, such that the

2. The correct acronyme GAN was already very well-known in ML as Generative Adversarial Networks.

aggregation operation is a soft-max, traditionally found in Transformers.

While this is not an S-GNN, note that the aggregation step in GATs can still be written as

$$H^{(k)} = S(Z^{(k)})Z^{(k)} \tag{3.6}$$

instead of SZ with a fixed matrix S.

Example MPGNN2 (Generalized mean)**.** *Generalized means [80] are defined as bla*

$$z_i^{(k)} = g\left(\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} g^{-1}\left(a_{ij}m^{(k)}(z_j^{(k-1)})\right)\right) \tag{3.7}$$

for some invertible function g (possibly defined on a bounded domain), generally applied componentwise. In this case, following the popular formalism from [78], in order to legitimately be considered as a "generalized mean", some regularity conditions are required on g, typically, it must be a strictly increasing continuous function. For instance, taking $g(x) = e^x$ (assuming positivity of the inputs for invertibility) yields the geometric mean, while taking $g(x) = x^{1/p}$ (again, assuming nonnegativity of the message if necessary) yields a power mean as employed in the moment-based aggregations from [34] (up to centering, which we omit for simplicity).

Note that, by taking $g = Id$, we indeed find a S-GNN with the random walk matrix $S = D^{-1}A$. However, we will still treat it as a general MP rather than a S-GNN.

Example MPGNN3. *Finally, an example drastically different from the previous ones is the so-called maximum aggregation. It is mentioned in the literature [61] but less common in practice.*

$$z_i^{(k)} = \max_j \left(a_{ij}m^{(k)}(z_j^{(k-1)})\right) \tag{3.8}$$

where the max is taken componentwise along the dimension d_k.

3.1.3.3 Examples of PEs

Let us now turn to some example of PEs, from our paper [NK13].

Example PE1 (Spectral PE)**.** *It has been proposed [45, 46] to feed the first q eigenvectors of the graph into the GNN, for a fixed q. A potential problem with this approach is the sign ambiguity of the eigenvectors, or even the basis ambiguity in case of eigenvalues with multiplicities. Here we consider only the sign ambiguity for simplicity. The sign ambiguity was alleviated in [89] by taking a* sign-invariant *function: considering an eigenvector u_i^S of S,*

$$(\mathbf{Q}f)(u_i^S) \overset{\text{def}}{=} f(u_i^S) + f(-u_i^S) \in \mathbb{R}^{n\times p} \tag{3.9}$$

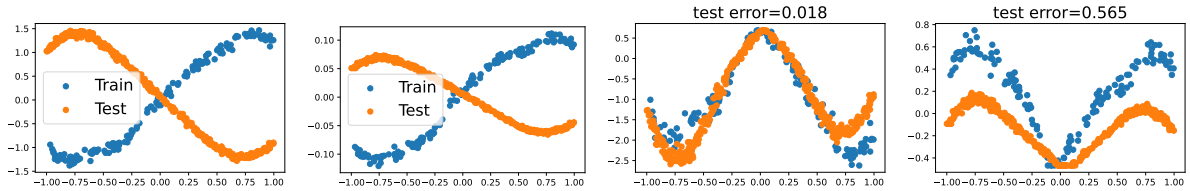


Figure 3.1 – Illustration of the role of the SignNet architecture and of the renormalization by $\sqrt{n}$ of the eigenvectors on synthetic data, with a latent space $\mathcal{X} = [-1, 1]$ (x-axis), a Gaussian kernel w, and uniform distribution P_x. Blue dots represent a graph from the training set, orange dot a test graph that is twice bigger. **From left to right:** eigenvectors with renormalization (with a different sign for the two graphs), eigenvectors without, PEs with, and PEs without, with the regression test errors of a GNN trained using these PE with or without renormalization by $\sqrt{n}$. We observe that SignNet indeed fixed the sign ambiguity. The absence of renormalization yields *unconsistent PEs* across graphs of different sizes, which results in a high test error on test graphs than training graphs. Figure from [NK13].

where $f : \mathbb{R} \to \mathbb{R}^p$ is a function applied to each coordinate of u_i^S to preserve permutation-equivariance. The resulting function is sign-invariant, and one can parameterize f e.g. as a MultiLayer Perceptron (MLP). Given the first q eigenvectors u_i^S and a collection of MLPs $f_{\gamma_i} : \mathbb{R} \to \mathbb{R}^{p_i}$ for some output dimensions p_i, the entire PE concatenates the outputs:

$$Z_\gamma^{(0)}(S) = [(\mathbf{Q} f_{\gamma_i})(\sqrt{n} u_i^S)]_{i=1}^q \in \mathbb{R}^{n \times p} \tag{3.10}$$

where $p = \sum_{i=1}^q p_i$ and the MLP are applied element-wise. The parameter γ gathers the γ_i. The equation (3.10) involves a renormalization of the eigenvectors u^S by the square root of the size of the graph $\sqrt{n}$: indeed, as u_i^S is normalized in $\mathbb{R}^n$ as $\left\| u_i^S \right\| = 1$, this is necessary for consistency across different graph sizes. See Fig. 3.1 for an illustration.

Example PE2. *In [88], the authors propose to define PEs through the aggregation of a set of random walk "distances" $\xi(i, j)$ from each node i to a set $j \in V_T$ of target nodes (typically, labelled nodes in semi-supervised learning, or anchor nodes selected randomly [166]).*

$$(Z_\gamma^{(0)})_{i,:} = AGG(\{\xi(i, j) \mid j \in V_T\})$$

For simplicity, since here we do not consider any particular set of target nodes, we just consider $V_T = V$ the set of all nodes. Moreover, we replace the random walk matrix with some graph shift matrix $\hat{S}$. As aggregation, we opt for the deep-set architecture [169], which applies an MLP on each $\xi(i, j)$ then a sum. Deep sets can approximate any permutation-invariant function. As we will see below, with the proper normalization to ensure convergence, we obtain:

$$Z_\gamma^{(0)} = \frac{1}{n} \sum_j f_\gamma \left(n \cdot [\hat{S} e_j, \dots, \hat{S}^q e_j] \right) \in \mathbb{R}^{n \times q} \tag{3.11}$$

where $f_\gamma : \mathbb{R}^q \to \mathbb{R}^p$ is an MLP applied row-wise and $e_j \in \mathbb{R}^n$ are one-hot basis vectors. As we

will see later, for our examples of S, to ensure proper convergence $\hat{S}$ can be chosen as some filtering $\hat{S} = h_{\gamma'}(S)$ *to remove the low eigenvalues of S, similar to the USVT method [28].*

3.2 Generalization error of S-GNNs

In this section, we fully bound the excess risk for S-GNNs (3.2). As we have seen in the last chapter, our analysis mostly relies on the notion of **$\mathfrak{P}$-convergence** (Def. 2.1 and 2.2), that is, the notion that functions on random LPM graphs converge towards functions on the latent space. From this, we were able to relate all the various errors in the risk decomposition (2.29) to more traditional quantities for functions on $\mathcal{X}$.

Hence, we will naturally start by showing $\mathfrak{P}$-convergence for S-GNNs. It will entirely rely upon some notion of convergence **of the operator S itself**, which is a well-known type of results for graph operators.

In the whole section, the non-linearity will satisfy the following.
Assumption 3.1. *The functions ρ_k are 1-Lipschitz, and $|\rho_k(x)| \leqslant |x|$.*

3.2.1 $\mathfrak{P}$-convergence and transferability error

The main strategy for S-GNN is that, in many cases, the matrix S will converge to an *operator* $\mathbf{S} : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$. This type of results on graph operators has been studied for a long time [94, 129]. When similarly $Z^{(0)}$ is $\mathfrak{P}$-convergent to some $f^{(0)} \in \mathcal{B}(\mathcal{X})^{d_0}$, this allows us to define the family of limit functions $\mathcal{F}_{\mathcal{H}}$, that we like to refer to as "continuous GNNs" (cGNN). Here, **S**-cGNN are built by **propagating functions over the latent space instead of graph signals**, and replacing S with $\mathbf{S}$. For a S-GNN Φ_θ, the same set of parameters θ define a **S**-cGNN as

$$f^{(k)} = \rho_k \circ \left({\theta_0^{(k)}}^\top f^{(k-1)} + {\theta_1^{(k)}}^\top \mathbf{S} f^{(k-1)} + b^{(k)} \right), \quad \Phi_\theta^{\mathrm{c}}(\mathbf{S}, f^{(0)}) = f^{(L)} \tag{3.12}$$

where $f^{(k)} \in \mathcal{B}(\mathcal{X})^{d_k}$ are multivariate functions, for which $\mathbf{S}f^{(k)} = [\mathbf{S}f_1^{(k)}, \dots, \mathbf{S}f_{d_k}^{(k)}]$ is applied to each coordinate, and similarly for the non-linearities $\rho_k \circ f^{(k)} = [\rho_k \circ f_1^{(k)}, \dots, \rho_k \circ f_{d_k}^{(k)}]$.

Let us now state our assumptions on the convergence of individual elements. In the rest of the section, we consider some $q \geqslant 1$ to express $\mathfrak{P}$-convergence of order q (Def. 2.2), which brings a measure of flexibility to the various assumptions that we need to prove in each example. While $q = 1$ is sufficient, for S-GNNs $q = 2$ will often be more convenient, as it involves the spectral norm. The convergence of the operator S will be the following.
Assumption 3.2 ($\mathfrak{P}$-convergence of S)**.** *For any $f \in \mathcal{B}(\mathcal{X})$, assume that the function $(A, X) \mapsto S\iota_X f$ is $\mathfrak{P}$-convergent of order q towards $\mathbf{S}f$ with rate $\|f\|_\infty r_n^{(S,q)}$, where $r_n^{(S,q)} \to 0$.*

Remark that $(A, X) \mapsto S\iota_X f$ is a valid equivariant function that uses directly the (unobserved) latent variables X: it is mostly an abstract object that serves to express the convergence of S, and not something that is computed in practice.

Naturally, we also need convergence of the input node features/PE.

Assumption 3.3. *The Positional Encodings $Z^{(0)}(A, Z)$ are $\mathfrak{P}$-convergent of order q towards $f^{(0)} \in \mathcal{B}(\mathcal{X})^{d_0}$ with rate $r_n^{(Z,q)} \to 0$.*

The prototypical example of convergent input node features/PE is when the node features $Z = \iota_X f^{(0)}$ are themselves a sampling of some function over the latent variables, in which case $r_n^{(Z)} = 0$.

Finally, various elements need to be bounded, which for some examples will only happen with high probability, similarly to Assumption 2.2. The assumption below is generally mild.

Assumption 3.4. *For the same set of graphs $\mathcal{G}_n \subset \mathfrak{G}_n$ as Assumption 2.2, assume that:*

$$\|S\|_q \leqslant \begin{cases} c_n^{(S,q)} & \text{if } (A,Z) \in \mathcal{G}_n \\ C_n^{(S,q)} & \text{otherwise} \end{cases}, \quad \left\|Z^{(0)}\right\|_{q,2,n} \leqslant \begin{cases} c_n^{(Z,q)} & \text{if } (A,Z) \in \mathcal{G}_n \\ C_n^{(Z,q)} & \text{otherwise} \end{cases} \tag{3.13}$$

Then we have the following result, presented here in a simplified version. A proof and detailed version are available in Appendix.

Theorem 3.1 ($\mathfrak{P}$-Convergence of S-GNN)**.** *Suppose that Assumptions 3.2, 3.3 and 3.4 hold. Denote by $p_n = P_n(\mathcal{G}_n^c)$ the probability that $(A, Z) \notin \mathcal{G}_n$. Then Φ_θ is $\mathfrak{P}$-convergent of order q towards $\Phi_\theta^c(\mathbf{S}, f^{(0)})$, with rate*

$$r_n^{(\theta)} \lesssim r_n^{(Z,q)} + r_n^{(S,q)} + p_n(1 + C_n^{(Z,q)})(1 + C_n^{(S,q)})^L \tag{3.14}$$

where the multiplicative constant depends polynomially on $\left\|\theta_i^{(k)}\right\|$, $\left\|b^{(k)}\right\|$, $\left\|f^{(0)}\right\|_\infty$ and $c_n^{(S,q)}$.

As seen in the last Chapter, this means that such S-GNNs are indeed in $\mathcal{H}_{all,\infty}$, and that under Assumption 2.1 on the loss, their limit risk is the regular ML risk applied to the cGNN:

$$\mathcal{R}_n(\Phi_\theta) \xrightarrow[n\to\infty]{} \mathcal{R}_\infty(\Phi_\theta) = \mathcal{R}_{\mathrm{ML}}(\Phi_\theta^c(\mathbf{S}, f^{(0)})) \tag{3.15}$$

Moreover, by applying Prop. 2.3, if we consider a set of S-GNNs $\mathcal{H}_\Theta = \{\Phi_\theta \mid \theta \in \Theta\}$ with bounded parameters θ, the **transferability error** is directly bounded:

$$\sup_{\theta\in\Theta} |\mathcal{R}_n(\Phi_\theta) - \mathcal{R}_\infty(\Phi_\theta)| \lesssim \sup_\theta r_n^{(\theta)} \to 0 \tag{3.16}$$

where the multiplicative constant is either the Lipschitz constant of the loss, or depends on Θ

for the square loss (see Prop. 2.3 and Lemma 3.1 in Appendix).

To summarize: when the graph operator S converge to a functional operator $\mathbf{S}$ (Assumption. 3.2), then the risk for node tasks of the discrete GNN $\mathcal{R}_n(\Phi_\theta)$ converges to the *regular ML risk* of the continuous GNNs $\mathcal{R}_{\mathrm{ML}}(\Phi^c_\theta(\mathbf{S}, f^{(0)}))$. Continuous GNNs then become the central object of interest: in particular, we will be able to prove that they are sometimes universal *as functions on the latent space* $\mathcal{X}$ in Sec. 3.2.4.

3.2.2 Estimation error

In the previous chapter, we bound the estimation error with a stability bound (Assumption 2.2) and a covering numbers approach. As before, Assumption 2.2 can be deduced from a stability property directly on S and $Z^{(0)}$. We state this in the assumption below, which is (rather pointlessly, but unavoidably) verbose.

Assumption 3.5. *In the following, we denote by $S = S(A)$, $S' = S(A')$, $Z^{(0)} = Z^{(0)}(A, Z)$ and $Z'^{(0)} = Z^{(0)}(A', Z')$. Assume that there is a set of graphs $\mathcal{G}_n$ such that, for all $(A, Z), (A', Z') \in \mathcal{G}_n$:*

— *if $Z = Z'$ and A, A' differs on only one coordinate A_{ij},*

$$\sum_{\ell,k=1}^{n} |S_{\ell k} - S'_{\ell k}| \leqslant \Delta_n^{a,S}, \quad \sum_k \left\| Z^{(0)}_{k,:} - Z'^{(0)}_{k,:} \right\| \leqslant \Delta_n^{a,Z}$$

— *if $Z = Z'$, and A, A' differs on one line $A_{i,:}$ and the corresponding column $A_{:,i}$,*

$$\sum_{\ell,k=1}^{n} |S_{\ell k} - S'_{\ell k}| \leqslant \Delta_n^{A,S}, \quad \sum_k \left\| Z^{(0)}_{k,:} - Z'^{(0)}_{k,:} \right\| \leqslant \Delta_n^{A,Z}$$

— *if $A = A'$ and Z, Z' differs on one sample $z_i \neq z'_i$,*

$$\sum_k \left\| Z^{(0)}_{k,:} - Z'^{(0)}_{k,:} \right\| \leqslant \Delta_n^{z,Z}$$

The proof of the following result, and a more detailed version, is in the Appendix.

Theorem 3.2. *Let Φ_θ be a S-GNN. Suppose that Assumption 3.5 holds, and that Assumption 3.4 holds for $q = 1$ and $q = \infty$. Suppose that the loss satisfies Assumption 2.1. Then Assumption*

2.2 and 2.3 are satisfied, with

$$\begin{aligned}\Delta_n^a &\lesssim \Delta_n^{a,S} + \Delta_n^{a,Z}\\ \Delta_n^A &\lesssim \Delta_n^{A,S} + \Delta_n^{A,Z}\\ \Delta_n^z &\lesssim \Delta_n^{z,Z}\\ c_n^{(\ell)} &= O(1)\end{aligned}$$

where the multiplicative constants depend polynomially on $\left\|\theta_0^{(k)}\right\|, \left\|\theta_1^{(k)}\right\|, \left\|b^{(k)}\right\|$, $c_n^{(S,1)}, c_n^{(S,\infty)}$, *and* $c_n^{(Z,\infty)}$. *Moreover,* $C_n^{(\ell)}$ *depends polynomially on* $\left\|\theta_0^{(k)}\right\|, \left\|\theta_1^{(k)}\right\|, \left\|b^{(k)}\right\|$, *and* $C_n^{(S,\infty)}, C_n^{(Z,\infty)}$.

Now let $\mathcal{H}_\Theta = \{\Phi_\theta \mid \theta \in \Theta\}$ be a set of S-GNNs with parameters in a bounded set Θ. To apply theorem 2.3 and bound the estimation error, we need to bound its covering numbers. For this we introduce the metric between parameters θ, θ':

$$\|\theta\|_\Theta = \max_k \max\left(\left\|\theta_0^{(k)}\right\|, \left\|\theta_1^{(k)}\right\|, \left\|b^{(k)}\right\|\right) \tag{3.17}$$

and we suppose that $\max_{\theta\in\Theta} \|\theta\|_\Theta \leqslant D_\Theta$. Then we have the following.

Proposition 3.1. *Suppose that Assumption 3.5 holds, and that Assumption 3.4 holds for* $q = 1$ *and* $q = \infty$. *Then*

$$\mathcal{N}(\mathcal{H}_\Theta, \varepsilon, d_{\mathsf{Y},n}) \lesssim \left(\frac{D}{\varepsilon}\right)^{dim(\theta)}$$

where $dim(\theta)$ *is the number of parameters in the GNN, and* D *is polynomial is in* $O(D_\Theta^{2L} {C_n^{(S,\infty)}}^{2L} C_n^{(Z,\infty)})$.

Hence, applying Theorem 2.3, assuming that all the multiplicative constants are in 0(1) and considering that p_n is negligible, we obtain that the rate for the estimation error of S-GNNs is approximately

$$\left(\Delta_n^{A,S} + \Delta_n^{A,Z} + \sqrt{n}(\Delta_n^{a,S} + \Delta_n^{a,Z})\right)\sqrt{\frac{\dim(\theta)\log n}{n}} \tag{3.18}$$

We apply it to our examples in the next section.

3.2.3 Application to examples

In this section we bound the transferability and estimation error for our examples of S-GNNs S1 and S2, and PEs PE1 and PE2. In particular, we give their limit values $\mathbf{S}$ and $f^{(0)}$.

Recall that to bound the estimation *and* tranferability error, we need to satisfy:

— Assumptions 3.2 and 3.4 for the same q. Here, we will consider $q = 2$. This will yield $\mathfrak{P}$-convergence of order q, and recall that order 2 implies order 1.

— Assumptions 3.4 or $q = 1$ and $q = \infty$ (technically, for $q = 1$ only the constant $c^{(S,1)}$ is

required), and Assumption 3.5, which bounds the estimation error.

3.2.3.1 Graph operators

Here we study our two examples of S, and give the final rate of convergence when we omit the PEs.

Proposition 3.2 (Tranferability and Estimation error for Ex. S1)**.** *Consider the LPM random graphs* (2.32) *with iid latent positions* $x_i \sim P_x$ *and Bernoulli edges* $a_{ij} \sim Ber(\alpha_n w(x_i, x_j))$*, with* $\alpha_n \gtrsim \frac{\log n}{n}$*. Take* $S = A/(\alpha_n n)$*, and define*

$$\mathbf{S}f(x) = \int w(x, x')f(y)dP_x(x')\,. \tag{3.19}$$

Then

— *Assumption 3.2 is satisfied with* $q = 2$ *and*

$$r_n^{(S,2)} = \frac{1}{\sqrt{\alpha_n n}} + \sqrt{\frac{d_x}{n}}$$

— *For any* M*, there is a constant* C_M *such that Assumption 3.4 is satisfied for* S *with* $p_n = n^{-M}$ *and* $c_n^{(S,2)} = 2$*,* $C_n^{(S,2)} = C_M/\alpha_n$*.*

— *For the same* p_n *as above, Assumption 3.4 is satisfied with* $c^{(S,1)} = c^{(S,\infty)} = O(1)$*, and* $C^{(S,\infty)} = 1/\alpha_n$*.*

— *for the same* p_n *as above, Assumption 3.5 is satisfied with* $\Delta_n^{a,S} = 1/(n\alpha_n)$ *and* $\Delta_n^{(A,S)} = O(1)$*.*

As a consequence, in the risk decomposition (2.29)*, omitting the rate of eventual PEs, the transferability error is in*

$$O\left(\frac{1}{\sqrt{\alpha_n n}} + \sqrt{\frac{d_x}{n}}\right)\,,$$

and the estimation error is in

$$O\left(\left(\frac{1}{\alpha_n n} + \frac{1}{\sqrt{n}}\right)\sqrt{dim(\theta)\log(n)}\right)\,.$$

The proof relies on concentration for matrices with Bernoulli entries [83], to get rid of the random edges, then convergence towards continuous operators, which can be shown e.g. by chaining [NK14].

Proposition 3.3 (Transferability and Estimation error for Ex. S2)**.** *Consider the LPM random graphs* (2.32) *with iid latent positions* $x_i \sim P_x$ *and Bernoulli edges* $a_{ij} \sim Ber(\alpha_n w(x_i, x_j))$*. Assume that* $d_w(x) := \int w(x, y)dP_x(y) \geqslant c_{\min} > 0$*, and* $\alpha_n \gtrsim \frac{\log n}{c_{\min}^2 n}$*.*

Take $S = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$*, and define*

$$\mathbf{S}f(x) = \int_{\mathcal{X}} \frac{w(x,x')}{\sqrt{d_w(x)d_w(x')}} f(x')dP_x(x')\,. \tag{3.20}$$

Then

— *Assumption 3.2 is satisfied with* $q = 2$ *and*

$$r_n^{(S,2)} = c_{\min}^{-2}\left(\frac{1}{\sqrt{\alpha_n n}} + \sqrt{\frac{d_x}{n}}\right)$$

— *Assumption 3.4 is satisfied for* S *and* $q = 2$ *with* $c_n^{(S,2)} = 1$, $p_n = 0$.

— *for any* M*, there is a constant* C_M *such that Assumption 3.4 is satisfied for* S *with* $p_n = n^{-M}$ *and* $c^{(S,1)} = c^{(S,\infty)} = C_M/c_{\min}$.

— *for the same* p_n *as above, Assumption 3.5 is satisfied with* $\Delta_n^{a,S} = O(1/(c_{\min} n\alpha_n))$ *and* $\Delta_n^{(A,S)} = O(1/c_{\min}^2)$.

As a consequence, the transferability error and estimation error are of the same order as Prop. 3.2 for the adjacency matrix, up to some multiplicative factor depending on $c_{\min}$.

The proof is similar, using a concentration bound for Laplacian that we developed in [NK12].

3.2.3.2 Positional Encodings

The results in this section are from [NK13], in collaboration with S. Vaiter.

Here we show $\mathfrak{P}$-convergence for our examples of PEs (Assumptions 3.3), whose proof can be found in [NK13]. For simplicity, we do not compute the probabilitic bounds (Assumption 3.4): in preliminary computations, I found them to be extremely verbose and filled with proof artifacts and suboptimal rates that would need to be fixed (e.g. using convergence in supremum norm for eigenvectors, etc.). I focus on the rate of convergence $r_n^{(Z)}$.

For simplicity, we place ourselves in the framework of our paper [NK13] and assume that the LPM is either a SBM (finite $\mathcal{X}$) or that w is a p.s.d. kernel [3], although the proof could probably be generalized.

Assumption 3.6. *Assume that* $\mathcal{X}$ *is finite, or that* $w(x,x')$ *is a p.s.d. kernel.*

Proposition 3.4 (Tranferability and Estimation error for Ex. PE1)**.** *Consider the LPM random graphs* (2.32) *with iid latent positions* $x_i \sim P_x$ *and Bernoulli edges* $a_{ij} \sim Ber(\alpha_n w(x_i,x_j))$*. Suppose that Assumption 3.6 holds, and that* S *is either Ex. S1 or S2.*

3. such that we may call convergence results from the RKHS theory as in [129].

Consider the spectral PE Ex. PE1:

$$Z_\gamma^{(0)}(S) = [(\mathbf{Q} f_{\gamma_i})(\sqrt{n} u_i^S)]_{i=1}^q$$

and define $u_i^{\mathbf{S}}$ *the eigenfunctions corresponding to the operator* $\mathbf{S}$ *(defined in* (3.19) *or* (3.20)*), and*

$$f_\gamma^{(0)} = [(\mathbf{Q} f_{\gamma_i})(u_i^{\mathbf{S}})]_{i=1}^q \tag{3.21}$$

Then, $Z_\gamma^{(0)}$ *is* $\mathfrak{P}$*-convergent towards* $f_\gamma^{(0)}$ *with rate*

$$r_n^{(Z,2)} = r_n^{(S,2)}$$

The proof, which we do not detail here and can be found in part in [NK13], essentially relies on Davis-Kahan theorem [167] to relate the eigenvectors to our convergence results in operator norm, and convergence of the eigenvectors of S to those of $\mathbf{S}$ for our simplified kernels [129].

Proposition 3.5 (Tranferability and Estimation error for Ex. PE2)**.** *Consider the LPM random graphs* (2.32) *with iid latent positions* $x_i \sim P_x$ *and Bernoulli edges* $a_{ij} \sim Ber(\alpha_n w(x_i, x_j))$*. Suppose that Assumption 3.6 holds, and that* S *is either Ex. S1 or S2.*

Consider the distance PE Ex. PE2:

$$Z_\gamma^{(0)} = \frac{1}{n} \sum_j f_\gamma \left(n \cdot [\hat{S} e_j, \dots, \hat{S}^q e_j] \right)$$

and define

$$f_\gamma^{(0)} = \int f_\gamma \left([\mathbf{S}\delta_x(\cdot), \dots, \mathbf{S}^q \delta_x(\cdot)] \right) dP_x(x) \tag{3.22}$$

where $\mathbf{S}\delta_x(\cdot) = w(\cdot, x)$ *in the adjacency case or* $\mathbf{S}\delta_x(\cdot) = w(\cdot, x)/\sqrt{d_w(\cdot) d_w(x)}$ *in the Laplacian case.*

Then, choosing $\hat{S} = 1_{\lambda \geqslant \gamma_n}(S)$ *a filtering of* S *that removes the low eigenvalues below a threshold* γ_n*,* $Z_\gamma^{(0)}$ *is* $\mathfrak{P}$*-convergent towards* $f_\gamma^{(0)}$ *with rate*

$$r_n^{(Z,2)} = O\left(\frac{1}{(n\alpha_n)^{1/4}} \right)$$

The proof relies on the choice of $\hat{S}$ as the so-called Universal Singular Value Thresholding (USVT) [28], which allows to obtain concentration of S in Frobenius norm rather than operator norm, to the price of a slower rate.

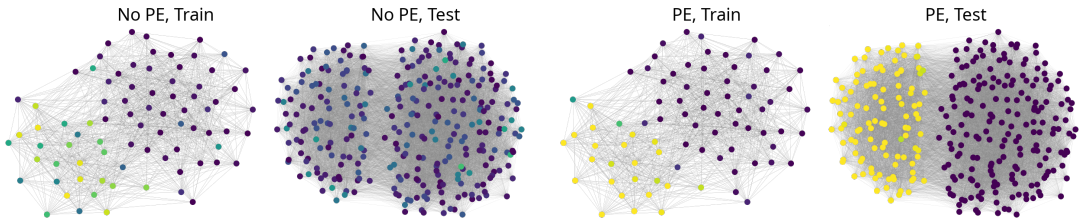


Figure 3.2 – Illustration of universality of GNNs with PEs on SBM. On an SBM with constant degree function, a GNN *without PE* might overfit the training set, but converges to a constant function, and fail at test. On the contrary, with PEs, GNNs are universal, and training succed (bottom). Figure from [NK15].

> *I use the USVT in several papers [NK8, NK5], where I show additional concentration properties depending on the choice of the threshold.*

3.2.4 Approximation error and universality

The results in this section are from [NK15, NK13], in collaboration with A. Bietti and S. Vaiter.

The only missing piece in the risk decomposition (2.29) is now the approximation error. As detailed in the previous chapter, through $\mathfrak{P}$-convergence, in can be linked to the approximation power and eventual universality *of the limit functions* on the latent space, arguably a far more convenient/familiar notion than discrete GNN "expressivity".

We report here the results from [NK15, NK13] without proof. For all the following results, we consider S as the adjacency matrix (ex. S1) and distance-encoding PE (ex. PE2), and LPM with random edges $W_n = \alpha_n w$ with SBM or p.s.d. kernel w (Assumption 3.6), with other additional hypotheses in each cases. Hence, universality means that, when we let $\theta_i^{(k)}, b^{(k)}$ grow in norm and dimensionality (as measured by some $\Omega(\theta) \leqslant R$ for increasing R), **S**-cGNNs can approximate any function in some set $\mathcal{C}$: for all $f \in \mathcal{C}$ and $\varepsilon > 0$, there is R^ε and θ with $\Omega(\theta) \leqslant R^\varepsilon$ such that

$$\left\| \Phi_\theta^{\mathrm{c}}(\mathbf{S}, f^{(0)}) - f \right\|_\infty \leqslant \varepsilon$$

where here $\mathbf{S}$ is defined as (3.19) and $f^{(0)}$ as (3.22). We have universality in the following cases.

SBM. If $\mathcal{X} = \{1, \dots, K\}$ is finite, the probability matrix $C = [w(k,\ell)]_{k,\ell=1}^K$ is invertible, and the (finite) distribution P_x is such that for all $s \in \{-1, 0, 1\}^K$, $s^\top P_x = 0$ implies $s = 0$, then GNNs are universal for all functions. This is illustrated in Fig. 3.2. It means that, asymptotically, GNNs are able to identify communities in SBMs with $\alpha_n \sim \log n/n$, which is the state-of-the-art for spectral clustering. However, GNNs must be trained! The sample complexity depends on the

approximation rate, which is unknown. In all probability, unsupervised approaches remain far superior to GNNs for clustering on SBMs, and the interest here is mostly theoretical.

Additive kernel. If

$$w(x, x') = v(u(x) + u(x'))$$

with u, v continuous and *injective* functions on their respective domains, then GNNs are universal for continuous functions. The proof essentially relies on the ability of (universal) MLPs to invert u, v and "recover" the latent variables, from which universality can be obtained.

Note that any kernel might be *approximated* by such additive kernel [169], but here we consider *exact* kernels of this form, which are not common.

Unidimensional radial kernel : if $\mathcal{X} = [-1, 1]$ and

$$w(x, x') = v(|x - x'|)$$

with continuous injective v, and P_x is symmetric (that is, $P_x([a, b]) = P_x([-b, -a])$ for all intervals), then GNN are universal among continuous **and symmetric** functions. If P is not symmetric, then GNNs are universal for continuous functions. This is illustrated in Fig. 3.3.

Radial kernels are very common, in fact most kernels in the random graph literature depend on some notion of distance between the latent variables: this is the basic intuition to create homophilic graphs. Our current proof technique is however limited to uni-dimensional latent variables.

Spherical kernels : If $\mathcal{X} = \mathbb{S}^{d-1}$ is the d-dimensional sphere and

$$w(x, x') = v(x^\top x')$$

with continuous injective v, and P_x has a density p w.r.t. the uniform distribution on the sphere such that: the unique decomposition $p(x) = \sum_{k \geqslant 0} \sum_{j=1}^{N(d,k)} a_{k,j} Y_{k,j}(x)$ where $Y_{k,j}$ are spherical harmonics is such that $x \mapsto [\sum_{j=1}^{N(d,k)} a_{k,j} Y_{k,j}(x)]_k$ is injective (see [NK15] and references therein for details on harmonic decomposition), then GNNs are universal.

Dot product kernels are in fact radial kernels on the sphere, and are therefore very common in the literature [113, 4, 144]. In particular, statistical estimation using dot-product kernels can be done using the powerful machinery of spherical harmonics [3].

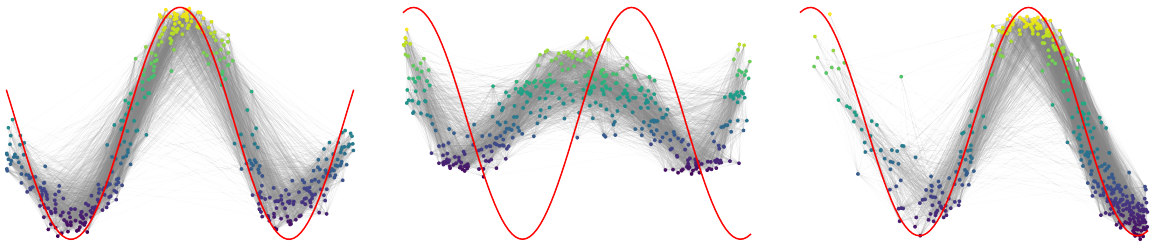


Figure 3.3 – Illustration of universality for unidimension radial kernel. The x-axis is the (unknown) latent variables in $\mathcal{X} = [-1, 1]$. The y-axis is the output of a GNN trained to approximate some function f (red curve). On the left, both distribution P_x and f are symmetric, GNNs are universal in that case. In the center, P_x is symmetric but not f, and the training expectedly fails since the limit GNN is symmetric. On the right, P_x and f are non-symmetric, and universality holds again. Figure from [NK15].

3.3 General message-passing

The material in this section was developed during M. Cordonnier's PhD thesis. It is contained in the paper [NK1], in collaboration with N. Tremblay and S. Vaiter.

Let us now turn to our other example of message-passing, that are *not* S-GNNs: attention-based (Ex. MPGNN1), generalized mean (Ex. MPGNN2) and maximum aggregation (Ex. MPGNN3). Here, I am going to report the results in [NK1], which proves $\mathfrak{P}$-convergence with a bounded difference approach. Crucially, the theory that we develop here is valid only for *non-random weighted edges* $a_{ij} = w(x_i, x_j)$, and we assume $w(\cdot) \geqslant w_{\min}$ for some $w_{\min}$ that will vary depending on the example. In most cases, handling the Bernoulli edges is still open. Roughly speaking, in the previous case of S-GNNs the S operator has a "smoothing" effect that averaged out the Bernoulli noise, which can no longer be supposed to hold for general MP. For simplicity, we also avoid dealing with the convergence of Positional Encodings: input node features are $Z^{(0)} = \iota_X f^{(0)}$.

> *For the sake of brevity, I am not going to show stability and bound the estimation error, although this could probably be done without too much difficulty under the right hypotheses. Similarly, convergence of PEs could without doubt be incorporated, but I skip it here.*

In terms of notation, we will shorten general MP as:

$$z_i^{(k)} = F^{(k-1)}(z_i^{(k-1)}, \{(z_j^{(k-1)}, a_{ij})\}_{i=1}^n), \quad \Phi(A, Z) = Z^{(L)} \tag{3.23}$$

where $F^{(k)}$ are aggregation functions that treat the set of neighbor representations $\{(z_j^{(k-1)}, a_{ij})\}_{i=1}^n$ as a (multi)set, i.e., in a permutation-invariant manner. Our general strategy is to assume that individual aggregation fucntion converge at each layer, and express convergence with stability hypotheses and an approach based on McDiarmid's inequality.

Definition of the limit. Unlike the previous S-GNN, for general MP the very definition of the "limit" cGNN is more convoluted than simply replacing the discrete operator S by its continuous counterpart. Let us start by noticing that, *if* the node representations at layer $k-1$ *were* a sampling of a function $Z^{(k-1)} = \iota_X f$, then the layer k would be of the form:

$$z_i^{(k)} = F^{(k-1)}(f(x_i), \{(f(x_j), w(x_i, x_j))\}_{j \neq i})$$

since $a_{ij} = w(x_i, x_j)$. Hence, when, $n \to \infty$, this motivates us to consider limit functions of the form

$$f^{(k)}(x) := \mathbf{F}^{(k-1)}\left(f^{(k-1)}(x), \left(f^{(k-1)}, w(x, \cdot)\right)\right), \quad \Phi^{\mathrm{c}} = f^{(L)} \tag{3.24}$$

where $\left(f^{(k)}, w(x, \cdot)\right)$ is a shortcut notation for the map $x' \mapsto \left(f^{(k)}(x'), w(x, x')\right)$.

By stacking such layers, we can recursively build a "continuous GNN" which propagates functions over the latent space, akin to replacing S with an operator in the S-GNN case.

We will proceed in two steps: *define* the limit by assuming convergence of the expectation of F, then prove a large deviation bound with McDiarmid inequality, by assuming bounded difference with respect to each x_i. The next definition is a simplified version of Def. 5 from [NK1].

Definition 3.1 ($\mathfrak{P}$-convergence of MP layers)**.** *Let F be an GNN layer with input features of dimension d_k. We say that F is $\mathfrak{P}$-convergent towards $\mathbf{F}$ with rate $r_n(\cdot)$ if, for all $f \in \mathcal{B}(x)^{d_k}$,*

$$\sup_x \|\mathbb{E}_{x_i} F\left(f(x), \{f(x_i), w(x, x_i)\}_{i=2}^n\right) - \mathbf{F}(f(x), (f, w(x, \cdot)))\| \leqslant r_n(f) \to 0 \tag{3.25}$$

Note that here, this is not the "vanilla" definition of $\mathfrak{P}$-convergence from the last chapter, but a specialized definition. The supremum over x, in particular, indicates that we are reasoning with the stronger (strongest) "uniform" convergence, rather that any order-q convergence in expectation as before. Again, it may be possible to obtain different versions of these results, but we stick with those from [NK1] in this manuscript.

Convergence of GNN Consider a sequence of layers $F^{(1)}, \ldots, F^{(L)}$, and assume that they are $\mathfrak{P}$-convergent towards $\mathbf{F}^{(k)}$ with rate $r_n^{(k)}(\cdot)$. Denote by $f^{(k)}$ the intermediate functions of the continuous GNN (3.24). We are now ready to state our convergence theorem: under a bounded difference assumption and mild stability bounds, we will have $\mathfrak{P}$-convergence of the GNN.

Our notion of bounded difference is a bit verbose, in the sense that we authorize some transform of the function before computing the stability bounds. This is a very generic notion, but in practice we introduce this especially to deal with generalized mean (Ex. MPGNN2).

Assumption 3.7 (Generalized bounded difference)**.** *There is a function ψ, assumed invertible (on the proper domain) such that: for all k,*

1. *For all $x_i = x'_i$ except for $i = 2$, we have the bounded difference*

$$\sup_x \Big\| \psi^{-1}\left(F^{(k)}\left(f^{(k-1)}(x), \{f^{(k-1)}(x_i), w(x,x_i)\}_{i\geqslant 2}\right)\right) \tag{3.26}$$

$$- \psi^{-1}\left(F^{(k)}\left(f^{(k-1)}(x), \{f^{(k-1)}(x'_i), w(x,x'_i)\}_{i\geqslant 2}\right)\right) \Big\| \leqslant \Delta_n^F \tag{3.27}$$

2. *We also have bounded difference in expectation:*

$$\sup_x \Big\| \psi\left(\mathbb{E}_{x_i}\psi^{-1}\left(F^{(k)}\left(f^{(k-1)}(x), \{f^{(k-1)}(x_i), w(x,x_i)\}_{i\geqslant 2}\right)\right)\right) \tag{3.28}$$

$$- \mathbb{E}_{x_i}\left(F^{(k)}\left(f^{(k-1)}(x), \{f^{(k-1)}(x'_i), w(x,x'_i)\}_{i\geqslant 2}\right)\right) \Big\| \leqslant \tilde{\Delta}_n^F \tag{3.29}$$

3. *The function ψ is α_ψ-Hölder on its domain with $0 < \alpha_\psi \leqslant 1$:*

$$\|\psi(y) - \psi(y')\| \leqslant K_\psi \|y - y'\|^{\alpha_\psi} \tag{3.30}$$

4. *Finally, we also need a global Lipschitz bound: there exists an exponent $0 < \alpha_F \leqslant 1$ such that, for all z_i, z'_j, and $w_{\min} \leqslant w_i \leqslant 1$:*

$$\left\|F^{(k)}(z_i, \{z_j, w_j\}_{j=2}^n) - F^{(k)}(z'_i, \{z'_j, w_j\}_{j=2}^n)\right\| \lesssim \max_i \|z_i - z'_i\|^{\alpha_F} \tag{3.31}$$

Then, our theorem is the following.

Theorem 3.3 ([Thm. 15 NK1], simplified)**.** *Consider a sequence of layers $F^{(1)}, \dots, F^{(L)}$, and assume that they are $\mathfrak{P}$-convergent towards $\mathbf{F}^{(k)}$ with rate $r_n^{(k)}$. Denote $r_n = \max_k r^{(k)}(f^{(k-1)})$. Under Assumptions 3.7, the GNN Φ is $\mathfrak{P}$-convergent of order ∞ towards Φ^{c} with rate*

$$r_n^{(\Phi)} \lesssim \left[\left((\Delta_n^F)^2 n \log n\right)^{\alpha_\psi/2} + \tilde{\Delta}_n^F + r_n\right]^{\alpha_F^{L-1}} \tag{3.32}$$

where the multiplicative constant depends polynomially on all the quantities in the problem.

The version in [Thm. 15 NK1] is a high-probability result rather than convergence in expectation, with a dependency on the probability of failure δ that is in $\log(1/\delta)$, so it is easy to obtain a result in expectation.

As before, an immediate corollary is that a set of GNNs $\mathcal{H}$ with bounded parameters converges to the set of the corresponding cGNNs $\mathcal{F}_{\mathcal{H}}$ with uniform rate $\sup_\Phi r_n^{(\Phi)}$. Unlike the previous examples of S-GNN, where the rate in $\frac{1}{\sqrt{n\alpha_n}}$ depended on the sparsity, with the classical $1/\sqrt{n}$ for dense graphs, here the convergence rate depends on multiple factors, that must be studied on a case-by-case basis: *a minima*, we need $\tilde{\Delta}_n^F \to 0$ and $\Delta_n^F = o(\sqrt{n \log n})$ to obtain a rate that goes to 0. As we will see, this is the case for all of our examples except for max aggregation,

which requires a special proof in the next section.

Examples Let us now look at our first two examples. The rates are summarized in Tab. 3.1.

Proposition 3.6 (Convergence for Example MPGNN1)**.** *Consider the GAT example. Define the continuous counterpart of* (3.5)*:*

$$f^{(k)}(x) = \int \frac{c^{(k)}(f^{(k-1)}(x), f^{(k-1)}(y), w(x,y))}{\int c^{(k)}(f^{(k-1)}(x), f^{(k-1)}(z), w(x,z))dP_x(z)} m^{(k)}(y)dP_x(y) \tag{3.33}$$

Assume that the message functions $m^{(k)}$ are Lipschitz and bounded. Assume that the attention coefficients are lower bounded $c(x,y,t) \geqslant c_{\min} > 0$, and are Lipschitz $|c(x,y,t) - c(x',y',t)| \lesssim \|x-x'\| + \|y-y'\|$. Then the hypotheses of Theorem 3.3 are satisfied with $\Delta_n^F = O(1/n)$, $\tilde{\Delta}_n^F = 0$, $r_n = O(1/\sqrt{n})$, and $\alpha_\psi = \alpha_F = 1$. The final rate is therefore $O(\sqrt{\log n/n})$.

Proposition 3.7 (Convergence for Example MPGNN2)**.** *Consider the generalized mean example. Define the continuous counterpart of* (3.7)*:*

$$f^{(k)}(x) = g\left(\int g^{-1}\left(w(x,y)m^{(k)}(f^{(k-1)}(y))\right)dP_x(y)\right) \tag{3.34}$$

Assume that the message functions $m^{(k)}$ are Lipschitz and bounded. Assume that g is α_g-Hölder and bounded, and that Assumption 3.7 is satisfied for some α_F (see examples below). Then the hypothese of Theorem 3.3 are satisfied with $\Delta_n^F = O(1/n)$, $\tilde{\Delta}_n^F = O(1/n^{\alpha_g/2})$, $r_n = O(1/n^{\alpha_g/2})$, and $\alpha_\psi = \alpha_g$. The final rate is therefore $O((\log n/n)^{\alpha_g \alpha_F^L/2})$.

For α_g, α_F, we give several examples:

1. *If $g(x) = x^{1/p}$ (moment-based aggregation), then $\alpha_g = 1/p$ and $\alpha_F = 1$*
2. *If g^{-1} is Lipschitz (and g is α_g-Hölder), then $\alpha_F = \alpha_g$. Note that for geometric mean on a bounded domain, g is also Lipschitz, such that $\alpha_g = 1$.*

Max aggregation The only example not covered until now is the maximum aggregation (Ex. MPGNN3). Unlike the previous examples, it does not satisfies the bounded difference (Assumption 3.7) with decreasing rate Δ^F. Nevertheless, we can still prove convergence, with a dedicated proof. We give a brief overview here, see [NK1] for details.

For maximum aggregation, convergence basically amounts to approximating a maximum by sampling the space. First of all, this will unsurprisingly give us a slow rate of convergence that depends on the dimension of $\mathcal{X}$, as this amounts to covering the space with balls. Secondly, this will work if the space $\mathcal{X}$ and the distribution P_x are sufficiently "regular", in the following sense.

Definition 3.2. *We say that the probability space $(\mathcal{X}, P_x)$ has the (r_0, κ)-volume retaining prop-*

Example		Random Edges	PE $Z^{(0)}$	$\mathfrak{P}$-convergence	Estim. error
S1.	Adj.	Yes	Yes	$1/\sqrt{\alpha_n n} + r_n^{(Z)}$	$\frac{\sqrt{\log n}}{\min(n\alpha_n, \sqrt{n})}$
S2.	Lapl.	Yes	Yes	$1/\sqrt{\alpha_n n} + r_n^{(Z)}$	$\frac{\sqrt{\log n}}{\min(n\alpha_n, \sqrt{n})}$
MPGNN1.	GAT	No	No	$\sqrt{\log n / n}$	
MPGNN2.	Gen. mean	No	No	$(\log n/n)^{\alpha_g \alpha_F^L/2}$	
MPGNN3.	Max.	No	No	$\left(\frac{\log n}{n}\right)^{1/d_x}$	

Table 3.1 – Summary of $\mathfrak{P}$-convergence rates for all examples. The S-GNNs admit random edges, PEs, and estimation error, unlike the generalized MP for which we leave it open.

erty if for any $r \leqslant r_0$ and any $x \in \mathcal{X}$,

$$P_x(B(x,r) \cap \mathcal{X}) \geqslant \kappa \lambda_d(B(x,r)) \tag{3.35}$$

where $B(x,r)$ is the ball centered in x of radius r (in supremum norm) and λ_d is the Lebesgue measure in $\mathbb{R}^d$.

Under this assumption we have the following theorem.

Proposition 3.8 (Convergence for Example MPGNN3)**.** *Consider the maximum aggregation example. Define the continuous counterpart of* (3.8)*:*

$$f^{(k)}(x) = \sup_y \left(w(x,y) m^{(k)}(f^{(k-1)}(y)) \right) \tag{3.36}$$

Assume that the message functions $m^{(k)}$ are Lipschitz and bounded, and that the space $(\mathcal{X}, P_x)$ has the (r_0, κ)-volume retaining property. Then the GNN Φ is $\mathfrak{P}$-convergent towards Φ^{c} with rate

$$r_n \lesssim \left(\frac{\log n}{\kappa n} \right)^{1/d_x} \tag{3.37}$$

Hence we obtain, as expected, the "slowest" rate n^{-1/d_x} that one usually gets when filling a d_x-dimensional space. Note however that, when they can be computed, many approximation rates R^ε are also exponential [6].

The rates of $\mathfrak{P}$-convergence for all of our examples are summarized in Tab. 3.1. A small comparison of the rates of S-GNN S2 and max aggregation MPGNN3 is illustrated in Fig. 3.4.

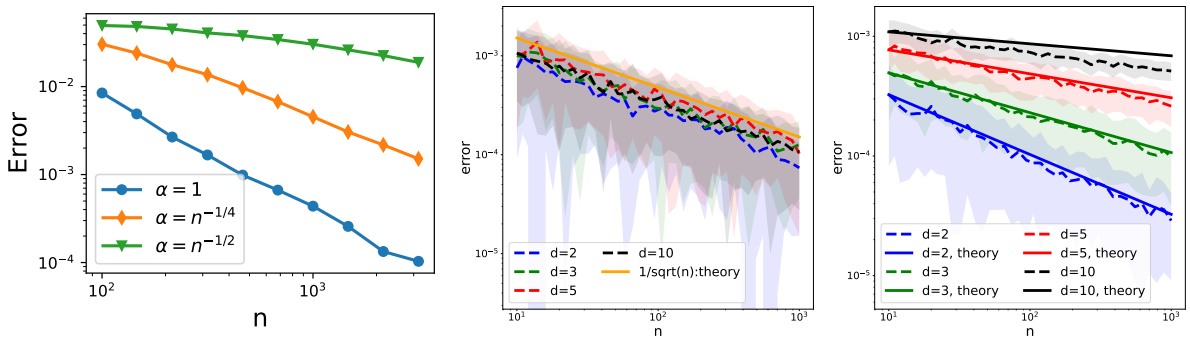


Figure 3.4 – Illustration of $\mathfrak{P}$-convergence for S-GNN Ex. S2 on random edges (left) or deterministic edges (center), and max aggregation Ex. MPGNN3 (right), with respect to α_n (left) or to the dimension of the latent space d (center and right). As described by the theory, the convergence of the S-GNN do not depend on d, while maximum aggregation results in the slow rate $O(n^{-1/d})$. Similarly, the convergence is severely slowed when α_n is small. From [NK14, NK1].

> *For brevity's sake, for this manuscript I elected to report only some numerical experiments already present in my papers, although more experiments could (and should) certainly be implemented to illustrate our generalization framework. All details of these numerics can be found in the corresponding papers.*

3.4 Conclusion and outlooks

In this (quite long, quite verbose, sometimes unnecessarily technical) chapter, I implemented the results of the previous chapter for GNNs. I covered almost entirely the case of S-GNNs with PEs, while for more general cases I limited myself to $\mathfrak{P}$-convergence, the most interesting notion in our framework.

Unsurprisingly, this chapter was quite dense, with many unavoidable notations, not all of them being very interesting (probabilistic bounds...). For brevity's sake, I volontarily ignored some results that could have been shown with a bit more work (probabilistic bounds for PEs, stability bounds for general MP...).

However, unlike the chapter itself, the (few) takeaway messages are rather straightforward:

— Just as GNNs iteratively propagate graph signals with various convolution or aggregation operations, they are $\mathfrak{P}$-convergent towards architectures that *iteratively propagate functions on the latent space*, with continuous convolution or aggregation operations.

That is, we do obtain *deep* "continuous" models (operating on functions). Just as the ML machinery was deployed in the previous chapter for expressing the generalization error, the *deep learning* literature could be leveraged to further analyze these models. For instance,

they bear similarities with other "limit" continuous deep models used in theoretical studies, e.g. on images [97], and inspired by this we study the stability of cGNNs to *deformations* of the LPM in [NK14].

— For S-GNNs, all results can be deduced from the convergence of S to a continuous graph operator $\mathbf{S}$, a known type of results with a long history.

— For generalized MP, the proof is more of a case-by-case basis, with limit models that are less systematic and must be redefined each time.

— Except in a few cases, approximation error and universality are still very much open.

There are of course many open questions left, from small results that I volontarily left aside (probabilistic bounds for PE, stability of general MP) to more profound questions.

Open questions

— In the S-GNN case, it seems possible to obtain a "*uniform*" type of convergence, that would allow to report the analysis of the estimation error *in the continuous domain*. Then, instead of suboptimal covering numbers, one may use Rademacher approaches directly on the continuous limit GNNs, which, unlike their discrete counterpart, can be easily done with classical tools. Such uniform convergence result may be obtained leveraging convergence of S towards $\mathbf{S}$ *in operator norm* [129].

— As of now, the approximation error is still pretty much open. Several interesting questions remain:

  — approximation *rates* are still open but may be obtained, since they are (partially) known for MLPs [123], and our results largely rely on the universality of MLPs within GNNs.

  — The most common case of LPMs, that of radial kernel in multiple dimensions, is still open.

  — It would be interesting to obtain other universality results with *symmetries* beyond unidimensional radial kernel, since symmetries are at the core of geometric deep learning [21].

  — Finally, we may hope to show that general MPGNNs are more powerful than S-GNNs, since this "hierarchy" of expressivity is at the core of the current GNN literature. My best guess would be that attention-based models such as GAT, which perform $S(Z)Z$ instead of SZ, is a good candidate for this: it should be possible to characterize situations where GAT are strictly more powerful than S-GNNs, in terms of approximation error and universality.

3.A Appendix

3.A.1 Proofs

Theorem 3.4 (Theorem 3.1, detailed)**.** *Suppose that Assumptions 3.2, 3.3 and 3.4 hold. Denote by $p_n = P_n(\mathcal{G}_n^c)$ the probability that $(A, Z) \notin \mathcal{G}_n$. Then Φ_θ is $\mathfrak{P}$-convergent of order q towards $\Phi_\theta(\mathbf{S}, f^{(0)})$, with rate*

$$r_n^{(\theta)} \lesssim \sum_{\ell=0}^{L-1} \left[\prod_{p=\ell+1}^{L-1} A_p \right] B_\ell + r_n^{(Z,q)} \prod_{\ell=0}^{L-1} A_\ell$$

where

$$\begin{aligned}
A_k &= \left\|\theta_0^{(k)}\right\| + \left\|\theta_1^{(k)}\right\| c_n^{(S,q)} \\
B_k &= \left\|\theta_1^{(k)}\right\| \left(p_n C_n^{(S,q)} (D_k + D_k') + r_n^{(S,q)} D_k' \right) \\
D_k &= \sum_{\ell=1}^{k} \left(\prod_{\ell=l}^{k-1} u_p \right) \left\|b^{(\ell)}\right\| + \left(\prod_{p=0}^{k-1} u_p \right) C_n^{(Z,q)} \\
D_k' &= \sum_{\ell=1}^{k} \left(\prod_{\ell=l}^{k-1} u_p' \right) \left\|b^{(\ell)}\right\| + \left(\prod_{p=0}^{k-1} u_p' \right) \left\|f^{(0)}\right\|_\infty \\
u_k &= \left\|\theta_0^{(k)}\right\| + C_n^{(S,q)} \left\|\theta_1^{(k)}\right\|, \quad u_k' = \left\|\theta_0^{(k)}\right\| + \|\mathbf{S}\|_\infty \left\|\theta_1^{(k)}\right\|
\end{aligned}$$

where $\|\mathbf{S}\|_\infty = \sup_f \|\mathbf{S}f\|_\infty / \|f\|_\infty$.

Proof. We bound recursively $E_k = \mathbb{E}_{P_n} \left\| Z^{(k)} - \iota_X f^{(k)} \right\|_{q,2,n}$. Using the Lipschitz property of ρ_k and Lemma 3.2,

$$\begin{aligned}
E_k &= \mathbb{E}_{P_n} \left\| Z^{(k)} - \iota_X f^{(k)} \right\|_{q,2,n} \\
&= \mathbb{E}_{P_n} \Big\| \rho_k \left(Z^{(k-1)} \theta_0^{(k-1)} + S Z^{(k-1)} \theta_1^{(k-1)} + 1_n b^{(k)\top} \right) \\
&\qquad - \rho_k \left((\iota_X f^{(k-1)}) \theta_0^{(k-1)} + (\iota_X \mathbf{S} f^{(k-1)}) \theta_1^{(k-1)} + 1_n b^{(k)\top} \right) \Big\|_{q,2,n} \\
&\leqslant \left\|\theta_0^{(k-1)}\right\| E_{k-1} + \left\|\theta_1^{(k-1)}\right\| \mathbb{E}_{P_n} \left\| S Z^{(k-1)} - (\iota_X \mathbf{S} f^{(k-1)}) \right\|_{q,2,n}
\end{aligned}$$

From there, again with Lemma 3.2

$$\begin{aligned}
\mathbb{E}_{P_n} \left\| S Z^{(k-1)} - (\iota_X \mathbf{S} f^{(k-1)}) \right\|_{q,2,n} \leqslant{}& \mathbb{E}_{P_n} \|S\|_q \left\| Z^{(k-1)} - \iota_X f^{(k-1)} \right\|_{q,2,n} \\
&+ \underbrace{\mathbb{E}_{P_n} \left\| S \iota_X f^{(k-1)} - (\iota_X \mathbf{S} f^{(k-1)}) \right\|_{q,2,n}}_{\leqslant r_n^{(S,q)} \left\| f^{(k-1)} \right\|_\infty}
\end{aligned}$$

Hence, the first term is bounded by decomposing under the event $\mathcal{E}_n = (\|S\|_q \leqslant c^{(S,q)})$, by Assumption 3.4:

$$\mathbb{E}_{P_n} \|S\|_q \left\| Z^{(k-1)} - \iota_X f^{(k-1)} \right\|_{q,2,n} \leqslant c^{(S,q)} \mathbb{E}_{P_n} \left(\left\| Z^{(k-1)} - \iota_X f^{(k-1)} \right\|_{q,2,n} \mid \mathcal{E}_n \right) + p_n C_n^{(S,q)} (D_{k-1} + \left\| f^{(k-1)} \right\|_\infty)$$

where D_k is the maximum value of $\left\| Z^{(k)} \right\|_{q,2,n}$ under $\mathcal{E}_n^c$.

Finally, we have naturally

$$\mathbb{E}_{P_n} \left(\left\| Z^{(k-1)} - \iota_X f^{(k-1)} \right\|_{q,2,n} \mid \mathcal{E}_n \right) \leqslant \mathbb{E}_{P_n} \left(\left\| Z^{(k-1)} - \iota_X f^{(k-1)} \right\|_{q,2,n} \right) = E_{k-1}$$

which gives us the desired recursion. We conclude using Lemma 3.1 to bound D_k and $\left\| f^{(k)} \right\|_\infty$. □

Theorem 3.5 (Detailed version of Thm. 3.2)**.** *Let Φ_θ be a S-GNN. Suppose that Assumption 3.5 holds. Then Assumption 2.2 is satisfied, with*

$$\Delta_n^a \lesssim \left(\sum_{\ell=1}^{k} \left[\prod_{p=\ell}^{k-1} u_k^1 \right] C_k \right) \Delta_n^{a,S} + \left(\prod_{\ell=1}^{k-1} u_\ell^1 \right) \Delta_n^{a,Z} \tag{3.38}$$

$$\Delta_n^A \lesssim \left(\sum_{\ell=1}^{k} \left[\prod_{p=\ell}^{k-1} u_k^1 \right] C_k \right) \Delta_n^{A,S} + \left(\prod_{\ell=1}^{k-1} u_\ell^1 \right) \Delta_n^{A,Z} \tag{3.39}$$

$$\Delta_n^z \lesssim \left(\prod_{\ell=1}^{k-1} u_\ell^1 \right) \Delta_n^{z,Z} \tag{3.40}$$

where

$$u_k^q = \left\| \theta_0^{(k)} \right\| + c_n^{(S,q)} \left\| \theta_1^{(k)} \right\|$$
$$C_k = \sum_{\ell=1}^{k} \left[\prod_{p=\ell}^{k-1} u_k^\infty \right] \left\| b^{(k)} \right\| + \left(\prod_{\ell=1}^{k-1} u_\ell^\infty \right) c_n^{(Z,\infty)}$$

For simplicity we do not give the details for Assumption 2.3 here.

Proof. Let $(A, Z), (A', Z') \in \mathcal{G}_n$ be two graphs, S, S' their associated operator, and $Z^{(k)}, Z'^{(k)}$ the

intermediate layers propagating by the GNN. Denote by $E_k = \sum_i \left\| Z_i^{(k)} - Z'^{(k)}_i \right\|$. We compute

$$\begin{aligned} E_k &\leqslant \left\|\theta_0^{(k)}\right\| \sum_i \left\| Z_i^{(k-1)} - Z'^{(k-1)}_i \right\| \\ &\quad + \left\|\theta_1^{(k)}\right\| \left(\left\| Z^{(k)} \right\|_{\infty,2} \sum_{ij} \left| S_{ij} - S'_{ij} \right| + \left\| S' \right\|_1 \sum_i \left\| Z_i^{(k-1)} - Z'^{(k-1)}_i \right\| \right) \\ &= u_k^1 E_{k-1} + C_k \left| S_{ij} - S'_{ij} \right| \end{aligned}$$

where $C_k = \left\| Z^{(k)} \right\|_{\infty,2}$, which by Lemma 3.1 is bounded using $\left\| S \right\|_\infty \leqslant c_n^{(S,\infty)}$ since $(A, Z) \in \mathcal{G}_n$. A simple recursion concludes the proof.

Assumption 2.3 is shown in the same manner, we do not detail the proof here. □

Proof of 3.1. Our strategy is to show that $d_{\Upsilon,n}(\Phi_\theta, \Phi_{\theta'}) \leqslant D_n \left\| \theta - \theta' \right\|_\Theta$, such that $\mathcal{N}(\mathcal{H}, \varepsilon, d_{\Upsilon,n}) \leqslant \mathcal{N}(\Theta, \varepsilon/D_n, \left\|\cdot\right\|_\Theta) \lesssim (D_\Theta D_n/\varepsilon)^{dim(\Theta)}$, where the last inequality is the classical covering numbers of a compact set of finite dimension.

For any $A, Z \in \mathfrak{G}_n$, denote by $Z^{(k)}$ the layer of the GNN with parameters θ and $Z'^{(k)}$ the layers with parameters θ', and $E_k = \left\| Z^{(k)} - Z'^{(k)} \right\|_{\infty,2}$. Note that the PEs are the same $Z^{(0)} = Z'^{(0)}$. We have, using Lemma 3.2,

$$\begin{aligned} E_k &\leqslant \max_i \left\| \rho_k \left(\theta_0^{(k)} Z_{i,:}^{(k)} + \theta_1^{(k)} \sum_j S_{ij} Z_{j,:}^{(k)} + b^{(k)} \right) - \rho_k \left(\theta'^{(k)}_0 Z'^{(k)}_{i,:} + \theta'^{(k)}_1 \sum_j S_{ij} Z'^{(k)}_{j,:} + b'^{(k)} \right) \right\| \\ &\leqslant \left\| Z^{(k)} \right\|_{\infty,2} \left\| \theta_0^{(k)} - \theta'^{(k)}_0 \right\| + \left\| \theta'^{(k)}_0 \right\| E_{k-1} \\ &\quad + \left\| S \right\|_\infty \left(\left\| Z^{(k)} \right\|_{\infty,2} \left\| \theta_1^{(k)} - \theta'^{(k)}_1 \right\| + \left\| \theta'^{(k)}_1 \right\| E_{k-1} \right) + \left\| b^{(k)} - b'^{(k)} \right\| \\ &\leqslant u_\infty E_{k-1} + C_k \left\| \theta - \theta' \right\|_\Theta \end{aligned}$$

where $u_\infty = D_\Theta(1 + \left\| S \right\|_\infty)$ and using Lemma 3.1, $C_k = (D_\Theta + C_n^{(Z,\infty)}) u_\infty^k$. We obtain a global bound in $D_\Theta^{2L} \left\| S \right\|_\infty^{2L} C_n^{(Z,\infty)}$, which concludes the proof. □

Proof of Prop. 3.2. Let $W = [w(x_i, x_j)]$. By [83], for any M there is a constant C_M such that with probability $1 - n^{-M}$,

$$\left\| S - W/n \right\| \leqslant C_M / \sqrt{n\alpha_n}$$

and thus, since $\left\| S - W/n \right\| \lesssim 1/\alpha_n$ with probability 1,

$$\mathbb{E}_{P_n} \left\| S - W/n \right\| \lesssim \frac{n^{-M}}{\alpha_n} + \frac{C_M}{\sqrt{n\alpha_n}} \lesssim \frac{1}{\sqrt{n\alpha_n}}$$

Hence

$$\|(S - W/n)\iota_X f\|_{2,2,n} \leqslant \|S - W/n\| \, \|\iota_X f\|_{2,2,n} \lesssim \frac{\|f\|_\infty}{\sqrt{n\alpha_n}}$$

Moreover, by [Lem. 4 NK14], for any $f \in \mathcal{B}(\mathcal{X})$, with probability at least $1 - \delta$

$$\|W\iota_X f - \iota_X \mathbf{S} f\|_{2,2,n} \lesssim \frac{\sqrt{d_x} + \sqrt{\log(1/\delta)}}{\sqrt{n}}$$

which similarly yields

$$\mathbb{E}_{P_n} \|W\iota_X f - \iota_X \mathbf{S} f\|_{2,2,n} \lesssim \sqrt{\frac{d_x}{n}}$$

which concludes the proof. Assumption 3.4 is similarly shown with [83].

Then, we have $\|S\|_1 = \|S_\infty\| = \sup_i \sum_j |S_{ij}| = \max_i d_i/(n\alpha_n)$ (since S is symmetric), where d_i are the degrees. Under the same probability as above, by [Lemma 3 NK12] the degrees are all such that

$$\forall i, \quad c_{\min} n\alpha_n \leqslant d_i \leqslant n\alpha_n \tag{3.41}$$

where we recall that $c_{\min}$ lower-bounds the degre *function*. Hence with probability $1 - p_n$, $c^{(S,1)} = O(1)$.

Finally, Assumption 3.5 is direct, again by using bounds on the degrees (3.41) for Δ_n^A. □

Proof of Prop. 3.3. The proof is similar to that oof Prop. 3.2, but with [Thm. 4 NK12] to bound $\left\|S - D_K^{-1/2} K D_K^{1/2}\right\|$ where D_K is the diagonal degrees of K, and [Lem. 5 NK14] to bound the second term in the triangular inequality.

The rest of the proof is similar, bounding the degrees with (3.41), and using for assumption 3.5

$$\begin{aligned}
\sum_{ij} \left|S_{ij} - S'_{ij}\right| &= \sum_{ij} \left| \frac{A_{ij}}{\sqrt{d_i d_j}} - \frac{A'_{ij}}{\sqrt{d'_i d'_j}} \right| \\
&= \sum_{ij} \left| \frac{A_{ij} - A'_{ij}}{\sqrt{d_i d_j}} \right| + \sum_{ij} \left| A'_{ij} \left(\frac{1}{\sqrt{d_i d_j}} - \frac{1}{\sqrt{d'_i d'_j}} \right) \right| \\
&= \sum_{ij} \left| \frac{A_{ij} - A'_{ij}}{\sqrt{d_i d_j}} \right| + \sum_{ij} \left| A'_{ij} \left(\frac{\sqrt{d_i d_j} - \sqrt{d'_i d'_j}}{\sqrt{d_i d_j d'_i d'_j}} \right) \right|
\end{aligned} \tag{3.42}$$

The first term is easy to bound using $d_i = O(n\alpha_n)$.

For the second term, when $A_{ij} = A'_{ij}$ except for i_0, j_0, we have $d_i = d'_i$ except for $i = i_0$ or j_0 for

which $d_i - d'_i = 1$, such that

$$\sum_{ij} \left| A'_{ij} \left(\frac{\sqrt{d_i d_j} - \sqrt{d'_i d'_j}}{\sqrt{d_i d_j d'_i d'_j}} \right) \right| = \sum_i \left| A'_{ij_0} \left(\frac{\sqrt{d_i}(\sqrt{d_{j_0}} - \sqrt{d'_{j_0}})}{\sqrt{d_i^2 d_{j_0} d'_{j_0}}} \right) \right| + \sum_j \left| A'_{i_0 j} \left(\frac{\sqrt{d_j}(\sqrt{d_{i_0}} - \sqrt{d'_{i_0}})}{\sqrt{d_j^2 d_{i_0} d'_{i_0}}} \right) \right|$$

and

$$\left| \sqrt{d_i}(\sqrt{d_{j_0}} - \sqrt{d'_{j_0}}) \right| = \frac{1}{2} \sqrt{\frac{d_i}{d}}$$

for some $d = O(n\alpha_n)$, such that $\sqrt{\frac{d_i}{d}} = O(1)$. We conclude with $\sum_i A_{ij_0} = O(n\alpha_n)$ and $\sqrt{d_j^2 d_{i_0} d'_{i_0}} = O(n^2\alpha_n^2)$.

When A and A' differ by one line and one column i_0, the reasoning is the same, except that $d_i - d'_i = 1$ when $i \neq i_0$, and $d_{i_0} - d'_{i_0} = O(n\alpha_n)$ otherwise.

The first term in (3.42) is again easily bounded by bounding the degrees. The second term involves

$$\sum_j \left| A'_{i_0 j} \left(\frac{\sqrt{d_{i_0} d_j} - \sqrt{d'_{i_0} d'_j}}{\sqrt{d_i d_j d'_i d'_j}} \right) \right|$$

which is easily in $O(1)$ by bounding all the degrees and using $\sum_j A'_{i_0 j} = d'_{i_0}$. Similarly when $j = i_0$.

The remaining term is more involved. Note that, when $i \neq i_0$, the event $d_i - d'_i = 1$ can only happen for $O(\alpha_n n)$ nodes, since this is the number of 1's in both $A_{i_0,:}$ and $A'_{i_0,:}$. Define $s_i = 1$ when this happens and 0 otherwise, such that $\sum_i s_i = O(n\alpha_n)$. By the same reasoning as above, we have

$$\sqrt{d_i d_j} - \sqrt{d'_i d'_j} = \sqrt{d_i}(\sqrt{d_j} - \sqrt{d'_j}) + \sqrt{d'_j}(\sqrt{d_i} - \sqrt{d'_i}) = O(s_i + s_j)$$

Finally,

$$\sum_{ij} A'_{ij}(s_i + s_j) = d'_i \sum_j s_j + d'_j \sum_i s_i = O(n^2\alpha_n^2)$$

and since $\sqrt{d_i d_j d'_i d'_j} = O(n^2\alpha_n^2)$ we have indeed $C^{(A,S)} = O(1)$. □

3.A.2 Technical Lemma

Lemma 3.1. *We have*

$$\left\|Z^{(k)}\right\|_{q,2,n} \leqslant \sum_{l=1}^{k} \left(\prod_{p=l}^{k-1} u_p\right) v_l + \left(\prod_{p=0}^{k-1} u_p\right) \left\|Z^{(0)}\right\|_{q,2,n} \tag{3.43}$$

where $u_k = \left\|\theta_0^{(k)}\right\| + \|S\|_q \left\|\theta_1^{(k)}\right\|$ *and* $v_k = \left\|b^{(k)}\right\|$.

Similarly,

$$\left\|f^{(k)}\right\|_{\infty} \leqslant \sum_{l=1}^{k} \left(\prod_{p=l}^{k-1} u_p\right) v_l + \left(\prod_{p=0}^{k-1} u_p\right) \left\|f^{(0)}\right\|_{\infty} \tag{3.44}$$

where $u_k = \left\|\theta_0^{(k)}\right\| + \|\mathbf{S}\|_{\infty\to\infty} \left\|\theta_1^{(k)}\right\|$.

Proof. Recursively, using $|\sigma(t)| \leqslant |t|$, Lemma 3.2 and $\left\|1_n b^\top\right\|_{q,2,n} = \|b\|$,

$$\left\|Z^{(k)}\right\|_{q,2,n} = \left\|\rho_k\left(Z^{(k-1)}\theta_0^{(k-1)} + SZ^{(k-1)}\theta_1^{(k-1)} + 1_n {b^{(k)}}^\top\right)\right\| \leqslant u_{k-1}\left\|Z^{(k-1)}\right\|_{q,2,n} + v_k$$

Hence the result. The proof for $f^{(k)}$ is similar. □

Lemma 3.2. *For* $X \in \mathbb{R}^{n\times d}$, $\theta \in \mathbb{R}^{d\times d'}$ *and* $S \in \mathbb{R}^{n\times n}$ *with* $S_{ij} \geqslant 0$, *we have*

$$\|X\theta\|_{q,2,n} \leqslant \|X\|_{q,2,n} \|\theta\| \tag{3.45}$$

$$\|SX\|_{q,2,n} \leqslant \|S\|_q \|X\|_{q,2,n} \tag{3.46}$$

Proof. The first inequality is obvious:

$$\|X\theta\|_{q,2,n} = \left(\frac{1}{n}\sum_{i=1}^{n}\left\|\theta^\top X_{i,:}\right\|^q\right)^{1/q} \leqslant \|\theta\| \left(\frac{1}{n}\sum_{i=1}^{n}\|X_{i,:}\|^q\right)^{1/q} = \|X\|_{q,2,n}\|\theta\|$$

For the second inequality, we denote by $v = [\|X_{i,:}\|] \in \mathbb{R}_+^n$, since $S_{ij} \geqslant 0$ we have

$$\begin{aligned}\|SX\|_{q,2,n} &= \left(\frac{1}{n}\sum_{i=1}^{n}\left\|\sum_j S_{ij}X_{j,:}\right\|^q\right)^{1/q} \\ &\leqslant \left(\frac{1}{n}\sum_{i=1}^{n}\left(\sum_j S_{ij}\|X_{j,:}\|\right)^q\right)^{1/q} = \frac{1}{n^{1/q}}\|Sv\|_q \leqslant \frac{1}{n^{1/q}}\|S\|_q\|v\|_q = \|S\|_q\|X\|_{q,2,n}\end{aligned}$$

□

Chapter 4

Bonus: Graph Neural Networks and Oversmoothing

Contents

Summary

This "bonus" chapter presents two contributions on the so-called *oversmoothing* phenomenon, a major limitation of GNNs that severely limits their depth.

— In the first contribution, I show that GNNs **do need to be deep**, even when they "oversmooth" when they are too deep. For this I introduce several estimation problems where two phenomena of oversmoothing but beneficial depth *co-exist*. I consider the *intuitions* behind these phenomena to be the most interesting part of this contribution.

— In the second contribution, I examine oversmoothing from an **optimization** point of view. I show that the backpropagation equations are also subjected to oversmoothing, and that this creates many "spurious" stationary points with provably high loss.

This chapter is decorrelated from the rest of the manuscript and presents a set of contributions that I made around the so-called *oversmoothing* phenomenon in Graph Neural Networks.

Broadly, oversmoothing refers to the fact that, since each layer of a GNN tends to "smooth out" the node representations towards the low frequencies of the graph, when the depth of GNNs increases, node representations become indistinguishable, which renders any kind of node prediction task impossible. Identified early as a main factor limiting GNNs' depth and their expressivity, oversmoothing has been the subject of a very large literature, from theoretical analysis to myriads of mitigating strategies, to position papers arguing that oversmoothing only appears due to poor benchmarking and coding practices.

In my work, I of course focused more on the theoretical aspects of oversmoothing. I mostly made two contributions that are a bit "on the side" compared to the mainstream oversmoothing literature [1]. First I examined the question: do we depth in GNNs? Or, more precisely, can oversmoothing *coexist* with an optimal depth of more than one layer? Second, I examined oversmoothing from an optimization point of view: since theoretically GNNs could learn to *not* oversmooth (by expanding their weights beyond the oversmoothing rate), why does it not happen in practice? As it turns out, these oversmoothing works have been a bit of a pet peeve project of mine, and they are published in two single-author papers [NK9, NK10].

I will first introduce some background tools on oversmoothing in Sec. 4.1, before presenting these two contributions in Sec. 4.2 and 4.3.

4.1 Background on oversmoothing

In this chapter we adopt a simplified S-GNNs, without skip connections or bias:

$$Z^{(k)} = \rho\left(SZ^{(k-1)}\theta^{(k)}\right) \tag{4.1}$$

Interestingly, many early GNNs were under this form, such as the GCN of Kipf et al [77], and they continue to be very strong baselines today. Note that here, crucially and unlike the previous chapters, *we do not assume that S is symmetric*, and generally, it will not be.

Oversmoothing happens when, by repeatedly applying such layers, $Z^{(k)}$ to a constant node representation: $Z^{(k)} \xrightarrow[k\to\infty]{} 1_n v^\top$ for some vector v. We will measure oversmoothing with the following quantity

$$\mathcal{E}(Z) := n^{-1/2} \min_{c\in\mathbb{R}^d} \left\| Z - 1_n c^\top \right\|_F \tag{4.2}$$

which is the (normalized) distance from Z to the subspace $\mathrm{span}(1_n)$. It is used in several landmark studies on oversmoothing [115]. Different variants of oversmoothing may exist, where the limit is not a constant node representation anymore, but a different vector. Hence we introduce the

1. which mostly consists of proposing strategies to avoid oversmoothing

generalization for some $u \in \mathbb{R}^n$:

$$\mathcal{E}_u(X) := n^{-1/2} \min_{c \in \mathbb{R}^d} \left\| X - uc^\top \right\|_F \tag{4.3}$$

such that $\mathcal{E} = \mathcal{E}_1$. For instance, by repeatedly applying $S = D^{-1/2}AD^{-1/2}$ to a node representation, the limit is $u = \sqrt{deg(A)}$ the square root of the degree vector. However, in my work I focus on oversmoothing towards $u = 1_n$, which is the most common definition of oversmoothing [131].

Propagation matrix. The basic idea to theoretically prove oversmoothing is the following: at each application of S, the node representations are brought "closer" to the dominant eigenvector (or eigenspace) of S, which *de facto* becomes the oversmoothed limit. Hence, the basic hypothesis to obtain vanilla oversmoothing towards $u = 1_n$ is to consider *stochastic matrices*, i.e. that satisfies $S1_n = 1_n$.

Assumption 4.1 (Stochastic matrix)**.** *Assume that S is a stochastic irreducible*[2] *matrix.*

As Markov chain transition matrices, irreducible stochastic matrices have a unique stationary distribution $\pi \in \mathbb{R}^n_+$ with $\pi^\top 1_n = 1$ which is a left-eigenvector for the 1 eigenvalue: $\pi^\top S = \pi^\top$. In the rest of the chapter, we will also assume that $\pi_i > 0$ and denote by $\mu_\pi = \frac{\max(\pi_i)}{\min(\pi_i)}$.

It is known that

$$S^k \xrightarrow[k\to\infty]{} 1_n \pi^\top \tag{4.4}$$

Typically, this convergence is exponentially fast, which we formulate in the following assumption.

Assumption 4.2 (Convergence speed)**.** *There is a constant $C_S > 0$ and $0 < \lambda_S < 1$ such that for all $k > 0$,*

$$\left\| S^k - 1_n \pi^\top \right\|_2 \leqslant C_S \lambda_S^k \tag{4.5}$$

When S is diagonalizable $S = U^{-1}\Lambda U$ and 1 is a simple eigenvalue, this assumption is satisfied with $\lambda_S = |\lambda_2|$ the absolute value of the second largest eigenvalue of S, which is indeed smaller than 1 by the Perron-Frobenius theorem, and $C_S = \|U\|_2 \left\|U^{-1}\right\|_2$.

The quintessential example of stochastic message-passing matrix for GNNs is the random walk transition matrix:

$$S = D^{-1}A \tag{4.6}$$

It is indeed diagonalizable for connected graph, with eigenvalues $1 - \lambda_i^L$, where λ_i^L are the eigenvalues of the symmetric normalized Laplacian $L = L_{sym} = Id - D^{-1/2}AD^{-1/2}$, since $S =$

2. a matrix is irreducible if it cannot be permuted to an upper-triangular matrix. If S has the same zero pattern as A, it is irreducible if the graph is strongly connected

$D^{-1/2}(Id-L)D^{1/2}$. Hence, in this case $\lambda_S = 1-\lambda$, where $\lambda > 0$ is the spectral gap of L. However, S is *not symmetric* in this case, and in particular it is possible [3] to have $\|S\|_2 > 1$. This will be important below. On the contrary, when S is symmetric it is *bi*-stochastic, and computations are much simplified. However bistochastic matrices are quite rare in MPGNNs, despite the fact that they are quite common in adjacent fields such as e.g. decentralized optimization [20], where they are called *gossip matrices*.

As the matrix S may contract Z towards constant representation, the weights $\theta^{(k)}$ may on the contrary "expand" the representation in various ways. Hence an important quantity in oversmoothing studies is the maximal operator norm of the weights, that we will denote by $t = \max_k \left\|\theta^{(k)}\right\|$.

Using this proof strategy based on contracting eigenspaces, convergence towards oversmoothing typically happens at an exponential rate. In fact, this is even in the *definition* of oversmoothing adopted by certain authors [131]. Following this, we first formulate oversmoothing as a definition.

Definition 4.1. *We will say that* ***oversmoothing*** *holds with rate* $0 < \lambda_f < 1$ *and constant* $C_f > 0$ *if*

$$\mathcal{E}(Z^{(k)}) \leqslant C_f(\lambda_f t)^k \tag{4.7}$$

where $t = \max_k \left\|\theta^{(k)}\right\|_2$.

That is, at each layer S may contract by some λ_f, while the weights may expand by t. If the product of the two is smaller than 1, then the node representations will typically contract towards 1_n. This authorizes *some* expansion $t > 1$, but t is still bounded.

Interestingly, and despite the overwhelmingly large literature on oversmoothing, the most general case is still not proven, even if oversmoothing is almost always observed in practice. Instead, we have the following.

Theorem 4.1 ([NK10])**.** *Suppose that Assumptions 4.1 holds. If either:*

1. $\rho = Id$ *and Assumption 4.2 holds, in which case we consider* $\lambda_f = \lambda_S$*, or*
2. *there is* $\lambda_f < 1$ *such that*

$$\left\|J^{\perp} S J^{\perp}\right\|_2 \leqslant \lambda_f \tag{4.8}$$

where $J^{\perp} := Id - 1_{n\times n}/n$ *is the orthogonal projector onto span*$(1_n)^{\perp}$.

Then forward oversmoothing holds with rate λ_f *and some constant* C_f.

In other words, oversmoothing is proven in two cases: either the GNN is *linear* with $\rho = Id$, or S contracts every directions but the constant one.

The linear case is very often adopted in the literature [159, 117, 31] (and in the previous section),

3. More precisely, $\|S\|_2 = 1$ if and only if S is symmetric!

as computations are greatly simplified and the constant limit can actually be characterized (see after). In this case, the mild Assumption 4.2 directly yields forward oversmoothing, and we recall that it holds as soon as the graph is strongly connected, with $\lambda_S = |\lambda_2| < 1$ the second-largest eigenvalue of S.

For non-linear ρ, the assumption (4.8) is satisfied *when S is symmetric* (and thus bistochastic) and the eigenvalue 1 is simple. In this case, it is easy to see that (4.8) holds with $\lambda_f = |\lambda_2| < 1$, since S is *orthogonally* diagonalizable. However, **when S is non-symmetric**, it does *not* necessarily hold. Indeed, for many real datasets it is not satisfied even if oversmoothing is still observed in practice [NK10].

Hence there is no theoretical proof of oversmoothing in the "non-linear ρ, non-symmetric stochastic S" case, which is arguably an important missing piece in the literature. A potential solution would be to characterize the relationship between the eigenspaces of S and the non-linearity ρ (especially if it is simple such as ReLU), but this remains an open question as of today.

Limit signal. Remark that while $Z^{(k)}$ converges to a constant vector across nodes, due to the presence of the non-linear activation function ρ it is generally difficult to explicitly characterize *what* this limit constant vector is (although some fixed point theorems may apply in certain circumstances [182]). This is one of the reasons why many theoretical analyses of oversmoothing focus on the linear case $\rho = Id$, where the limit *can* be described with $S^k \to 1_n \pi^\top$ [159, 117, 31], as we do in the next section.

Mitigating oversmoothing Naturally, there has been many strategies to address the oversmoothing issue. They can be broadly sorted in two categories. First, *skip connections*, that is, putting back the term "$+Z^{(k)}$" in the MPGNNs (1.13) is a natural way to preserve the node representations and potentially avoiding contraction towards a constant. Surprisingly, such skip connections were not part of early GCN and GNN models, such that many baselines still operate without. While nowadays skip connections start to become a well-adopted practice, they are not without cons: they may lead to signal/gradient explosion instead, in particular at initialization [4]. Moreover, vanilla skip connections often have limited efficiency in mitigating oversmoothing, hence many exotic variants of skip connections have been proposed without any real consensus [86, 119, 31]. In practice, most GNNs remain quite shallow, with less than 10 layers.

The other natural strategy to mitigate oversmoothing is to perform some kind of renormalization at each layer to enforce the node representation to not collapse. Again, there are a few variants of normalization with no real consensus and no real adoption in practice [175].

4. especially since batching is hard for graphs and BatchNorm is not as used as in regular deep learning

4.2 Not too little, not too much: do we need depth?

The results in this section are from [NK9].

In this section, I examine the following question: since deep GNNs oversmooth, do GNNs actually need to be deep? More precisely, there *are* theoretical results that prove that depth is needed for, e.g., GNN expressivity [161]. But such results do not put a limit on GNNs' depth: they examine very specific, theoretical architectures (that of course do not oversmooth) and show that such GNNs with unbounded width and depth are expressive or can implement some graph algorithms. On the other hand, it is known that MPGNNs that are used in practice, and in particular (4.1), *do* tend to oversmooth. So the question becomes: can we have a model that provably needs to be deep, while still oversmoothing in the limit $k \to \infty$? The difficulty in this question is that we need to analyze the behavior of such GNN at *finite* depth, knowing that at infinite depth it will oversmooth.

Linear GNNs. To express these results we place ourselves in specific settings. We adopt the random walk message-passing matrix (4.6), and linear GNN model:

$$S = D^{-1}A, \quad \rho = Id$$

Hence, according to the previons section and Theorem 4.1, when the graph is connected oversmoothing do occur with rate $\lambda_f = 1 - \lambda_L$ where $0 < \lambda_L < 1$ is the spectral gap (smallest non-zero eigenvalue) of the normalized Laplacian.

Since the GNN is linear, it can be rewritten with a single parameter, and we simply write

$$\Phi_\theta(A, Z) = Z^{(L)} = S^L Z^{(0)} \theta$$

Graph model. Here we consider an LPM on a set $\mathcal{X} \subset \mathbb{R}^{d_x}$ with *deterministic edges* $a_{ij} = w(x_i, x_j)$, with an ajusted Gaussian kernel

$$w(x, x') = \varepsilon + e^{-\frac{1}{2}\|x - x'\|^2}$$

Note here that $\mathcal{X}$ is not necessarily bounded, hence we add a small $\varepsilon > 0$ to the Gaussian kernel to avoid unnecessary complication with kernels that would go to 0.

Here $Z^{(0)} = Z$, and the node features z_i are a linear transform of the x_i:

$$z_i = f^{(0)}(x_i) = M^\top x_i$$

for some matrix $M \in \mathbb{R}^{d_x \times d_0}$. As described in the previous chapters, intuitively we have $d_0 < d_x$, such that M is *not* information-preserving in any way [5], and information *is* lost between the unknown x_i and observed z_i.

Loss. Here we will always consider the square loss for simplicity,

$$\hat{\mathcal{R}}(\Phi_\theta) = \hat{\mathcal{R}}(\theta) = \frac{1}{n}\left\|Y - S^L Z^{(0)}\theta\right\|^2 \tag{4.9}$$

For our results, we will consider a Ridge regularization on the parameter $\theta \in \mathbb{R}^{d_0}$:

$$\min_\theta \hat{\mathcal{R}}(\theta) + \lambda \|\theta\| \tag{4.10}$$

We can implement all the framework of the previous chapter to describe the convergence of $\hat{\mathcal{R}}$. We define the operator corresponding to $S = D^{-1}A$:

$$[\mathbf{S}f](x) = \frac{1}{d(x)}\int w(x,x')f(x')dP_x(x')\,. \tag{4.11}$$

The matrix $S = D^{-1}A$ is naturally $\mathfrak{P}$-convergent towards $\mathbf{S}$: note that we have not shown this in previous chapters since we have limited ourselves to symmetric matrices for simplicity, but it is a known result [129]. Hence using the result of the previous chapter Φ_θ is $\mathfrak{P}$-convergent towards

$$\Phi_\theta(\mathbf{S}, f^{(0)}) = \theta^\top \mathbf{S}^L f^{(0)} \tag{4.12}$$

and thus the optimal risk $\mathcal{R}_\infty(\Phi_\theta)$ is equal to

$$\mathcal{R}_{\mathrm{ML}}(\Phi_\theta(\mathbf{S}, f^{(0)})) = \mathcal{R}_{\mathrm{ML}}(\theta) = \mathbb{E}\left\|y - \theta^\top [\mathbf{S}^L f^{(0)}](x)\right\|^2 \tag{4.13}$$

> *Note that in the original paper [NK9], I do not express these quantities within the new framework of this manuscript, but describe a transductive SSL problem instead. However the proofs rely on $n \to \infty$, and can easily be adapted to obtain the results presented here.*

Optimal depth? Let us go back to our original question: is there an optimal depth for the GNN (4.1) before oversmoothing kicks in? Denote by $\theta^{(L)}$ the optimal parameter for a depth L, that minimizes the empirical risk with a Ridge regularizaiton:

$$\theta^{(L)} = \operatorname*{argmin}_\theta \frac{1}{n}\left\|Y - S^L Z\theta\right\|^2 + \lambda\|\theta\| \tag{4.14}$$

5. typically, it does not satisfy the so-called Johnson-Lindenstrauss lemma that would state that distances between the x_i are almost preserved

we thus evaluate the quantity

$$\mathcal{R}^{(L)} = \mathcal{R}_{\mathrm{ML}}(\theta^{(L)}) \tag{4.15}$$

So the question becomes: on the one hand, for a zero-depth GNN (i.e. just a regression on the node features) we have $\mathcal{R}^{(0)}$, when the number of layers go to ∞ and the GNN oversmoothes we obtain $\mathcal{R}^{(\infty)}$, is there an optimal depth $L^\star$ such that $\mathcal{R}^{(L^\star)} < \min(\mathcal{R}^{(0)}, \mathcal{R}^{(\infty)})$?

Already, when the GNN oversmoothes, it is easy to show that

$$\mathcal{R}^{(L)} \xrightarrow[L\to\infty]{} \mathcal{R}^{(\infty)} = \mathrm{Var}(y) + O(\lambda) + O(1/\sqrt{n}) \tag{4.16}$$

Hence, for λ small enough and n large enough, the limit risk is approximately the variance of the labels. This is unsurprising, as it is the best possible result for constant features $\mathbf{S}^L f \to 1(\cdot)$. Hence, generally we will have $\mathcal{R}^{(0)} \leqslant \mathcal{R}^{(\infty)}$, and the goal often becomes to show that there exists $\mathcal{R}^{(L^\star)} < \mathcal{R}^{(0)}$.

In the next two subsections, we examine a regression problem and a classification problem (but still with the square loss for simplicity). We will indeed formulate sufficient conditions under which there exists $\mathcal{R}^{(L^\star)} < \min(\mathcal{R}^{(0)}, \mathcal{R}^{(\infty)})$, but more importantly, it is the *intuition* behind these results that is useful: why does a *finite* smoothing of the node features useful, when repeated smoothing leads to oversmoothing?

4.2.1 Regression

In this section, we consider a regression problem, with true labels:

$$y_i = {\theta^\star}^\top x_i \,.$$

We consider that the labels have no noise for simplicity, and assume that the latent variables are Gaussian

$$P_x = \mathcal{N}(0, \Sigma) \tag{4.17}$$

for some positive-definite covariance matrix Σ.

For a symmetric positive semi-definite matrix $T \in \mathbb{R}^{d\times d}$, we define the following function

$$R_{\mathrm{reg.}}(T) \overset{\mathrm{def}}{=} (\Sigma^{\frac{1}{2}}\theta^\star)^\top \left(\mathrm{Id} - T^{\frac{1}{2}} M(\lambda \mathrm{Id} + M^\top T M)^{-1} M^\top T^{\frac{1}{2}}\right)^2 (\Sigma^{\frac{1}{2}}\theta^\star) \in \mathbb{R}_+ \tag{4.18}$$

where we recall that M is the projection matrix to obtain the node features $z = M^\top x$. Our main result can be stated informally as follows, it is detailed in the next section along with a sketch of proof.

Theorem 4.2 ([Thm 2 NK9])**.** *Take any $\delta > 0$, and suppose that*

$$R_{\text{reg.}}(\Sigma) > R_{\text{reg.}}((\text{Id} + \Sigma^{-1})^{-2}\Sigma) \tag{4.19}$$

If ε is sufficiently small and n is sufficiently large, then with probability $1-\delta$, there is $L^\star > 0$ such that $\mathcal{R}^{(L^\star)} < \min(\mathcal{R}^{(0)}, \mathcal{R}^{(\infty)})$.

4.2.1.1 Sketch of proof

As mentioned before, it is easy to show that $\mathcal{R}^{(0)} < \mathcal{R}^{(\infty)}$ with high probability. To show the theorem, I simply prove that $\mathcal{R}^{(1)} < \mathcal{R}^{(0)}$ with high probability, which is sufficient to show the existence of an optimal $L^\star \geqslant 1$.

For this, we will show a rigorous non-asymptotic bound on $\mathcal{R}^{(1)}$, based on the intuitive fact that one step of smoothing of Gaussian data with a Gaussian kernel results in approximately Gaussian data, but *with a modified covariance*. This new covariance can be advantageous for the estimation problem, as we will intuitively explain in the next section. We define

$$\Sigma^{(k)} = (\text{Id} + \Sigma^{-1})^{-2k}\Sigma$$

Theorem 4.3 ([Thm 3 and 4 NK9])**.** *With probability at least $1-\delta$,*

$$\mathcal{R}^{(0)} = R_{\text{reg.}}(\Sigma) + \mathcal{O}\left(\frac{\|\Sigma\| \left\|\theta^\star\right\|^2 d_x \sqrt{\log(1/\delta)}}{(\lambda + \lambda_{\min})\sqrt{n}}\right) \tag{4.20}$$

where $\lambda_{\min} = \lambda_{\min}(M^\top \Sigma M)$. Similarly, with probability at least $1-\delta$,

$$\mathcal{R}^{(1)} = R_{\text{reg.}}(\Sigma^{(1)}) + \mathcal{O}\left(C\varepsilon^{1/5}\right) + \mathcal{O}\left(\frac{C' \log n \sqrt{d_x + \log(1/\delta)}}{(\lambda + \lambda_{\min})\sqrt{n}}\right) \tag{4.21}$$

where $C = \text{poly}(\|\Sigma\|, e^{d_x}, |\text{Id} + \Sigma|)$, $C' = \text{poly}(\varepsilon^{-1}, \|\Sigma\|, \|\theta^\star\|)$ and $\lambda_{\min} = \lambda_{\min}(M^\top \Sigma^{(1)} M)$.

We have thus shown that $\mathcal{R}^{(i)} \approx R_{\text{reg.}}(\Sigma^{(i)})$ for $i = 0, 1$, when ε is small and n is large. Hence, if $R_{\text{reg.}}(\Sigma^{(1)}) < R_{\text{reg.}}(\Sigma)$, which is exactly the hypothesis (4.19), the theorem is proven.

As mentioned above, the proof relies on the fact that the smoothed latent variables $X^{(k)} = S^k X$ are approximately Gaussian, but with covariance $\Sigma^{(k)}$ instead, and since the GNN is linear $Z^{(k)} = X^{(k)} M$ the risk can be approximated in closed-form. The difficulty lies in handling the *non-independence* of the various quantities due to the smoothing, and that is also why I was only able to prove the rigorous approximation for $k = 1$, which is sufficient to prove the theorem but not for further results: computing the exact $L^\star$, etc.

4.2.1.2 Intuition and exact computation in dimension $d = 2$

We proved above that $x^{(1)}$ behaves approximately like $(\mathrm{Id} + \Sigma^{-1})^{-1}x$, whose covariance is $\Sigma^{(1)}$. By applying repeated smoothing we can extrapolate that $x^{(k)}$ behaves like $(\mathrm{Id} + \Sigma^{-1})^{-k}x$, such that

$$\mathcal{R}^{(k)} \approx R_{\text{reg.}}(\Sigma^{(k)})\,.$$

As mentioned above, the rigorous proof of this fact is still out-of-reach for now. Now, the interesting question becomes: why would having Gaussian data with covariance $\Sigma^{(k)}$ instead of Σ help for regression?

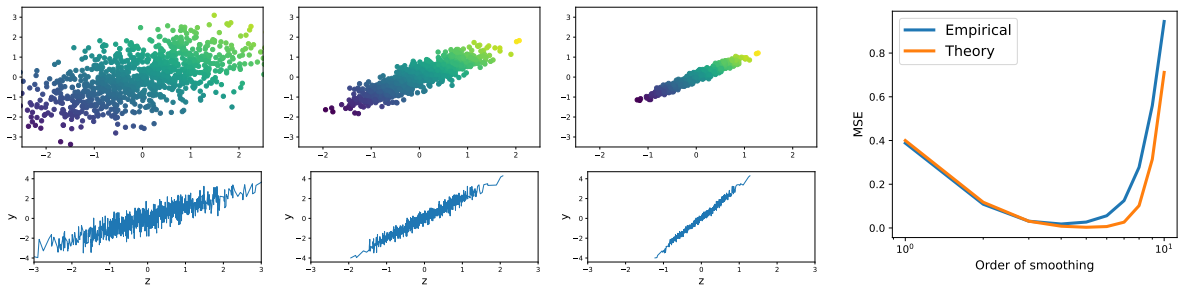
Observe that the matrix $\Sigma^{(k)}$ has the same eigendecomposition as Σ, but where every eigenvalue λ_i is replaced by $\lambda_i^{(k)} = (1+1/\lambda_i)^{-2k}\lambda_i$. This can be interpreted as follows: when $\lambda_i \gg 1$ is large, $\lambda_i^{(1)} \sim \lambda_i$, while if $\lambda_i \ll 1$ is small, $\lambda_i^{(1)} \sim \lambda_i^{2k+1}$. Hence smoothing **shrinks the directions of the small eigenvalues *faster* than that of the large ones**. Thus, if $\theta^\star$ is mostly aligned with the eigenvectors of large eigenvalues, shrinking the small eigenvalues may *reduce unwanted noise* that emerges when projecting the node features $z = M^\top x$. On the other hand, if all eigenvalues of Σ are equal, then $\Sigma^{(k)} \propto \Sigma$, and smoothing *does not help*, since in the limit $\lambda = 0$, the risk is invariant to scaling $R_{\text{reg.}}(aS) = R_{\text{reg.}}(S)$. Worse, we will see on an example below that smoothing can actually degrade the performance when $\theta^\star$ is unproperly aligned.

We illustrate this in dimension $d = 2$. Consider the following settings: $d = 2, p = 1$, Σ has two eigenvalues $\lambda_1 \gg 1$ and $\lambda_2 \ll 1$, with respective eigenvectors $u_1 = [1,1]/\sqrt{2}$ and $u_2 = [-1,1]/\sqrt{2}$, and $\theta^\star$ is fully correlated with the first eigenvector: $\theta^\star = bu_1$. Finally, $M^\top = [1,0]$ is the projection on the first coordinate. This situation is represented in Fig. 4.1. In this case, we can compute explicitely:

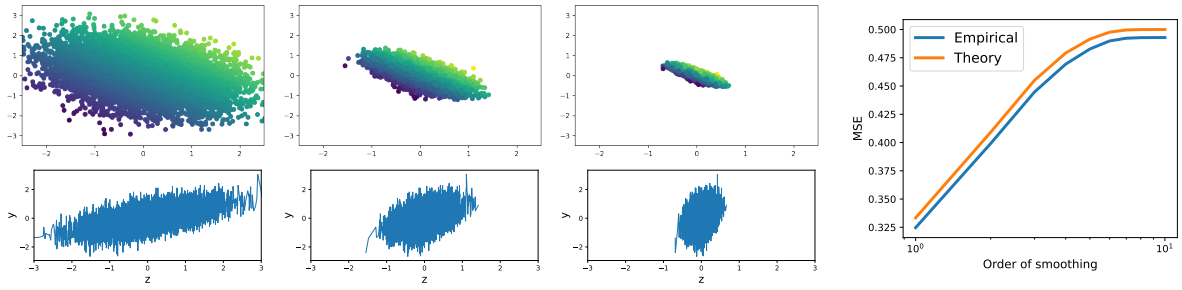
$$\mathcal{R}^{(L)} \approx R_{\text{reg.}}(\Sigma^{(L)}) = \lambda_1 b^2 \frac{(2\lambda + \lambda_2^{(L)})^2 + \lambda_2^{(L)}\lambda_1^{(L)}}{(2\lambda + \lambda_1^{(L)} + \lambda_2^{(L)})^2} \tag{4.22}$$

So, if $\lambda_2^{(L)}$ decreases faster than $\lambda_1^{(L)}$, this function will first decrease to a minimum of approximately $\lambda_1 b^2 \left(\frac{2\lambda}{2\lambda+\lambda_1^{(L^\star)}}\right)^2$ (when $\lambda_2^{(L)} \approx 0$), before increasing again to $\lambda_1 b^2 = \|\theta^\star\|_\Sigma^2 = \mathbb{E}\,|y|^2 \approx \mathcal{R}^{(\infty)}$. This is illustrated in Fig. 4.1, for $\lambda_1 = 2$ and $\lambda_2 = 1/2$, where we empirically observes a minimum $L^\star$ that matches rather well the one predicted by (4.22). On the other hand, when $\lambda_1 < \lambda_2$, we see in Fig. 4.1 that smoothing does not help at all, as the direction of $\theta^\star$ corresponds to an eigenvalue that will shrink faster with smoothing.

Interestingly, this success/failure example can somewhat be linked with homophily/heterophily. At first glance it seems that the very regular random graph model always results in homophilic graphs. This is partly true, however is it also possible that nodes linked by a "strong" edge (with



(a)



(b)

Figure 4.1 – Illustration of mean aggregation smoothing on the regression example described in Sec. 4.2.1.2. **For both subfigures: First three figures on the left, top:** *unobserved* latent variables $X^{(L)}$ in dimension $d = 2$ where the colors are the Y; **bottom:** observed node features $Z^{(L)} = X^{(L)}M$ in dimension $p = 1$ on the x-axis, labels Y on the y-axis. **From left to right**, three order of smoothing $L = 0, 1$ and 2 are represented. **Figure on the right:** comparison of empirical and theoretical MSE given by (4.22) with respect to order of smoothing L. **Subfig. a:** $\lambda_1 = 2, \lambda_2 = 1/2$ (smoothing does help), **Subfig. b:** $\lambda_1 = 1/2, \lambda_2 = 1$ (smoothing does not help). Figure from [NK9].

a high weight) have very different labels, which can be said to be a (toy) example of heterophily. Indeed, in this example, when the regression vector is in the direction of the eigenvector corresponding to a high eigenvalue, then beneficial smoothing appears, as it reduces the noise in the observed node features (Fig. 4.1a). However, when the regression vector is instead in the low-eigenvalue direction, then close-by latent variables have very different labels, the graph is more "heterophilic", and beneficial smoothing does not appear (Fig. 4.1b).

4.2.2 Finite smoothing: classification

In this section, we examine a simple classification problem for two balanced classes with Gaussian distribution with identity covariance [6] $\mathcal{N}_\mu = \mathcal{N}(\mu, Id)$. The distribution of the labels and latent variables is:

$$(x, y) \sim (1/2)(\mathcal{N}_\mu \otimes \{1\} + \mathcal{N}_{-\mu} \otimes \{-1\}) \tag{4.23}$$

That is, with equal probability x is drawn from $\mathcal{N}_\mu$ and $y = 1$, or $x \sim \mathcal{N}_{-\mu}$ and $y = -1$. As $\|\mu\|$ increases, the problem become simpler: there is an extensive literature on this problem [37, 152, 13]. Note that in this case z_i are also Gaussian, with mean $\nu \overset{\text{def}}{=} M^\top \mu$ or $-\nu$ and identity covariance.

Our main result is the following.

Theorem 4.4 ([Thm 5 NK9])**.** *Take any $\delta > 0$. If ε is sufficiently small, and $\|\mu\|, n$ are sufficiently large, and $\left\|M^\top \mu\right\| > 0$, then with probability $1 - \delta$, there is $L^\star > 0$ such that $\mathcal{R}^{(L^\star)} < \min(\mathcal{R}^{(0)}, \mathcal{R}^{(\infty)})$.*

Note that we have assumed $\|\mu\|$ to be sufficiently large here, such that the classification problem on the x_i is generally very "easy". However, we do *not* assume that $\left\|M^\top \mu\right\|$ is large (just non-zero), and the classification problem on the z_i alone may be very difficult.

4.2.2.1 Sketch of proof and intuition

As in the previous section, it will be easy to show that $\mathcal{R}^{(0)} < \mathcal{R}^{(\infty)}$ with high probability, and we will prove that $\mathcal{R}^{(1)} < \mathcal{R}^{(0)}$ with high probability. We define the following function

$$R_{\text{cl.}}(t) = \frac{(t + \lambda)^2 + t \|\nu\|^2}{(t + \lambda + \|\nu\|^2)^2} \tag{4.24}$$

Recalling that $\nu = M^\top \mu \in \mathbb{R}^p$, the next result is the equivalent of Thm 4.3.

6. We note that this is not a *difficult* problem *per se*, and that *linear regression with the MSE* is certainly not the method of choice to solve it. The interest here is mostly illustrative

Theorem 4.5 ([Thm 5 and 6 NK9])**.** *With probability at least* $1-\delta$*,*

$$\mathcal{R}^{(0)} = R_{\text{cl.}}(1) + \mathcal{O}\left(\frac{\|\nu\|^4 p\sqrt{\log(1/\delta)}}{\sqrt{n}}\right) \tag{4.25}$$

With probability at least $1-\delta$*,*

$$\mathcal{R}^{(1)} = R_{\text{cl.}}(1/4) + \mathcal{O}\left(C\left(\varepsilon^{\frac{1}{4}} + \frac{1}{\varepsilon^3}e^{-\frac{\|\mu\|^2}{4}}\right)\right) + \mathcal{O}\left(\frac{C'(\log n)(\sqrt{d_x + \log(1/\delta)})}{\sqrt{n}}\right) \tag{4.26}$$

where $C = \text{poly}(\|\mu\|, e^{d_x})$ *and* $C' = \text{poly}(\varepsilon^{-1}, \|\mu\|)$*.*

As before, having $R_{\text{cl.}}(1/4) < R_{\text{cl.}}(0)$ suffices to prove the theorem, and this time this happens as soon as $\|\nu\| > 0$, that is, the matrix M is not orthogonal to the mean μ. We remark that in the expression for $\mathcal{R}^{(1)}$, unlike the previous section where the error terms vanished in the limit $\varepsilon \to 0, n \to \infty$, here there is a non-zero error term due to $\|\mu\|$. Hence, for instance, the discrepancy between the empirical observations and the theory in Fig. 4.2 compared to Fig. 4.1.

This time, the intuition is the following. As can be seen in the proof in [NK9], *when the communities are sufficiently separated*, after one round of smoothing the data *in the first community* $y = 1$ behaves approximately like $\frac{x+\mu}{2}$, whose distribution is $\mathcal{N}(\mu, \text{Id}/4)$. For the second community it is similar with $\frac{x-\mu}{2} \sim \mathcal{N}(-\mu, \text{Id}/4)$. Hence, up to some error $\mathcal{O}\left(e^{-\|\mu\|^2/4}\right)$, **the smoothed features in each community have the same mean but a reduced variance** Id/4, thus the limit risk $R_{\text{cl.}}(1/4)$. Intuitively, as can be seen in the proof and the experiments, the approximation error is due to the communities *getting closer to each other*. In other words, **the communities shrink faster than they collapses together**, and this reflects on the projected node features. An illustration of this phenomenon is given in Fig. 4.2.

Beyond $k = 1$, intuitively applying a second smoothing would transform the first community to approximately $\frac{x_i+3\mu}{4} \sim \mathcal{N}(\mu, \text{Id}/16)$, and so on. However the errors due to the communities "leaking" into each other accumulate: if we look at the proof, we can postulate without rigorous proof that we get something like

$$\mathcal{R}^{(k)} \approx R_{\text{cl.}}(4^{-k}) + \mathcal{O}\left(\sum_{\ell=0}^{k-1} e^{-\frac{\|\mu\|^2}{2(1+4^{-\ell})}}\right) \tag{4.27}$$

Unlike the expression (4.22), the term $R_{\text{cl.}}(4^{-k})$ is strictly decreasing when k increases. Oversmoothing is thus modelled by the error term, for which we do not have an exact expression, and for which we suspect that the quality of approximation degrades as k increases. Nevertheless, we evaluate this expression on an example in Fig. 4.2 (with an adjusted multiplicative constant for the error term in (4.27)) and find that it is a reasonably good approximation, at least for

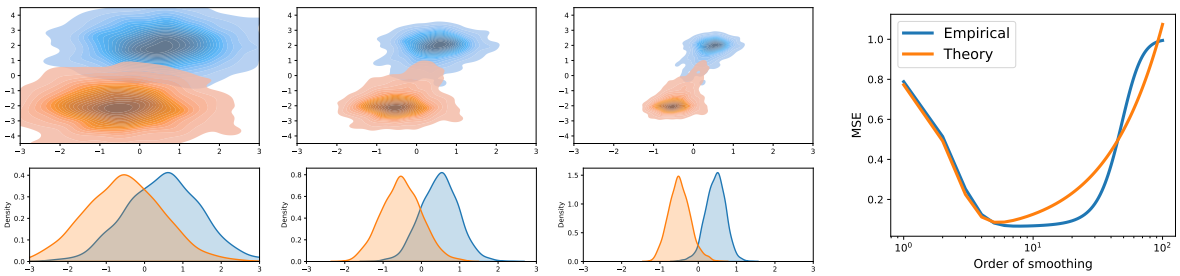


Figure 4.2 – Illustration of mean aggregation smoothing on a classification task with two Gaussians with dimensions $d = 2$, $p = 1$, where M projects on the first coordinate. **First three figures on the left, top:** density of *unobserved* latent variables $X^{(k)}$ in dimension $d = 2$; **bottom:** density of observed node features $Z^{(k)} = X^{(k)}M$ in dimension $p = 1$. **From left to right**, three order of smoothing $k = 0, 1$ and 2 are represented (recall that the smoothing is agnostic to the labels, we cannot perform in-community smoothing). **Figure on the right:** comparison of empirical and theoretical MSE given by (4.27) with respect to order of smoothing k. For low k, the node features communities are indeed more and more separated, and learning improves. Figure from [NK9].

small k.

4.2.3 Discussion

The original motivation for this contribution was to examine the discrepancy between the models of GNNs that were strictly expressive with no limit to their depth, and GNNs in practice that easily and obviously oversmooth. In retrospect, the most useful outcome of this study really was novel *intuitions* on the mechanisms of something as simple as smoothing with a Gaussian kernel. The key element here is the *discrepancy of information* between the x_i and the z_i, whose smoothing allows to somewhat reduce. For regression, smoothing in the x space allows to reduce the noise in the z space if done in the right direction, as the shrinking is not done at the same speed in every direction. For classification, even if communities are well-separated in the x space, they may appear very close in the z space. After which, smoothing in the x space allows to concentrate them faster than they collapse together, helping the learning process.

Remark that these observations on the discrepancy of information between x and z were a key motivation for my subsequent works on random graphs, up until the framework presented in this manuscript, including Chapter 2.

4.3 Backward oversmoothing: an optimization point of view

The results in this section are from [NK10].

Let us now turn to the second contribution: an *optimization* point of view on deep GNNs and oversmoothing. Indeed, as remarked in the introduction, if at some point during training

$t > 1/\lambda$, then the GNN does not necessarily oversmooth [47]. Hence the GNN could learn to *not* oversmooth [162], but this does not really happen in practice: does depth has an effect on GNNs' optimization process? As is generally the case in deep learning, the answer is yes, but the fine mechanisms are *not* the usual suspects, aka vanishing (or exploding) gradients. Here I will describe phenomena that are *specific* to GNNs, which is the main takeaway message of this work [NK10].

4.3.1 Preliminaries

In this section, we consider any (fixed) graph G with message-passing matrix S that satisfies Ass. 4.1 and 4.2, and are no longer concerned with random graphs and stochasticity.

GNN: forward. Let us first introduce additional notations to descirbe the GNN (4.1), which will we useful for the rest:

$$\begin{aligned}
&F^{(k)} = SZ^{(k-1)} \in \mathbb{R}^{n\times d_{k-1}}, && \text{(message-passing)}\\
&\zeta^{(k)} = F^{(k)}\theta^{(k)} \in \mathbb{R}^{n\times d_k} && \text{(weights multiplication)}\\
&Z^{(k)} = \rho(\zeta^{(k)}), && \text{(activation function)}\\
&out = \zeta^{(L)} \in \mathbb{R}^{n\times d_L}, && \text{(output after } L \text{ layers)}
\end{aligned}$$

For reasons that will be made clear just after, we call $F^{(k)}$ the **forward signal**, going "from" input ($k = 0$) to output ($k = L$). As in the previous chapters, we assume that the input node features $Z^{(0)} = Z$ are bounded $\left\|Z^{(0)}_{i,:}\right\|_2 \leqslant D_z$.

Loss function. Here, for convenience, we express the empirical risk as a loss function $\mathcal{L} : \mathbb{R}^{n\times d_L} \to \mathbb{R}_+$:

$$\mathcal{L}(\zeta) = \tfrac{1}{n}\sum_{i=1}^{n} \ell_i(\zeta_i) \tag{4.28}$$

for some functions $\ell_i : \mathbb{R}^{d_L} \to \mathbb{R}_+$ and $\zeta_i = \zeta_{i,:}$ for short. In particular, our results would be applicable for any loss function of this form, although ERM is the overwhelming example. We assume the following bound on the loss, which is a stronger and more convenient version of the assumption 2.1.

Assumption 4.3. *There are constants $D_{\mathcal{L},1}, D_{\mathcal{L},0} \geqslant 0$ such that $\left\|\frac{\partial \ell_i(z)}{\partial z}\right\|_2 \leqslant D_{\ell,1}\|z\|_2 + D_{\mathcal{L},0}$.*

It is easy to compute that the square loss satisfies Assumption 2.1 with $D_{\mathcal{L},1} = D_{\mathcal{L},0} = 1$, while cross-entropy satisfies Assumption 2.1 with $D_{\mathcal{L},1} = 0$ and $D_{\mathcal{L},0} = C + 1$ where C is the number of classes.

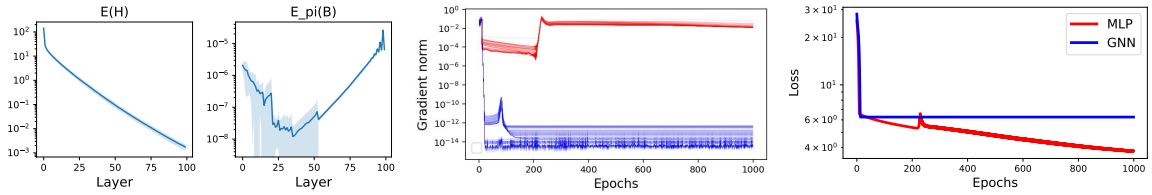


Figure 4.3 – GNN with $S = D^{-1}A$ or MLP with 100 layers, on the WikiCS dataset [107]. **Left**: illustration of forward oversmoothing by measuring $\mathcal{E}(\zeta^{(k)})$ (left) and backward oversmoothing by measuring $\mathcal{E}_\pi(B^{(k)})$ (right). **Center**: norms of gradients $\partial\mathcal{L}/\partial W^{(k)}$ for each layer wityh respect to epoch, for MLP (red) and GNN (blue). **Right**: corresponding losses. Figure from [NK10].

GNN: backward. First-order optimization methods rely on gradients $\frac{\partial\mathcal{L}}{\partial\theta^{(k)}}$, generally computed by backpropagation. For GNNs, the backpropagation equations are

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\theta^{(k)}} &= {F^{(k)}}^\top B^{(k)}, \text{ where}\\ B^{(k)} := \frac{\partial\mathcal{L}}{\partial\zeta^{(k)}} &= \rho'(\zeta^{(k)}) \odot \left(S^\top B^{(k+1)}(\theta^{(k+1)})^\top\right)\\ \text{and } B^{(L)} &= \frac{\partial\mathcal{L}}{\partial\zeta^{(L)}}\end{aligned} \tag{4.29}$$

We have written these equations here to emphasize the object $B^{(k)} = \frac{\partial\mathcal{L}}{\partial\zeta^{(k)}} \in \mathbb{R}^{n\times d_k}$ that we call the **backward signal** [7]. It is indeed a d_k-dimensional signal on the nodes of the graph, that satisfies a recursion equation, going "from" output ($k = L$) to input ($k = 0$). Generally, the "initialization" $B^{(L)} = \frac{\partial\mathcal{L}}{\partial\zeta^{(L)}}$ at the last layer represents the predictive error of the GNN: for instance, for the MSE loss, it is directly $B^{(L)} = \frac{1}{n}(\zeta^{(L)} - Y)$.

The full gradient $\frac{\partial\mathcal{L}}{\partial\theta^{(k)}} = {F^{(k)}}^\top B^{(k)}$ is a product between the forward and the backward signal in the node dimension. As we will see, just as depth has as effect on the forward signal with oversmoothing [8], it also has an effect on the backward signal, but in more subtle ways.

4.3.2 Backward Oversmoothing

Due to the multiplication by $S^\top$ from the output layer $k = L$ to the input $k = 1$, the backward signal will naturally also be (over)smoothed. Note however that, as S is not symmetric, the limit will be different than the forward signal.

We need the following assumption on the non-linearity.
Assumption 4.4. *Assume that ρ' is D_ρ-Lipschitz.*

Note that this excludes ReLU (which is not fully differentiable anyway) but includes smoothed

7. It has several names in the literature, but here we emphasize its similarity to the forward signal.
8. That we will now refer to as *forward* oversmoothing

variants like softplus [59]. To formulate backward oversmoothing, recall that we have formulated forward oversmoothing as a *definition*, and that we explicitly assume that it holds, without precising if Thm 4.1 is indeed statisfied.

Theorem 4.6 ([Thm. 2 NK10])**.** *Assume that forward oversmoothing holds with rate λ_f and constant C_f, and that Assumptions 4.1, 4.3 and 4.4 hold. Then:*

$$\mathcal{E}_\pi(B^{(k)}) \leqslant C_L(\lambda_S t)^{L-k} + C'_L \lambda_f^k$$

where $C_L = \frac{C_S A_L}{n}$, $C'_L = \frac{C_S C_f D_\rho \mu_\pi t^L A_L}{(1-\lambda_S \lambda_f s)n}$ *and* $A_L = D_{\mathcal{X}} D_{\mathcal{L},1} t^L + D_{\mathcal{L},0}$.

Unsurprisingly, when multiplying by $S^\top$, the limit is now π: we recall that $\mathcal{E}_\pi$ is the distance to $span(\pi)$. The bound on $\mathcal{E}_\pi(B^{(k)})$ involves two terms: one that decreases exponentially in k, the other one in $L-k$. Hence backward oversmoothing happens *at the middle layers* (Fig. 4.3, left), when *both* k and $L-k$ are high. This is due to the term $\rho'(\zeta^{(k)})$ in the backpropagation equations (4.29) which modulates the rows of $B^{(k)}$. *When the forward signal is oversmoothed* (high k), this modulation is almost constant, and backpropagation is simple: $B^{(k)} \propto S^\top B^{(k+1)} {\theta^{(k+1)}}^\top$. It leads to backward oversmoothing after sufficiently many iterations (high $L-k$). On the contrary, at low k, the modulation $\rho'(\zeta^{(k)})$ can be high. Hence, it is the *interaction* between forward and backward smoothing that leads to backward oversmoothing at the middle layers.

Theorem 4.6 describe a non-trivial relationship between λ and t, since t^L is also present in the multiplicative constants. We can simplify it by instantiating the loss.

Corollary 4.1 ([Cor. 1 NK10])**.** *Take $q = 2$ for a regression problem with the MSE loss, or $q = 1$ for a classification problem with the cross-entropy loss. Consider that the assumptions of Theorem 4.6 hold, denote $\lambda = \max(\lambda_S, \lambda_f)$. If $t \leqslant \lambda^{-\alpha}$ where $0 \leqslant \alpha < 1 - \sqrt{1 - \frac{1}{q}}$, then for all $q\alpha < \beta < \frac{1-q\alpha}{1-\alpha}$,*

$$\mathcal{E}_\pi\left(B^{(\beta L)}\right) \lesssim \lambda^{(\beta - q\alpha)L} + \lambda^{(1-q\alpha-(1-\alpha)\beta)L} \xrightarrow[L\to\infty]{} 0$$

In other words, taking the layer index $k = \beta L$ proportional to the depth L, with sufficiently low α we obtain an explicit exponential rate.

Backward oversmoothing is not (necessarily) vanishing gradients. While one might feel that backward oversmoothing is detrimental for GNN training, it is not clear exactly why this is true. In [182], the authors relate oversmoothing to vanishing gradients, however: a) they assume Gaussian weights, and b) to our understanding, by assuming contracting Jacobians they place themselves in a case where *the forward signal itself converges to zero* with increasing layer index. This of course results in oversmoothing (a zero signal is of course constant).

However, under our general framework this is not necessarily true, as we show below.

Proposition 4.1. *[Prop. 1 NK10] There is a graph that satisfies Ass. 4.1 with a symmetric S with $\lambda_2 < 1$ such that the following holds. For all L, there is a GNN of depth L that satisfies all the assumptions of Thm. 4.6 with $t \leqslant 1$ such that for all k:*

$$\left\|\partial \mathcal{L}/\partial \theta^{(k)}\right\|_F = 1 \tag{4.30}$$

That is, we can have both backward oversmoothing (by Thm. 4.6) but with arbitrarily high gradients. Hence the answer is *not* vanishing gradients, at least not *automatically*. The main contribution of this work is to relate backward oversmoothing to *(spurious) stationary points*.

4.3.3 Spurious stationary points

While backward oversmoothing is not directly responsible for vanishing gradients, in practice we observe that deep GNNs tend to quickly and easily get stuck in local minima (Fig. 4.3, center). This prompts us to analyze the *stationary points* of deep GNNs. We first introduce the following definition of near-stationary points [141].

Definition 4.2 (δ-stationary point)**.** *The weight $\theta^{(k)}$ is said to be at a δ-**stationary point** if*

$$\left\|\partial \mathcal{L}/\partial \theta^{(k)}\right\|_F \leqslant \delta \tag{4.31}$$

*If this is true for all $1 \leqslant k \leqslant L$, then the GNN is said to be at a **global** δ-**stationary point**.*

Note that stationary points can be detrimental – e.g., "bad" local minima, saddle points – or simply the sign of successful training: after all, characterizing the rates of convergence to near-stationary points is the main goal of non-convex optimization [141, 67].

We now turn to the main result of this section. For convenience we define the function

$$\xi_q(\alpha) = \tfrac{1-(2q+1)\alpha+q\alpha^2}{2(1-\alpha)} \tag{4.32}$$

and note that $\xi_q(\alpha) > 0$ for $\alpha \leqslant 1 + \frac{1}{2q} - \sqrt{1+\frac{1}{4q^2}}$, which is roughly equal to 0.38 for $q = 1$ and 0.21 for $q = 2$.

Theorem 4.7 ([Thm. 3 NK10])**.** *Assume that forward oversmoothing holds with rate λ_f and constant C_f, and that Assumptions 4.1, 4.3 and 4.4 hold. Denote $\lambda = \max(\lambda_S, \lambda_f)$. Suppose that $t \leqslant \lambda^{-\alpha}$, with $\xi_q(\alpha) > 0$. Assume that the output forward signal is not zero: $\frac{1}{\sqrt{n}}\left\|F^{(L)}\right\|_F \geqslant D_F$, and that **the parameter $\theta^{(L)}$ is at a δ_θ-stationary point**. For $\delta > 0$, if*

$$L \gtrsim \frac{\log(1/(D_F\delta))}{\xi_q(\alpha)\log(1/\lambda)} \quad \text{and} \quad \delta_\theta \lesssim \lambda^{\alpha L} D_F \delta \tag{4.33}$$

then ***the GNN is at a global δ-stationary point.***

This theorem shows that, for sufficiently deep GNNs, under some mild assumptions, **every δ_θ-stationary point *for the output layer* L is also a *global* δ-stationary point**, for some small δ related to δ_θ and L. In other words, as soon as the last layer of the GNN is "trained", gradients vanish *at every layers* and the GNN hardly trains anymore. As one might guess, the last (linear) layer is exceedingly easy to train, which results in many easy-to-reach stationary points. Note that, while backward oversmoothing is the underlying mechanism for this result as we will explain next, δ-stationarity indeed happens *at every layers*.

4.3.3.1 Sketch of proof

As in the previous section, the *intuition* behind the proof of the theorem is at least as useful as the result itself.

In forward oversmoothing, the signal $Z^{(k)}$ converges to a constant node representation when k increases. As mentioned in the previous section, in the linear case $\rho = Id$ it is easy to characterize the limit by leveraging $S^k \to 1_n \pi^\top$, which is not possible anymore in the non-linear case. The situation is different for backward oversmoothing. As mentioned earlier, *when the forward signal is oversmoothed* (near the output layer), the term $\rho'(\zeta^{(k)})$ is approximately constant, and **the update equation is approximately** $B^{(k)} \propto S^\top B^{(k+1)} (\theta^{(k+1)})^\top$, which is nothing more than a linear GNN! Hence we *can* approximate the limit representation of the backward oversmoothed signal, and it relates to the *average* of the output error: at the middle layers, $B^{(k)} \propto \pi 1_n^\top B^{(L)} (\theta^{(L)} \dots \theta^{(k+1)})^\top$, using $(S^\top)^k \to \pi 1_n^\top$.

We thus obtain the following theorem, which is the core of the proof of Thm 4.7.

Theorem 4.8 ([Thm. 4 NK10])**.** *Take $q = 2$ for a regression problem, or $q = 1$ for a classification problem. Assume that forward oversmoothing holds with rate λ_f and constant C_f, and that Assumptions 4.1, 4.3 and 4.4 hold. Denote $\lambda = \max(\lambda_S, \lambda_f)$. If $t \leqslant \lambda^{-\alpha}$ with $\xi_q(\alpha) > 0$, the GNN is at a global δ-stationary point, with*

$$\delta \lesssim \lambda^{\xi_q(\alpha) L} + \lambda^{-\alpha L} \left\| 1_n^\top B^{(L)} \right\|_2 \tag{4.34}$$

The key component here is the norm of $1_n^\top B^{(L)} = \frac{1}{n} \sum_i \frac{\partial \ell_i(\zeta_i)}{\partial \zeta_i}$, which is indeed the *average* of the output error of the GNN. One notable aspect of this result is that it applies *to every layer* k, even though oversmoothing is the underlying phenomenon. Indeed, first, middle and last layers are indeed treated differently in the proof

— in the last layer (high k), the forward signal is oversmoothed and the updates of $B^{(k)}$ are

approximately linear. Hence

$$\frac{\partial \mathcal{L}}{\partial \theta^{(k)}} = {F^{(k)}}^\top B^{(k)} \approx v 1_n^\top (S^\top)^{L-k} B^{(L)} (\theta^{(L)} \dots \theta^{(k+1)})^\top = v 1_n^\top B^{(L)} (\theta^{(L)} \dots \theta^{(k+1)})^\top$$

for some vector v, since S is stochastic. We see $1_n^\top B^{(L)}$ appear.

— in the middle layers, both $B^{(k)}$ and $F^{(k)}$ are oversmoothed. As described above, the updates to B are linear, and we can actually compute the limit: $B^{(k)} \approx c\pi 1_n^\top B^{(L)} (\theta^{(L)} \dots \theta^{(k+1)})^\top$ for some constant c. Again, we see $1_n^\top B^{(L)}$ appear directly in $B^{(k)}$ this time.

— In the first layers, we control the growth of the norm of $B^{(k)}$ from the middle layers, and again relate to $\left\| 1_n^\top B^{(L)} \right\|$.

To conclude with the proof of Theorem 4.7, we then bound the average output error $\left\| 1_n^\top B^{(L)} \right\|_2$ when the last layer is at a stationary point: *since the output of the GNN is almost constant due to forward oversmoothing*, when the last layer is trained, the *average* error automatically vanishes. In summary, it is indeed *the subtle interplay between forward and backward oversmoothing*, as well as the observation that updates on B are linear even when ρ is non-linear, that allows to prove the theorem.

4.3.3.2 Stationary points with high loss

As mentioned above, stationary points are not inherently bad: reaching a stationary point *is* the goal of optimization [141, 67], in the hope that it is a good (local) minimum in the non-convex case. Hence, we still haven't proven that backward oversmoothing was inherently bad *per se*, but it becomes increasingly clear that a model that stops training as soon as the last layer is trained is indeed problematic, and that it is what we somewhat observe in practice.

More rigorously, can we leverage Theorem 4.7 or 4.8 to show that deep GNNs can have a global δ-stationary point, for an arbitrary *high* loss? Such a result would be quite unusual in non-convex optimization, as the existence of stationary point is generally decorrelated from the value of the loss in itself. We give an example below.

Corollary 4.2 ([Cor. 2 NK10])**.** *Assume that forward oversmoothing holds with rate λ_f and constant C_f, and that Assumptions 4.1, 4.3 and 4.4 hold. Denote $\lambda = \max(\lambda_S, \lambda_f)$. Consider the regression case, and assume that the labels are centered:*

$$\sum_i y_i = 0$$

Denote by $\rho_y^2 = \frac{1}{n} \sum_i \|y_i\|_2^2$. Suppose that $t \leqslant \lambda^{-\alpha}$, with $\xi_2(\alpha) > 0$, and the last weight is such

that $\|W_L\|_2 \leqslant \varepsilon_{W_L}$. *For* $\delta, \varepsilon > 0$, *if*

$$L \gtrsim \frac{\log(1/\delta)}{\xi_2(\alpha)\log(1/\lambda)} \text{ and } \varepsilon_{W_L} \lesssim \min\left(\lambda^{2\alpha L}\delta,\ \frac{\lambda^{\alpha L}}{\rho_y}\varepsilon\right)$$

then the GNN is at a ***global*** δ***-stationary point****, and* ***the loss satisfies***

$$\mathcal{L}(\zeta^{(L)}) \geqslant \rho_y^2/2 - \varepsilon \tag{4.35}$$

In other words, for a centered regression problem and sufficiently deep GNNs, it is easy to find a flat region of near-stationary points such that *the loss is uniformly high*, close to the variance of the labels. While we constructed this particular example around $W_L = 0$, it is likely that many other flat regions with high loss could be constructed on the same principles, by leveraging Thm. 4.8.

4.3.3.3 GNNs are not MLPs

It is known, both in theory and in practice, that the landscape of MLPs is riddled with local minima and saddle points [153, 179, 40, 168]. Can we make certain that the stationary points identified by Thm. 4.7 are indeed *specific* to GNNs? The following proposition answers by the affirmative.

Proposition 4.2 (MLPs are not GNNs)**.** *Consider a regression problem, in the MLP case* $S = Id$. *For all* n *there is a data distribution such that the following holds. For all depth* $L > 0$, *there is an MLP that satisfies Assumptions 4.1, 4.3 and 4.4, such that* $t \leqslant 1$, $\frac{1}{\sqrt{n}}\left\|F^{(L)}\right\| = 1$, $\frac{\partial \mathcal{L}}{\partial \theta^{(L)}} = 0$, *but there exists* k *such that*

$$\left\|\frac{\partial \mathcal{L}}{\partial \theta^{(k)}}\right\| = 1 \tag{4.36}$$

This proposition indeed contradicts Thm. 4.7 for MLPs: there is a regression problem and an arbitrarily deep MLP that satisfy all the Assumptions of Thm. 4.7 – including that $F^{(L)}$ is lower-bounded and W_L is at a stationary point – such that at least one gradient does not vanish. Hence, our results are not another characterization of existing stationary points in deep learning, but are instead *specific* to GNNs.

4.4 Conclusion, discussion

Oversmoothing is a phenomenon that has been identified early in GNNs, that may severely limit their depth. Beyond the vanilla definition and the cases where it can be theoretically shown, the modern view on oversmoothing is often that:

— it is a rather vague definition, and while deep GNNs are definitely hard to implement, "oversmoothing" may refer to an ensemble of difficulties instead.

— vanilla oversmoothing can be countered by simple strategies: skip connections, normalizations, but deep GNNs remain hard to implement, and performance does not really increase with depth. This is unclear why.

In my point of view, "oversmoothing" is one manifestation of an underlying phenomenon: in GNNs, there is an inherent *interaction* between elements of the loss that are normally independent in ML, and this has many consequences (not necessarily detrimental): in the forward pass, the backpropagation, and so on. As such, recent position papers challenge the classical interpretation of oversmoothing [79].

In my case, these two papers on oversmoothing [NK9, NK10] were an interesting side project, that I do not think I will pursue further, except maybe more focused on GNN optimization in general. That said, there are still interesting open questions left.

Open questions

— As we have seen, forward oversmoothing is not entirely solved, and this is an important missing piece in the literature. The key would be to analyze the interaction between *non-linearities* and the eigenspaces of *non-symmetric* stochastic matrices, which is certainly useful in its own right.

— there is not much theoretical analysis of the mitigation strategies for oversmoothing, skip connection and normalization. The rare existing theoretical analyses of skip connections are highly non-trivial [31]. They might be adapted to the backward case.

— Can we design better *optimization algorithms* for GNNs, e.g. not based on vanilla backpropagation [174]?

Conclusion and a few words

I do not have a background in graph theory, or even graph machine learning. My first paper on graphs dates from two years after my PhD, in 2019 [NK11], when I stumbled upon a few papers on GNNs (well after they were invented, and already at the peak of their "hype") and thought that the topic looked fun. I did not plan to seriously start working on, or limit myself to, graphs. To this day, I still meet researchers that I knew from my PhD days that think of me as a specialist of data compression in ML (which I never really was either). But a few years later, I am still working on graphs, and just started an ERC grant on the topic. Nevertheless, I strongly feel that my "non-graph" background is reflected in my approach: paradoxically, I try to escape graphs as soon as I can (!) to fall back into the comfortable world of continuous prediction functions, concentration inequalities, and so on. I had the chance to encounter collaborators and colleagues that were interested in similar approaches, and to somewhat witness (and humbly participate in) random graphs and GNNs coming together in the literature.

When I started writing this manuscript, I had results on GNNs lying around about convergence towards functions, universality, concentration, and so on. I had initially intended Chapter 2 to be a section of the introduction, to give mathematical background on regular ML. But by attempting to formalize a bit node-tasks GML, I ended up patching most missing notions and developing a somewhat "novel[9]" framework in which all my existing results could neatly fit. Crucially, it drew a rigorous link between generalization in GML and generalization in regular ML, something that has always been evoked in my previous works but never really formalized.

Now what? To some extent, I feel that this work brings a sort of *closure* to what has been my main research topic in the last few years. Of course, there are still many outstanding open questions, of which we have outlined a few at the end of each chapter, but the *basic, fundamental* framework has been laid out, and that has been one of my main goal for a long time. All my previous results fit within this framework, and though they are by no mean complete, the "spirit" of future results should not change too much, apart from improving rates, proof techniques, treating more examples, and so on. So, while my work on generalization is not entirely finished, I still have to ask: now what? What's the next "big" step?

9. I am still unsure about to which extent these definitions can be called "novel", I think of this chapter as appropriate reformulations of general objects.

Beyond random graphs. My current opinion (it may change at any moment) is that the *Latent Position Models* random graph themselves are probably an over-simplifying step here. They could certainly be improved upon and complexified, while still staying in the realm of LPMs: for instance, node features and labels could depend on more than a single latent variable, not be independent from the graph structure, and so on. Or we could stray further from LPMs, and develop more complex **generative models**.

Generative models for graphs have been a popular topic in the last few years [49, 114, 75, 155, 29] – although incomparably so with generative AI for images or text naturally. Popular applications are often found in chemistry, where potential molecules may be generated for e.g. drug discovery [155]. As such, most existing methods handle *small, highly structured graphs*. Unsurprisingly, **diffusion models** [10] [155, 49, 29] are the most popular models, where the diffusion can happen at many stages: directly on the graph [155], on latent variables [178] (we then come back to an LPM but with highly non-iid latent variables), on some other encoding [49]...

However, the **theory** of such models is inexistant. There are outstanding questions though, as usual mostly coming from mixing ML objects with *permutation invariance/equivariance*. For instance, a pure noise, permutation invariant distribution over independent edges is necessarily an Erdös-Rényi model, and by our convergence results, applying any GNN to a large Erdös-Rényi graph results in a constant output, hence the "denoising" step on diffusion model is increasingly difficult. From preliminary discussions [11], it seems that **exchangeability** [12] really becomes the central concept in these models: for instance, starting from a point cloud of i.i.d. noisy samples, after a few steps of equivariant denoising, the distribution is only *exchangeable* and not i.i.d. anymore. Hence, I strongly feel that my next major endeavor could be titled something like

"From independence to exchangeability"

Allowing for exchangeable latent variables and exchangeable edges could be key to unlocking more powerful, realistic random graph models (through diffusion processes or equivalent), while remaining theoretically tractable. Fortunately, exchangeable random variables and random "arrays" (like graphs) have been studied in statistics for a long time [76], and powerful results such as de Finetti's or Aldous-Hoover's theorems exist to help characterize them.

Exchangeability is far from being a novel notion for random graphs [150, 116], or even for certain classes of deep neural nets [18], but I am convinced that its rigorous introduction in *modern Graph ML*, with advanced generative models and GNNs, could be a theoretical and practical breakthrough. As before, it is the *convergence* of tools and ideas from different communities that

10. roughly, diffusion models work by progressively adding noise to clean data and learning to denoise it, then starting from pure noise and denoising it several times to obtain new data
11. with T. Vayer, and in the context of the PhD of A. Jamadandi co-supervised with A. Roumy
12. that is, the probability is invariant by permutation of the data

did not use to speak to each other that is attractive to me. Of course, I describe here only early reflections on a new topic with which I am not familiar, that are probably somewhat naive. But I already see interesting questions, that may well still be open.

Open questions(?)

— Can an i.i.d. pure noise distribution really be mapped to an exchangeable distribution in an equivariant manner? (existence, unicity...)

— Do equivariant denoising maps used in practice generalize to larger graphs/point clouds? (such *transferability* to larger objects is not usual for regular diffusion models...)

— Can we define an efficient equivariant denoising model on graphs with exchangeable (instead of independent) edges? Leveraging Aldous-Hoover theorem for random exchangeable arrays?

— Can all the generalization analysis in this manuscript be done for exchangeable random graphs? What is the equivalent of $\mathfrak{P}$-convergence? What are the links with regular ML?

References of the Candidate

[NK1] M. Cordonnier, **N. Keriven**, N. Tremblay, et S. Vaiter. Convergence of Message Passing Graph Neural Networks with Generic Aggregation On Large Random Graphs. *Journal of Machine Learning Research (JMLR)*, pages 1–39, 2023.

[NK2] M. Cordonnier, **N. Keriven**, N. Tremblay, S. Vaiter, et S. J. V. L. A. Dieudonné. Seeking universal approximation for continuous limits of graph neural networks on large random graphs. In *Asilomar Conference on Signals, Systems and Computers*, pages 1–5, 2024.

[NK3] M. Gjorgjevski, **N. Keriven**, S. Barthelmé, et Y. D. Castro. Node regression on latent position random graphs via local averaging. pages 1–43, 2024.

[NK4] M. Gjorgjevski, **N. Keriven**, S. Barthelme, Y. D. Castro, S. Barthelmé, et Y. D. Castro. Graphical kernel ridge regression in latent position models. In *GRETSI*, 2025.

[NK5] H. Jaquard et **N. Keriven**. Statistical consistency of discrete-to-continuous limits of determinantal point processes. *Preprint*, 2026.

[NK6] A. Joly et **N. Keriven**. Graph coarsening with message-passing guarantees. In *Advances in Neural Information Processing System (NIPS)*, 2024.

[NK7] A. Joly, **N. Keriven**, et A. Roumy. Taxonomy of reduction matrices for graph coarsening. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[NK8] **N. Keriven**. Entropic Optimal Transport in Random Graphs. *SIAM Journal on Mathematics of Data Science (SIMODS) (In press)*, 2022.

[NK9] **N. Keriven**. Not too little, not too much: a theoretical analysis of graph (over)smoothing. *Advances in Neural Information Processing Systems (NeurIPS)* ***Oral***, 2022.

[NK10] **N. Keriven**. Backward oversmoothing: why is it hard to train deep graph neural networks? *Preprint*, pages 1–20, 2026.

[NK11] **N. Keriven** et G. Peyré. Universal Invariant and Equivariant Graph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1–19, 2019.

[NK12] **N. Keriven** et S. Vaiter. Sparse and Smooth: improved guarantees for Spectral Clustering in the Dynamic Stochastic Block Model. *Electronic Journal of Statistics*, 16(1):1330 – 1366, 2022.

[NK13] **N. Keriven** et S. Vaiter. What functions can Graph Neural Networks compute on random graphs? The role of Positional Encoding. In *Advances in Neural Information and Processing Systems (NeurIPS)*, 2023.

[NK14] **N. Keriven**, A. Bietti, et S. Vaiter. Convergence and Stability of Graph Convolutional Networks on Large Random Graphs. In *Advances in Neural Information and Processing Systems (NeurIPS)* ***Spotlight***, pages 1–26, 2020.

[NK15] **N. Keriven**, A. Bietti, et S. Vaiter. On the Universality of Graph Neural Networks on Large Random Graphs. In *Advances in Neural Information and Processing Systems (NeurIPS)*, 2021.

[NK16] M. Theveneau et **N. Keriven**. Stability of Entropic Wasserstein Barycenters and application to random geometric graphs. *GRETSI*, 2022.

BIBLIOGRAPHY

[1] U. Alon et E. Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representation (ICLR)*, 2021.

[2] R. K. Ando, T. Zhang, R. K. Ando, et T. Zhang. Learning on graph with laplacian regularization. In *NIPS 2006: Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 25–32, 2006.

[3] E. Araya et Y. de Castro. Latent distance estimation for random geometric graphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[4] A. Athreya, D. E. Fishkind, M. Tang, C. E. Priebe, Y. Park, J. T. Vogelstein, K. Levin, V. Lyzinski, Y. Qin, et D. L. Sussman. Statistical inference on random dot product graphs: A survey. *Journal of Machine Learning Research*, 18:1–92, 2018.

[5] L. Babai. Graph isomorphism in quasipolynomial time. *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2016.

[6] F. Bach. Breaking the curse of dimensionality with convex neural networks. *Journal of Machine Learning Research*, 18:1–53, 2017.

[7] F. Bach. Learning theory from first principles. *Book Draft*, page 229, 2021.

[8] G. Bar-Shalom, Y. Eitan, F. Frasca, et H. Maron. A flexible, equivariant framework for subgraph gnns via graph products and graph coarsening. *arXiv preprint arXiv:2406.09291*, 2024.

[9] A. L. Barabási et R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[10] F. Barbero, A. Velingker, A. Saberi, M. Bronstein, et F. D. Giovanni. Locality-aware graph-rewiring in gnns. In *International Conference on Learning Representation (ICLR)*, 2024.

[11] M. Bechler-Speicher, B. Finkelshtein, F. Frasca, L. Müller, J. Tönshoff, A. Siraudin, V. Zaverkin, M. M. Bronstein, M. Niepert, B. Perozzi, M. Galkin, et C. Morris. Position: Graph learning will lose relevance due to poor benchmarks. 2 2025.

[12] M. Belkin et P. Niyogi. Convergence of laplacian eigenmaps. *Advances in Neural Information Processing Systems*, pages 129–136, 2007.

[13] M. Belkin et K. Sinha. Polynomial learning of distribution families. In *IEEE 51st Annual Symposium on Foundations of Computer Science*. Ieee, 2010.

[14] J. Bento et S. Ioannidis. A family of tractable graph distances. 2018.

[15] B. Bevilacqua, F. Frasca, D. Lim, B. Srinivasan, C. Cai, G. Balamurugan, M. M. Bronstein, et H. Maron. Equivariant subgraph aggregation networks. pages 1–47, 2021.

[16] M. Black, Z. Wan, A. Nayyeri, et Y. Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *International Conference on Machine Learning (ICML)*, 2023.

[17] M. Black, Z. Wan, G. Mishne, A. Nayyeri, et Y. Wang. Comparing graph transformers via positional encodings. Technical report, 2024.

[18] B. Bloem-Reddy et Y. W. Teh. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21:1–61, 2020.

[19] G. Bouritsas, F. Frasca, S. Zafeiriou, et M. M. Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:657–668, 2023.

[20] S. Boyd, A. Ghosh, B. Prabhakar, et D. Shah. Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52:2508–2530, 6 2006.

[21] M. M. Bronstein, J. Bruna, T. Cohen, et P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv:2104.13478*, 2021.

[22] J. Bruna, W. Zaremba, A. Szlam, et Y. LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representation (ICLR)*, 2014.

[23] J. Böker, R. Levie, N. Huang, S. Villar, et C. Morris. Fine-grained expressivity of graph neural networks. pages 1–59, 6 2023.

[24] R. B. D. C, D. Madhivadhani, A. Sumathi, et L. S. Grace. Graph neural networks for modeling ecological networks and food webs. *The Scientific Temper*, 16:3832–3838, 3 2025.

[25] N. D. Cao et T. Kipf. Molgan: An implicit generative model for small molecular graphs. 2018.

[26] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, et P. Velickovic. Combinatorial optimization and reasoning with graph neural networks. *IJCAI International Joint Conference on Artificial Intelligence*, pages 4348–4355, 2021.

[27] O. Chapelle, B. Schölkopf, et A. Zien. *Semi-Supervised Learning*. 2010.

[28] S. Chatterjee. Matrix estimation by universal singular value thresholding. *Annals of Statistics*, 43: 177–214, 2015.

[29] H. Chen, C. Xu, L. Zheng, Q. Zhang, et X. Lin. Diffusion-based graph generative methods. 1 2024.

[30] Z. Chen, S. Villar, L. Chen, et J. Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *Advances in Neural Information Processing System (NeurIPS)*, pages 1–19, 2019.

[31] Z. Chen, Z. Lin, S. Chen, Y. Polyanskiy, et P. Rigollet. Residual connections provably mitigate oversmoothing in graph neural networks. 1 2025.

[32] F. Chung. Spectral graph theory. *ACM SIGACT News*, 30:1–214, 1999.

[33] R. Combes. An extension of mcdiarmid's inequality. *IEEE International Symposium on Information Theory - Proceedings*, pages 79–84, 2024.

[34] G. Corso, L. Cavalleri, D. Beaini, P. Liò, et P. Velickovic. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 2020-Decem, 2020.

[35] H. Crane. *Probabilistic Foundations of Statistical Network Analysis*. Chapman & Hall, 2018.

[36] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mahematics of Control, Signals, and Systems*, 2:303–314, 1989.

[37] S. Dasgupta. Learning mixtures of gaussians. In *IEEE 51st Annual Symposium on Foundations of Computer Science*, 1999.

[38] M. Defferrard, X. Bresson, et P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information and Processing Systems (NIPS)*, 2016.

[39] Y. Deshpande, A. Montanari, E. Mossel, et S. Sen. Contextual stochastic block models. *Advances in Neural Information Processing Systems*, 2018-Decem:8581–8593, 2018.

[40] T. Ding, D. Li, et R. Sun. Sub-optimal local minima exist for neural networks with almost all non-linear activations. pages 1–58, 2019.

[41] P. M. Djuric et C. Richard. *Cooperative and Graph Signal Processing*. 2018.

[42] N. T. Doncheva, J. H. Morris, J. Gorodkin, et L. J. Jensen. Cytoscape stringapp: Network analysis and visualization of proteomics data. *Journal of Proteome Research*, 18:623–632, 2 2019.

[43] S. Dutta et A. Bhattacharya. Rsvp: Beyond weisfeiler lehman graph isomorphism test. 9 2024.

[44] D. Duvenaud, D. Maclaurin, J. Aguilera-iparraguirre, G. Rafael, T. Hirzel, et R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information and Processing Systems (NeurIPS)*, pages 1–9, 2015.

[45] V. P. Dwivedi et X. Bresson. A generalization of transformer networks to graphs. 2020.

[46] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, et X. Bresson. Graph neural networks with learnable structural and positional representations. In *ICLR*, pages 1–25, 2022.

[47] B. Epping, A. René, M. Helias, et M. T. Schaub. Graph neural networks do not always oversmooth. pages 1–19, 2024.

[48] P. Erdös et A. Rényi. On random graphs i. *Publicationes Mathematicae*, 6:290–297, 1959.

[49] I. Evdaimon, G. Nikolentzos, C. Xypolopoulos, A. Kammoun, M. Chatzianastasis, H. Abdine, et M. Vazirgiannis. Neural graph generator: Feature-conditioned graph generation using latent diffusion models. 9 2024.

[50] F. D. V. Fallani, J. Richiardi, M. Chavez, et S. Achard. Graph analysis of functional brain networks: Practical issues in translational neuroscience. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369, 2014.

[51] L. Fesser et M. Weber. Mitigating over-smoothing and over-squashing using augmentations of forman-ricci curvature. 2023.

[52] M. Fey et J. E. Lenssen. Fast graph representation learning with pytorch geometric. *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[53] A. Fout, B. Shariat, J. Byrd, et A. Ben-Hur. Protein interface prediction using graph convolutional networks. In *Neural Information Processing Systems (NeurIPS)*, pages 6512–6521, 2017.

[54] V. Fung, J. Zhang, E. Juarez, et B. G. Sumpter. Benchmarking graph neural networks for materials chemistry. *npj Computational Materials*, 7, 12 2021.

[55] H. Gao et S. Ji. Graph u-nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:4948–4960, 2022.

[56] V. K. Garg, S. Jegelka, et T. Jaakkola. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning (ICML)*, pages 1–22, 2020.

[57] C. L. Giles, K. D. Bollacker, et S. Lawrence. Citeseer: an automatic citation indexing system. *Proceedings of the ACM International Conference on Digital Libraries*, pages 89–98, 1998.

[58] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, et G. E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, pages 1–14, 2017.

[59] X. Glorot, A. Bordes, et Y. Bengio. Deep sparse rectifier neural networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011.

[60] D. Grattarola, D. Zambon, F. M. Bianchi, et C. Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–11, 2022.

[61] W. L. Hamilton, R. Ying, et J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information and Processing Systems (NeurIPS)*, pages 1–19, 2017. inductive: predict when unseen nodes.

aggregation graph à la Jure.

[62] D. K. Hammond, P. Vandergheynst, et R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30:129–150, 2011.

[63] H. A. Helfgott, J. Bajpai, et D. Dona. Graph isomorphisms in quasi-polynomial time. *arXiv:1710.04574*, 2017.

[64] P. Hines, S. Blumsack, E. C. Sanchez, et C. Barrows. The topological and electrical structure of power grids. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2010.

[65] P. D. Hoff, A. E. Raftery, et M. S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098, 2002.

[66] P. W. Holland. Stochastic blockmodels: First steps. *Social Networks*, 5:109–137, 1983.

[67] A. Hollender et M. Zampetakis. The computational complexity of finding stationary points in non-convex optimization. *Mathematical Programming*, pages 1–55, 2024.

[68] K. Hornik, M. Stinchcombe, et H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[69] T. House. Modelling epidemics on networks. *Contemporary Physics*, 53:213–225, 2012.

[70] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, et J. Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Neural Information Processing Systems (NeurIPS)*, pages 1–34, 2020.

[71] D. R. Hunter, P. N. Krivitsky, et M. Schweinberger. Computational statistical methods for social network models. *Journal of Computational and Graphical Statistics*, 21:856–882, 2012.

[72] S. Ji, S. Pan, E. Cambria, P. Marttinen, et P. S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 4 2021.

[73] D. Jiang, Z. Wu, C. Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu, et T. Hou. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13:1–23, 2021.

[74] W. Jiang et J. Luo. Graph neural network for traffic forecasting: A survey. 2 2022.

[75] J. Jo, D. Kim, et S. J. Hwang. Graph generation with diffusion mixture. 2023.

[76] O. Kallenberg. *Probabilistic Symmetries and Invariance Principles.* 2005.

[77] T. N. Kipf et M. Welling. Semi-supervised learning with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[78] A. Kolmogorov et G. Castelnuovo. *Sur la notion de la moyenne.* G. Bardi, tip. della R. Accad. dei Lincei, 1930.

[79] N. Kormann, B. Doerr, et J. F. Lutzeyer. Position: Don't be afraid of over-smoothing and over-squashing. 2026.

[80] R. Kortvelesy, S. Morad, et A. Prorok. Generalised f-mean aggregation for graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[81] R. Kotoge, Z. Chen, T. Kimura, Y. Matsubara, T. Yanagisawa, H. Kishima, et Y. Sakurai. Evobrain: Dynamic multi-channel eeg graph modeling for time-evolving brain networks. In *Advances in Neural Information and Processing Systems (NeurIPS)*, 2025.

[82] Y. LeCun, Y. Bengio, et G. Hinton. Deep learning. *Nature*, 521:436–444, 5 2015.

[83] J. Lei et A. Rinaldo. Consistency of spectral clustering in stochastic block models. *Annals of Statistics*, 43:215–237, 2015.

[84] R. Levie, W. Huang, L. Bucci, M. Bronstein, et G. Kutyniok. Transferability of spectral graph convolutional neural networks. *Journal of Machine Learning Research*, 22:1–41, 2021.

[85] C. Li, H. Farkhoor, R. Liu, et J. Yosinski. Measuring the intrinsic dimension of objective landscapes. 2018.

[86] G. Li, M. Müller, B. Ghanem, et V. Koltun. Training graph neural networks with 1000 layers. *Proceedings of Machine Learning Research*, 139:6437–6449, 2021.

[87] J. Li, S. Lai, Z. Shuai, Y. Tan, Y. Jia, M. Yu, Z. Song, X. Peng, Z. Xu, Y. Ni, H. Qiu, J. Yang, Y. Liu, et Y. Lu. A comprehensive review of community detection in graphs. 2024.

[88] P. Li, Y. Wang, H. Wang, et J. Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 2020-Decem:1–29, 2020.

[89] D. Lim, J. Robinson, L. Zhao, T. Smidt, S. Sra, H. Maron, et S. Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *ICLR Workshop GTRL*, pages 1–42, 2022.

[90] A. Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representation (ICLR)*, 2020.

[91] A. Loukas et P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. *35th International Conference on Machine Learning, ICML 2018*, 7:5102–5118, 2018.

[92] L. Lovász. Large networks and graph limits. *Colloquium Publications*, 60:487, 2012.

[93] S. Luan, C. Hua, M. Xu, Q. Lu, J. Zhu, X.-W. Chang, J. Fu, J. Leskovec, et D. Precup. When do graph neural networks help with node classification? investigating the impact of homophily principle on node distinguishability. pages 1–33, 4 2023.

[94] U. V. Luxburg, M. Belkin, et O. Bousquet. Consistency of spectral clustering. *Annals of Statistics*, 36:555–586, 2008.

[95] S. Lv. Generalization bounds for graph convolutional neural networks via rademacher complexity. 2 2021.

[96] Y. Ma, X. Liu, N. Shah, et J. Tang. Is homophily a necessity for graph neural networks? *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.

[97] S. Mallat. Group invariant scattering. *Communications on Pure and Applied Mathematics*, 65: 1331–1398, 2012.

[98] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, et G. Stolovitzky. Revealing strengths and weaknesses of methods for gene network inference. *Proceedings of the National Academy of Sciences of the United States of America*, 107:6286–6291, 2010.

[99] H. Maron, H. Ben-Hamu, H. Serviansky, et Y. Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1–12, 2019.

[100] H. Maron, H. Ben-Hamu, N. Shamir, et Y. Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations (ICLR)*, pages 1–13, 2019.

[101] H. Maron, E. Fetaya, N. Segol, et Y. Lipman. On the universality of invariant networks. In *International Conference on Machine Learning (ICML)*, 2019.

[102] S. Maskey, R. Levie, et G. Kutyniok. Transferability of graph neural networks: an extended graphon approach. 2021.

[103] S. Maskey, R. Levie, Y. Lee, et G. Kutyniok. Generalization analysis of message passing neural networks on large random graphs. *Advances in Neural Information Processing Systems*, 35:1–63, 2022.

[104] S. Maskey, G. Kutyniok, et R. Levie. Generalization bounds for message passing networks on mixture of graphons. *arXiv preprint arXiv:2404.03473*, 2024.

[105] C. Mavromatis et G. Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. In *Findings of the Association for Computational Linguistics: ACL*, 2025.

[106] J. Menche, A. Sharma, M. Kitsak, S. D. Ghiassian, M. Vidal, J. Loscalzo, et A. L. Barabási. Uncovering disease-disease relationships through the incomplete interactome. *Science*, 347:841, 2 2015.

[107] P. Mernyei et C. Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. 1 2022.

[108] D. Mesquita, A. H. Souza, et Kaski. Rethinking pooling in graph neural networks. In *Advances in Neural Information Processing System*, 2020.

[109] M. Mohri, A. Rostamizadeh, et A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018.

[110] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, et M. Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 4602–4609, 2019.

[111] C. Morris, Y. Lipman, H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, et K. Borgwardt. Weisfeiler and leman go machine learning: The story so far. *arXiv:2112.09992*, pages 1–51, 2021.

[112] A. Y. Ng, M. I. Jordan, et Y. Weiss. On spectral clustering: Analysis and algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001.

[113] C. L. M. Nickel. *Random Dot Product Graphs a Model for Social Networks*. PhD thesis, 2006.

[114] L. O'Bray, M. Horn, B. Rieck, et K. Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. 3 2022.

[115] K. Oono et T. Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representation (ICLR)*, 2020.

[116] P. Orbanz et D. M. Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37:437–461, 2015.

[117] M. Park et D. Kim. Taming gradient oversmoothing and expansion in graph neural networks. pages 1–19, 2024.

[118] R. Pastor-Satorras et A. Vespignani. *Evolution and Structure of the Internet*. Cambridge University Press, 2010.

[119] H. Pei, Y. Li, H. Deng, J. Hai, P. Wang, J. Ma, J. Tao, Y. Xiong, et X. Guan. Multi-track message passing: Tackling oversmoothing and oversquashing in graph learning via preventing heterophily mixing. Technical report, 2024.

[120] L. Pellis, F. Ball, S. Bansal, K. Eames, T. House, V. Isham, et P. Trapman. Eight challenges for network epidemic models. *Epidemics*, 10:58–62, 2015.

[121] M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2008.

[122] L. L. Peterson et B. S. Davie. *Computer networks a systems approach*. Morgan Kaufmann, 2023.

[123] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numerica*, 8:143–195, 1999.

[124] A. Pirayre, C. Couprie, L. Duval, et J. C. Pesquet. Brane clust: Cluster-assisted gene regulatory network inference refinement. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 15:850–860, 2018.

[125] O. Platonov, D. Kuznedelev, A. Babenko, et L. Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. 2022.

[126] L. Rauchwerger, S. Jegelka, et R. Levie. Generalization, expressivity, and universality of graph neural networks on attributed graphs. pages 1–54, 2024.

[127] P. Rigollet. Generalization error bounds in semi-supervised classification under the cluster assumption. *Journal of Machine Learning Research*, 8:1369–1392, 2007.

[128] Y. Rong, W. Huang, T. Xu, et J. Huang. Dropedge: Towards deep graph convolutional networks on node classification. pages 1–18, 2019.

[129] L. Rosasco, M. Belkin, et E. D. Vito. On learning with integral operators. *Journal of Machine Learning Research*, 11:905–934, 2010.

[130] L. Ruiz, L. F. Chamon, et A. Ribeiro. Graphon neural networks and the transferability of graph neural networks. In *Advances in Neural Information and Processing Systems (NeurIPS)*, 2020.

[131] T. K. Rusch, M. M. Bronstein, et S. Mishra. A survey on oversmoothing in graph neural networks. 2023.

[132] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, et P. Battaglia. Graph networks as learnable physics engines for inference and control. *arxiv:1806.01242*, 2018.

[133] A. Sannai, M. Imaizumi, et M. Kawano. Improved generalization bounds of group invariant / equivariant deep networks via quotient feature spaces. *Proceedings of Machine Learning Research*, 161:771–780, 2021.

[134] R. Sato, M. Yamada, et H. Kashima. Random features strengthen graph neural networks. *SIAM International Conference on Data Mining, SDM 2021*, pages 333–341, 2021.

[135] A. Saxena, G. Fletcher, et M. Pechenizkiy. Nodesim: node similarity based network embedding for diverse link prediction. *EPJ Data Science*, 11, 2022.

[136] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, et G. Monfardini. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20:81–102, 2009.

[137] F. Scarselli, M. Gori, A. C. Tsoi, G. Monfardini, M. Hagenbuchner, et G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61–80, 2009.

[138] B. Schwikowski, P. Uetz, et S. Fields. A network of protein–protein interactions in yeast. *Nature Biotechnology*, 18:1257–1261, 2000.

[139] W. R. Scott et G. F. Davis. *Organizations and organizing : rational, natural and open systems perspectives*. Pearson Prentice Hall, 2016.

[140] S. Shalev-Shwartz et S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2013.

[141] O. Shamir. Can we find near-approximately-stationary points of nonsmooth nonconvex functions? 2 2020.

[142] N. Shervashidze, S. V. N. Vishwanathan, T. H. Petri, K. Mehlhorn, K. Borgwardt, et T. H. Petri. Efficient graphlet kernels for large graph comparison. *International conference on artificial intelligence and statistics*, 5:488–495, 2009.

[143] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, et P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30:83–98, 2013.

[144] M. Tang, A. Athreya, D. L. Sussman, V. Lyzinski, et C. E. Priebe. A nonparametric two-sample hypothesis testing problem for random dot product graphs. pages 1–44, 2014.

[145] J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, et M. M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *ICLR 2022 - 10th International Conference on Learning Representations*, 2021.

[146] N. G. Trillos, M. Gerlach, M. Hein, et D. Slepčev. Error estimates for spectral convergence of the graph laplacian on random geometric graphs toward the laplace–beltrami operator. *Foundations of Computational Mathematics*, 2019.

[147] J. E. van Engelen et H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109: 373–440, 2020.

[148] A. Vasileiou, S. Jegelka, R. Levie, et C. Morris. Survey on generalization theory for graph neural networks. pages 1–51, 2025.

[149] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, et I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing System (NeurIPS)*, 2017.

[150] V. Veitch et D. M. Roy. The class of random graphs arising from exchangeable random measures. 2015.

[151] P. Veličković, A. Casanova, P. Liò, G. Cucurull, A. Romero, et Y. Bengio. Graph attention networks. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–12, 2018.

[152] S. Vempala et G. Wang. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68:841–860, 2004.

[153] L. Venturi, A. S. Bandeira, et J. Bruna. Spurious valleys in one-hidden-layer neural network optimization landscapes. *Journal of Machine Learning Research*, 20:1–33, 2019.

[154] C. Vignac, A. Loukas, et P. Frossard. Building powerful and equivariant graph neural networks with message-passing. In *Advances in Neural Information and Processing Systems (NeurIPS)*, 2020.

[155] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, et P. Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representation (ICLR)*, 5 2023.

[156] H. Wang, J. Leskovec, F. Zhang, M. Zhao, W. Li, M. Zhang, et Z. Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 968–977, 2019.

[157] B. Y. Weisfeiler et A. A. Leman. The reduction of a graph to canonical form and the algebra which appears therein. *Nti*, 2:12–16, 1968.

[158] J. Wu, L. Ma, X. Chen, et T. Broyd. Research on interpretable graph neural network agent model for siltation diagnosis in urban-scale sewer systems. *Tunnelling and Underground Space Technology*, 162:106666, 8 2025.

[159] X. Wu, A. Ajorlou, Z. Wu, et A. Jadbabaie. Demystifying oversmoothing in attention-based graph neural networks. *Advances in Neural Information Processing Systems*, 36:1–24, 2023.

[160] B. Xu, H. Shen, Q. Cao, Y. Qiu, et X. Cheng. Graph wavelet neural network. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–13, 2019.

[161] K. Xu, W. Hu, J. Leskovec, et S. Jegelka. How powerful are graph neural networks? In *ICLR*, pages 1–15, 2019. aggregative GNN à la Jure are less powerful than WL.

[162] C. Yang, R. Wang, S. Yao, S. Liu, et T. Abdelzaher. Revisiting over-smoothing in deep gcns. 2020.

[163] Z. Yang, W. W. Cohen, et R. Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *33rd International Conference on Machine Learning, ICML 2016*, 1:86–94, 2016.

[164] D. Yarotsky. Universal approximations of invariant maps by neural networks. *ArXiv: 1804.10306*, pages 1–64, 2018.

[165] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, et J. Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing (NeurIPS)*, 2019.

[166] J. You, R. Ying, et J. Leskovec. Position-aware graph neural networks. 2019.

[167] Y. Yu, T. Wang, et R. J. Samworth. A useful variant of the davis-kahan theorem for statisticians. *Biometrika*, 102:315–323, 2015.

[168] C. Yun, S. Sra, et A. Jadbabaie. Small nonlinearities in activation functions create bad local minima in neural networks. *7th International Conference on Learning Representations, ICLR 2019*, pages 1–33, 2019.

[169] M. Zaheer, S. Kottur, S. Ravanbhakhsh, B. Póczos, R. Salakhutdinov, et A. J. Smola. Deep sets. *Advances in Neural Information Processing Systems*, 2017-Decem:3392–3402, 2017. permutation invariant function.

[170] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, et V. Prasanna. Graphsaint: Graph sampling based inductive learning method. 2019.

[171] B. Zhang, C. Fan, S. Liu, K. Huang, X. Zhao, J. Huang, et Z. Liu. The expressive power of graph neural networks: A survey. pages 1–20, 2023.

[172] R.-R. Zhang et M.-R. Amini. Generalization bounds for learning under graph-dependence: A survey. 3 2024.

[173] T. Zhang. *Mathematical Analysis of Machine Learning Algorithms*. Cambridge University Press, 2020.

[174] G. Zhao, T. Wang, C. Lang, Y. Jin, Y. Li, et H. Ling. Dfa-gnn: Forward learning of graph neural networks by direct feedback alignment. 6 2024.

[175] L. Zhao et L. Akoglu. Pairnorm: Tackling oversmoothing in gnns. *8th International Conference on Learning Representations, ICLR 2020*, pages 1–17, 2020.

[176] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, et P. S. Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv:2202.07082*, 2022.

[177] Y. Zheng, S. Luan, et L. Chen. What is missing in homophily? disentangling graph homophily for graph neural networks. 2024.

[178] C. Zhou, X. Wang, et M. Zhang. Unifying generation and prediction on graphs with latent graph diffusion. Technical report.

[179] Y. Zhou et Y. Liang. Critical points of linear neural networks: Analytical forms and landscape properties. *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pages 1–18, 2018.

[180] J. Zhu, R. A. Rossi, A. Rao, T. Mai, N. Lipka, N. K. Ahmed, et D. Koutra. Graph neural networks with heterophily. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 12B:11168–11176, 2021.

[181] M. Zitnik et J. Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33:i190–i198, 2017.

[182] Álvaro Arroyo, A. Gravina, B. Gutteridge, F. Barbero, C. Gallicchio, X. Dong, M. Bronstein, et P. Vandergheynst. On vanishing gradients, over-smoothing, and over-squashing in gnns: Bridging recurrent and graph learning. 2 2025.

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Apprentissage sur graphes et réseaux de neurones sur graphes appliqués aux grands graphes (aléatoires)

Mot clés : apprentissage sur graphes, réseaux de neurones sur graphes, graphes aléatoires

Résumé : Dans cette *Habilitation à diriger des recherches*, je présente certains de mes travaux en apprentissage sur graphes de ces dernières années, mêlant graphes aléatoires et réseaux de neurones sur graphes (GNN). Je m'intéresse plus particulièrement à la prédiction sur *nœuds* de graphes, qui brise le paradigme d'indépendence des données en apprentissage. Je développe dans ce manuscrit un cadre général pour l'analyse de l'erreur de généralisation de telles méthodes, inspiré de l'apprentissage classique. Combiné à des modèles de graphes à position latentes, je mets à jour des liens fort entre apprentissage sur graphes et apprentissage classique, et donne des méthodes génériques pour borner l'erreur de généralisation. J'applique ensuite ce nouveaux cadre aux GNNs, en exploitant de nombreux résultats issus de mes travaux de ces dernières années rassemblés pour la première fois dans un tout cohérent. Un chapitre "bonus" sur le surlissage des GNNs vient compléter ce manuscrit.

Title: Graph Machine Learning and Graph Neural Networks on Large (random) Graphs

Keywords: graph machine learning, graph neural networks, random graphs

Abstract: In this *Habilitation à diriger des recherches*, I present a selection of my works of the last few years on Graph Machine Learning (graph ML) and Graph Neural Networks (GNNs), combined with large random graphs. I particularly focus on *node-level* prediction tasks, which does not respect the classical i.i.d. assumption in Machine Learning. In this manuscript, I develop a general framework to analyze such prediction methods, inspired by classical Machine Learning. Combined with Latent Position Models for random graphs, I uncover strong links between graph ML and regular ML, as well as methodologies to bound the generalization error. I then apply this framework to GNNs, gathering many of my previous results into a coherent whole for the first time. A bonus chapter on GNN *oversmoothing* complete this manuscript.