THE UNIVERSITY OF CHICAGO

MULTIRESOLUTION ANALYSIS ON DISCRETE SPACES

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF STATISTICS

BY
JONATHAN ESKREIS-WINKLER

CHICAGO, ILLINOIS
MARCH 2020

"From all my teachers I have learned, as Your laws are my rumination."

Psalms 119:99

Table of Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

The work contained herein is the product of a long, and at times arduous, journey. There have been many guides along this trip and any of the positive elements in my thesis are certainly due to the supportive team that has helped me get where I am today. If I were actually giving sufficient credit where it is due, the acknowledgements section of this thesis would be longer than the rest of it. So what follows is an extreme abbreviation of the gratitude that lives behind these pages.

I entered the statistics PhD program at the University of Chicago without a clear sense of direction. The research world of statistical machine learning was extremely engaging and I was eager to join it. Through my exposure to foundational coursework and new research I have been able to clarify my interests and goals as I take my next step forward.

It is hard to believe that before September 2014, I had never understood the meaning of "least squares regression," let alone encountered computational linear algebra. The first year of PhD coursework has proven to be a deep source of information to mine as I continue to explore new areas in machine learning research and applied statistics. Generalized linear models, computational linear algebra, and machine learning, taught by Drs. Mihai Anitescu, Dan Nicolae, Lek-heng Lim, and Risi Kondor have been the seeds that my subsequent research has grown from. I know that this knowledge base will be a beacon for my future data explorations.

Navigating my way through the PhD program would have been extremely difficult without the guidance of Dr. Steven Lalley. His intellectual encouragement helped me clarify research interests that would be worth pursuing, and others that would be less fulfilling. The contribution of the department's office staff was so constant that it has been too easy to take for granted. The efforts of Matt Johnson, Laura Rigazzi, Jonathan Rodriguez, Keisha Prowoznik, Kirsten Wellman, Kathryn Kraynik, Edward Friedman, and John Zekos have been an ever-present source of support. Their collective efforts make it possible for PhD students to focus on courses, teaching, and research with all the resources needed to be

successful. I am also very grateful to Drs. Mihai Anitescu and Chao Gao for joining my dissertation committee. The time and generosity they have extended towards me have made completing my dissertation much easier.

I am extremely lucky to have found a research advisor as patient, kind, and thoughtful as Dr. Risi Kondor. Immediately after taking his machine learning course I approached him and asked how to partake in the fascinating world he had taught us about. His original work in computational multiresolution analysis has provided me with a rich source of ideas and directions. Though our plans have not always developed to fruition, he has maintained a level of attention and interest in my work that has given me the confidence to keep moving forward even when the going got tough. His guidance has pushed me either to move faster or slow down as needed, and he has always directed me to define reasonable expectations and measurable goals when my work felt overwhelming or aimless.

One thing I am sure of is that I would not have been able to grow as a statistician and researcher without the support of my friends. In Philadelphia, Lakeview, Hyde Park, and elsewhere, the connections I have formed in college, graduate school, and beyond have been a source of comfort and encouragement when I needed it most. In the last five years, whenever I have fallen down (or apart), friends have always been a conversation or phone call away and I aspire to give them a fraction of the gifts they have given me. Specifically, the friendship of Chris McKennan, Matt Bonakdarpour, Pramod Mudrakarta, Zvi Rosen, and Adam Saltzman has kept me sane when there were plenty of reasons not to be.

Graduate school has been a tumultuous, arduous period for me, and knowing I always had an amazing family to rely on made my time so much easier. My Arbit family - David, Sheara, Josh, Rachel, and Orly - are a constantly encouraging, always fun layer of support for me when school has been tough and I needed a break (or amazing vacation). Rabbi and Mrs. Levene have been and continue to be the most reliably present source of comfort and calm in my life for as long as I can remember. It is truly incredible how taking a single step into their house of laughter, love, and poetry makes my problems and stresses melt away.

Incredible beyond measure. My sisters Sarah and Lauren know me better than anybody and make me feel like I can do anything. I am blessed to live life with two such amazing role models for honesty, loyalty, and refinement. I live every day trying to emulate them and I appreciate that they show me the way. Their families - Ezi, Bevy, Naftali, Ari, Bevy, and Eve - do nothing but bring me happiness and positivity and I am excited to spend more time with them on the East Coast! I have always known I can rely on my father, but going through graduate school has shown me an entirely new side of his unconditional love and support. He has helped me navigate academic decisions, stress, and even showed up in person when I needed help. I thank my dad and Michele for their understanding of academic pressures, and their advice and encouragement along the way.

Throughout graduate school, and my entire life, the voice in my head that tells me to keep going, to take care of myself, to be kind, to be creative, to have fun, has belonged to my mom, Beverly Dinah Eskreis, of blessed memory. She is with me when I make hard decisions, when I struggle, and when I succeed. Her intellectual curiosity and enthusiasm inspires me to reach new vistas, and her depth of values reminds me to always improve myself. She has taught me to live a life of music and poetry, and I hope that my life will be, to some extent, a realization of that beauty.

Lastly, I wish to acknowledge the immeasurable amount of strength and love I receive from my wonderful wife, Tali. She is unique in her constant desire to take care of those around her. Whether they be plants or humans, her top priority is always to think of others and to give them what they need. Having been married now for over five years, her presence in my life has made me a better, more compassionate, more patient, and (perhaps most importantly) better-rested person. The future that awaits us is bright and I am lucky to hold her hand as we navigate life's adventures together.

Blessed are You, Lord our G-d, King of the universe, who has given us life, sustained us, and enabled us to reach this occasion.

ABSTRACT

It continues to be much cheaper to store data than to analyze it. This state of data analysis motivates methods that make minimal assumptions in order to reduce the complexity of data. In order to address scalability in certain applied, network-based problems, we bring together three distinct fields: multiresolution analysis (MRA), hyperbolic embeddings, and community detection. Together, the work in these fields provides a path forward for certain difficult problems in large-scale data analysis. We provide a theoretical background, a description of our contributions, and a number of applications to show the viability of our ideas in a real-world setting.

The primary contributions of this dissertation are threefold. First, we draw a connection between a class of currently-used methods in computational MRA to hyperbolic representations of data. This connection allows us to suggest a rationale for when different MRA methods are appropriate, and to define related tree-based kernels for wavelet construction.

Second, we broaden the existing work describing the mechanics of how the multiresolution matrix factorization (MMF) summarizes data. We look at this MRA framework from two perspectives. First, we consider how MMF may be viewed as a hyperbolic embedding. Second, we consider the way MMF may be seen as a regularization operator. We show that for certain graphs, the regularization imposed by MMF enables methods to perform more accurate inference than when no regularization is used. This work fits within the existing field of high dimensional statistics where regularization may turn a massive problem into a manageable one.

Third, we show the way that MMF, and other similar multiresolution methods, may be used efficiently in unsupervised and semisupervised settings. By taking advantage of the localization of wavelets in time and frequency space of graphs, we are able to detect complex, multiscale, overlapping community structures, as well as combine network structure with the small amounts of labeling for semisupervised learning.

CHAPTER 1

INTRODUCTION

1.1 Case Study: Wikipedia

We live in an age of information overload. To take a ubiquitous example, *Wikipedia: The Free Encyclopedia* currently hosts nearly 6 million English articles (48 million articles in total), with a growth rate of approximately 20 thousand per month.[2] There is a fun game, called "The Wiki Game," hosted at thewikigame.com, which makes a competitive sport out of the immenseness of Wikipedia.[1] To play, one is given two articles as starting and ending points, and one is tasked with clicking the fewest possible number of links in order to define a path of pages that begin at the start point and finish at the end point. The web of link connections between Wikipedia articles is massive and complicated, and a huge number of paths exist between any two pages (even when those two pages that are themselves link-connected).

Formally, Wikipedia may be interpreted as a graph whose nodes are articles and whose directed edges are the presence of hyperlinks. It has been identified that for a given Wikipedia article, the links at the top of article connect to more general topics, whereas links within the lower portions of the page refer to more specific topics. This small difference in link placement has a big impact on how users navigate Wikipedia and also indicates a latent hierarchical structure that is present in the Wikipedia network. In general, the shortest path between article A and article B (unless A and B are sufficiently close relatives) is to climb upward in generality until general topic C is reached, relating to both A and B, and then to descend from the generality of C to the specificity of B. Using data collected from The Wiki Game, researchers showed that this is in fact the typical way that players approach the game.[89] The up-then-down strategy may be modeled using spokes radiating from a wheel's axis. It is possible to travel from the end of one spoke to another in two ways: by traveling along the circumference of the wheel or by traveling to the hub of the spokes and then departing for the end point. If two spokes are sufficiently close together, it is wasteful

to travel to the hub, when a short stroll along the circumference will do. If two spokes are far apart, however, traveling through the hub can be a much appreciated shortcut.

Contemporary research in network routing relies heavily on the tradeoffs that arise when navigating complex networks, and may be seen as analogous to the foundational tradeoff between breadth-first search and depth-first search. In breadth-first search, all the neighbors of a starting node are weighed equally for further investigation. If the end point is nearby, this will be extremely efficient. If the end point is far away, however, the number of nodes that must be visited before completion will be on the order of the total number of graph nodes. In depth-first search, a path between two far-apart nodes may be identified quickly, provided there is some strategy for which directions make the most sense. Depth-first searches are unforgiving of small initial miscalculations, so connecting close-together nodes with a depth-first search path runs the dangerous risk of extreme overshooting.

We sum up our suggested strategy for The Wiki Game by saying that for a graph with an implicit hierarchy (such as that defined by levels of generality), two nodes may be connected efficiently by finding a "common ancestor" in the graph. Even though this strategy may be wasteful sometimes, exaggerating the distance between nearby nodes, it provides a robust, foundational approach for navigating complex networks.

The common ancestor strategy has another layer of utility. The strategy works at any scale and may be run recursively. For example, just as the quantitative subjects taught to a primary students may be organized in a hierarchy, the subtopics within any one of those subjects may also be organized in a hierarchy, and so on in both directions (primary school topics is a single spoke of a larger category, such as educational topics). This illustrates the existence of two salient features of many real-world graphs (of which Wikipedia is only one example): hierarchical structure and multiscale structure. By the former, we refer to the existence of a general hub-and-spokes structure; by the latter, we refer to the fact that each spoke is itself the hub of another set of spokes.

I have played The Wiki Game and it is not easy. The difficulty arises because there is no

single authoritative hierarchy that governs the database of Wikipedia articles. Given a realistic (and indeed real) example of a start point "Poseidon" and an ending point "Gregorian chant," there are several paths that would contain both articles, two of which are:

Poseidon → Ancient Greek Religion → Ancient Roman Religion →

Early Christianity → Catholic Church → Gregorian Chant

Poseidon → Twelve Olympians → Ancient Greece →

Ancient Greek → neume → Gregorian Chant

Whereas the first path considers a hierarchy defined by religious orders and the religious role of Gregorian chants, the second hierarchy defines the Gregorian chant in terms of its notes, which are described using Greek-rooted words. Even while similarities exist between the two hierarchies, they are assuredly different. The dual hierarchies pose a challenge in the field of machine learning due to the difficulty, one can imagine, a computer would have answering, "How similar are Poseidon and Gregorian chants?" or "Which articles are most similar to Poseidon *and* Gregorian Chants?" The primary aim of this thesis is to provide a framework for answering expansive questions of this type.

1.2 Our Approach

Machine learning (ML) methods contribute to graph-based analysis when the number of nodes, the number of connections, or the combination of both makes it intractable for a human being to use traditional brute-force methods. This thesis primarily relies on three areas of ML-related research in order to describe structures present in real-world graphs. These areas are multiresolution analysis (MRA), graph embeddings, and network analysis.

Chapter 2 provides the foundation for our research in MRA, starting with continuous

Fourier transforms and ending with examples of wavelet transforms of discrete objects. MRA is a well-developed topic in signal processing, but its application in discrete domains, such as graphs and matrices, is newer, with new MRA frameworks and their applications emerging in the last decade as a popular approach for describing graphs. The connection between Fourier analysis and MRA on discrete objects is critical for understanding our work in chapter 5, where we consider seeing MRA on graphs through the lenses of hyperbolic space and regularization theory.

Chapter 3 collects information on the many ways that trees are used to model observed graphs. The tree's simple structure[1] makes it an especially convenient way of transforming a complicated graph into a simpler graph on which it is easier to do inference. Tree-based approaches vary from simply exchanging a graph for one of its spanning tree subgraphs, to using collections of spanning tree subgraphs, to using subgraphs that are almost spanning trees, to embedding graphs in hyperbolic space. Mapping any graph to a subgraph (which contains all its nodes) is a form of metric embedding. We consider simple tree embeddings, along with the more abstract hyperbolic embedding. Hyperbolic embeddings are metric embeddings that map graphs to hyperbolic metric spaces, metric spaces with certain interesting non-Euclidean properties. "Tree-likeness" is a term often used to describe hyperbolic spaces, with Gromov's δ-hyperbolicity parameter being used to define the tree-likeness of a metric space. When $\delta = 0$ a space is simply a tree, whereas for $\delta = \infty$, the space is Euclidean. For $\delta > \mathbb{R}_+$, a space is defined with certain properties that have led to many contributions in hyperbolic embeddings of hierarchical graphs.[110, 131] Empirical observations of the hyperbolicity of real-world graphs have led many researchers to posit that hyperbolic space is the natural embedding space for massive real-world graphs such as the world wide web, as oppose to Euclidean embeddings that lead to higher levels of distortion.[30] We review existing research on making tree or tree-likeness assumptions about graphs, setting the groundwork

1. In the context of this thesis, the most important characteristic making trees "simple" is the fact that, between any two nodes, there exists a path which is a subpath of any other path connecting those nodes.

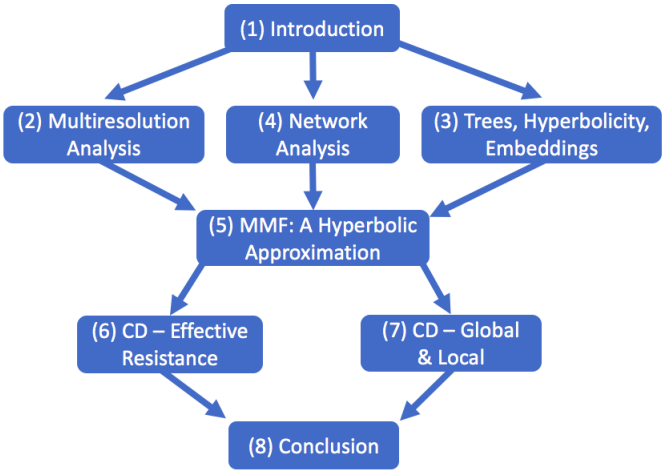


Figure 1.1: Graphical representation of thesis outline

for the analysis of MRA from the perspective of hyperbolic embeddings in chapter 5. Our discussion focuses primarily on the intersection between representing graphs using collections of trees and hyperbolic space.

Chapter 4 provides a background on areas in network analysis whose methods are most relevant for our research. Specifically, graph-based semi-supervised learning which is used in chapter 5, community detection which is the basis of chapters 6 and 7, and statistical testing for graphs, which is an area of future direction and discussed in chapter 9. Many methods in network analysis are limited in scalability, leading to the introduction of preprocessing approximation steps before inference.

Having provided a broad basis for the usage of MRA, graph metric embeddings, and network analysis, we turn to specific ways they may be used for developing new ideas in their respective fields. Chapter 5 does not specifically deal with network applications, its primary focus is the connection between a specific MRA framework - the multiresolution matrix factorization (MMF) - and hyperbolic embeddings. We point out how the use of MRA leads to a hyperbolic embedding and how a hyperbolic embedding can improve wavelet construction. We also provide an interpretation of MMF as a hyperbolic regularization tool.

In chapter 6 we use the resistance network model to develop a tool for a specific brand of community detection called *seed set expansion*. We demonstrate that the usage of a fast

preprocessing step to *germinate* a seed set (that is, to expand and improve the quality of a seed set) greatly improves the accuracy of subsequently applied seed set expansion methods. This chapter does not explicitly make use of MRA, but by using sets of spanning trees to define effective resistance on graphs, we exploit the hyperbolicity present in graphs, similar to existing work, such as that described in section 3.3.3.

Chapter 7 is the most unifying chapter of the thesis, where we use MMF to do community detection (CD) on large graphs. Though MMF is assuredly not the only approach for MRA on discrete spaces, we propose that MMF's method for learning graph wavelets, uniquely suits it for the task of CD. As a benchmark for comparison, we consider MMF's performance relative to a similar CD technique[133] that uses of spectral graph wavelets.[71] We then delve deeper into the interpretation of the communities found by MMF by considering a novel application: the US stock market using historic stock prices. In this context, networks are convenient models for understanding pairwise relationships, and communities defined on said networks allow for descriptive and predictive analysis.

We conclude in chapter 9, summarizing the findings presented up until that point, along with future directions that we are eager to follow up with. To visualize the structure of this thesis and the connections that exist between the subsequent chapters, please refer to the graph in figure 1.1.

CHAPTER 2

FROM FOURIER ANALYSIS IN STRUCTURED SPACES TO MULTIRESOLUTION ANALYSIS IN UNSTRUCTURED SPACES

To cope with too much information we need to practice the Art of Decimation - which means selecting what is most worthwhile and relevant. But, to do that well takes experience and a certain intuition that builds up over time.

Umberto Eco, "Against the Loss of Memory" (2013)

The ideas presented in this chapter describe general techniques for multiresolution analysis (MRA) in $\mathbb{R}^n$ leading up to the application of MRA on discrete spaces, such as graphs and matrices. In order to contextualize the subsequent computational work contained in this dissertation, it is necessary to discuss its motivation from the perspective of Fourier analysis. While computational implementations will often involve discretizations or approximations of mathematical concepts, the basis for the computational procedures we discuss is assuredly the theory of Fourier analysis and MRA.

2.1 Fourier Analysis Background

Behind MRA is a longer history of analyzing signals using Fourier analysis. The *Fourier series* is the central tool of the discrete-time Fourier transform (DFT) for representing N signal measurements as a linear combination of N sinusoidal basis functions. The set of basis functions are:

$$\mathcal{B}^N_{\text{Fourier}} = \{e^{\frac{-2\pi ik}{N}} | 1 \leq k \leq N\}$$

This set of functions is an orthogonal basis for a function space, where the inner product between two elements $f, g \in \text{span}(\mathcal{B}^N_{\text{Fourier}})$ is defined as $\langle f, g \rangle = \sum_{i=1}^N f_i g_i$. As orthogonal basis elements, $\forall v \in \text{span}(\mathcal{B}^N_{\text{Fourier}})$, $\exists \alpha_1, \ldots, \alpha_N \in \mathbb{R}$ such that $v = \sum_{i=1}^N \alpha_i e^{\frac{-2\pi ik}{N}}$. The

DFT provides a procedure for transforming any N signal measurements into N coefficients $(\alpha_i)_{1\leq i\leq N}$. On the one hand, this may seem like N elements in one vector space have simply been exchanged from N elements in a different vector space. However, the Fourier coefficients carry the potential to do many things other than simply decomposing and then reconstructing signals. For example, Fourier decompositions permit quick computation for a number of challenging tasks: the convolution of signals is equivalent to the multiplication of Fourier coefficients, signal smoothing is equivalent to thresholding which frequency-associated basis elements will be used, signal comparison may be equivalent to the comparison of Fourier coefficients.

Therefore, the approach to the decomposition of functions using a basis of orthogonal functions (specifically, functions that are localized in frequency) has a wide applicability in signal analysis, including signal denoising, generating spectrograms of audio data, or image processing. A signal denoising process, for instance, involves collecting the coefficients of the Fourier series and only using the lowest frequency components in the signal reconstruction. This *low-pass filter* will eliminate noise from a signal if it is the case that the signal is a smooth signal with corruptions in limited regions of the time domain.

The Fourier series is able to represent signals with a finite number of observations (using finite terms), square integrable functions in a fixed interval (using countable terms), or periodic signals without interval boundaries (using countable terms). To represent a broader class of functions on the real line, without periodicity conditions, the general Fourier transform (FT), or continuous Fourier transform, decomposes a function into an uncountable sum of sinusoidal basis functions. Specifically, the Fourier coefficients for a function f are defined in equation 2.1 as the inner product between f and the sinusoidal terms, just as they were in the discrete case, where for $\eta \in \mathbb{R}$:

$$\hat{f}(\eta) = \int_{\mathbb{R}} f(x) e^{\overline{-2\pi i x \eta}} \mathrm{d}x \tag{2.1}$$

The inverse transform for any $x \in \mathbb{R}$ is defined similar to the DFT where

$$f(x) = \int_{\mathbb{R}} \hat{f}(\eta) e^{\frac{2\pi i x \eta}{}} \mathrm{d}\eta$$

The work in this dissertation, however, deals with discrete spaces, primarily graphs, and thus deals with discrete signal transforms.

2.1.1 Large Scale Implementation of Fourier

The Fourier transform and Fourier series have provided researchers in the past few centuries with invaluable tools for analyzing complicated phenomena, such as heat, geophysics, and astronomy.[1] In modern applications, such as image processing, a high priority is the number of computations involved in a data analysis procedure. In a typical DFT where the input vector of measurements is $f = (f_0, f_1, \ldots, f_{N-1}) \in \mathbb{R}^N$, the N Fourier coefficients are

$$\left\{ \hat{f}_k = \sum_{j=0}^{N-1} f_k e^{\frac{-2\pi i j k}{N}} \mid 0 \leq k \leq N-1 \right\}$$

1. As an aside, the monumental debate between astronomers on whether the solar system is modeled by geocentrism or heliocentrism has its roots in Fourier analysis. Astronomers holding a geocentric view had to reckon with the reality that planets, as observed from Earth, sometimes (appear to) deviate from their orbital path to engage in (what appears to be) a small circle tangent to the larger orbit. Instead of changing the basic foundations of their beliefs, some astronomers chose to add new parameters to their astronomical model, introducing epicycles. Over centuries, even as additional evidence of heliocentrism was collected, many held firm to the belief that planets' paths were orbits around the Earth with some small, predictable sub-cycles. With Fourier decompositions in our toolbox, we can dissect this debate more closely - it is possible to approximate any path, especially periodic orbits, using Fourier basis functions. The heliocentric model of the solar system is more systematic, in that it is simpler and is more generalizable, but by using Fourier decompositions of planets' paths, geocentrism can fit observational data as well as heliocentrism. Ultimately, deciding the winner of this debate comes down to a preference in the form of scientific theories - low or high parameter - rather than which is falsified or vindicated by the data.

From a different perspective, the computation of Fourier coefficients may be considered the product of a matrix and a vector:

$$\begin{pmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \vdots \\ \hat{f}_{N-1} \end{pmatrix} = \begin{pmatrix} e^{\frac{-2\pi i(0)(0)}{N}} & e^{\frac{-2\pi i(0)(1)}{N}} & \cdots & e^{\frac{-2\pi i(0)(N-1)}{N}} \\ e^{\frac{-2\pi i(1)(0)}{N}} & e^{\frac{-2\pi i(1)(1)}{N}} & \cdots & e^{\frac{-2\pi i(1)(N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ e^{\frac{-2\pi i(N-1)(0)}{N}} & e^{\frac{-2\pi i(N-1)(1)}{N}} & \cdots & e^{\frac{-2\pi i(N-1)(N-1)}{N}} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{-2\pi i}{N}} & \cdots & e^{\frac{-2\pi i(N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{-2\pi i(N-1)}{N}} & \cdots & e^{\frac{-2\pi i(N-1)(N-1)}{N}} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{N-1} \end{pmatrix}$$

It is evident that this complete computation of the Fourier transform requires $\mathcal{O}(N^2)$ operations, one for each entry of the matrix. The inverse transform exchanges $\hat{f}$ for f and $e^{\frac{2\pi ijk}{N}}$ for $e^{\frac{-2\pi ijk}{N}}$. This reverse operation thus also requires at least $\mathcal{O}(N^2)$ operations. Polynomial scaling is prohibitive in the world of "big data" and makes the contribution of a fast DFT (FFT) of utmost importance. Different varieties of clever algorithms - the decimation in time or frequency[34] - provide a means of computing the Fourier Transform in $\mathcal{O}(Nlog(N))$ steps. The computational FFT is an example of applying old techniques to new problems, specifically problems where the amount of data is very large. From one perspective, the FFT solves an important problem – the infeasibility of computing large scale DFTs. After all, any method, whether Fourier or not, must be able to produce results in sub-polynomial time in order to be practical for large scale implementations. From another perspective, though computationally feasible, the premise of Fourier analysis itself leaves something to be desired.

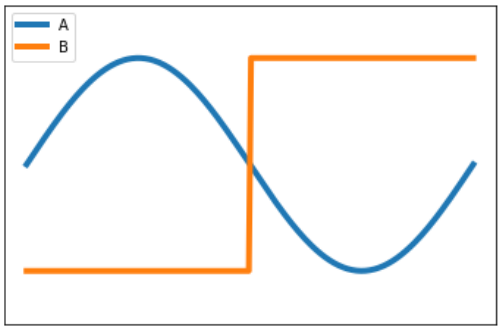


Figure 2.1: Example of two types of functions: a wave (A) and a box function (B). They both look like simple functions, but depending on the function basis used to represent them, they are quite different.

2.1.2 Limitations of Fourier

It is apparent that the use of a Fourier transform will capture and store valuable signal information, but it also suffers from limitations, specifically in its description of local behavior. Consider the difference between the two functions in Figure 2.1. In both cases the signal being transmitted is visually easy to process and interpret. One's eyes see signal A as a single wave of a given frequency and amplitude and signal B as a single box function of a given interval and magnitude. While visually these signals are both simple (and one might even say that B appears even simpler than A), using the Fourier basis it is much more difficult to represent B, a signal with large discontinuities and finite support, because it is not known a priori what that support will be and where those discontinuities will be. This is a very relevant scenario for real world applications. In the parlance of signal analysis, the Fourier transform is unable to efficiently represent a signal that is *localized* in the time domain (though Fourier functions are very good at representing signals that are localized in the frequency domain). This is important in images, for instance, where it is unlikely to find periodicities and quite likely to find large discontinuities. For instance, it is very common to observe one part of an image, such as a text character, having very different characteristics than another part of the image nearby, such as another letter or a blank area with no letters.

To describe a broader and less predictable class of inputs, it is necessary to develop a richer class of basis functions. The new class aims to represent a function space where we are not restricted to assuming those functions are uniformly regular. If we want to be able to decompose naturally occurring signals into the simplest and - for the sake of computation - fewest components possible, we must to develop a dictionary of basis elements that are as similar as possible to the features of naturally occurring signals. There is no "free lunch" and it may very well be the case that the sparse representations of Fourier-alternatives are not right in many cases, but in circumstances where a Fourier transform is inappropriate, we consider classes of sparse wavelet dictionaries as an alternative.

2.2 Sparse Representations: Multiresolution Analysis in Continuous Spaces

A basis set of sparse functions is convenient for a number of reasons. For example, sparse functions are easy to store and projections onto sparse basis vectors require less computation than a non-sparse basis set.[2] The principle benefit of sparse function bases that we focus on is their ability to efficiently approximate parts of signals that are localized in time. In the toy example featured in Figure 2.1, a basis consisting of orthogonal indicator functions would require only a few functions to perfectly reproduce signal B.

Multiresolution analysis (MRA) is a term for the analytical framework whereby signals in $L^2(\mathbb{R})$ are decomposed using a basis set of functions $\mathcal{B}$, the elements of which may be filtered into a sequence of nested subspaces ordered according to their level of smoothness:

$$\{0\} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots \subset V_{n-1} \subset V_n \cdots \subset L^2(\mathbb{R})$$

2. *Sparsity* here refers to sparsity in the time domain and in the frequency domain. A function that is sparse in the time domain means that the support of the function is primarily zero and only nonzero in a small region of the domain. Sparsity in the frequency domain is the same property that underlies Fourier basis functions.

The smoothest subspace is denoted $\{0\}$ and consists of the the set of constant functions indexed by $c \in \mathbb{R}$ where $\forall x \in \mathbb{R}\ c(x) = c$; the roughest subspace $L^2(\mathbb{R})$ includes the very broad class of square integrable functions. The definition of the subspaces V_i depends on which variety of sparse basis functions are being used. Irregardless of the precise definition, though, the subspaces are sequentially obtained by splitting each V_i into V_{i-1} and W_{i-1}, smoother and rougher parts respectively. For a general function $f \in L^2(\mathbb{R})$, the projection of f into the subspace V_i will be progressively smoother as i grows smaller. We will feature an example of how this works in practice at the end of this section (2.2.2).

2.2.1 Principles of Multiresolution Analysis

To define the necessary principles for a practical and interpretable MRA, Mallat composed a set of axioms, whereby subspaces $\{0\} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots \subset V_{n-1} \subset V_n \cdots \subset L^2(\mathbb{R})$ - and the functions which they contain - may be defined:[101]

MRA1 $\cap_{i\in\mathbb{Z}} V_i = \{0\}$ or, equivalently, since we are dealing with a nested sequence of subspaces, $\lim_{i\to-\infty} V_i = \{0\}$

MRA2 $\cup_{i\in\mathbb{Z}} V_i = L^2(\mathbb{R})$ or equivalently, since we are dealing with a nested sequence of subspaces which gradually increases to include all square integrable functions, $\lim_{i\to\infty} V_i = L^2(\mathbb{R})$

MRA3 $\forall f \in V_i,\ \forall m \in \mathbb{Z},\ \exists g \in V_i$ where $f(x) = g(x - 2^l m)$

Choosing to use the length of dyadic intervals as the basic unit of translation, for a fixed smoothness level l, translating one wavelet by the given level's degree of coarseness produces another wavelet.

MRA4 $\forall f \in V_i,\ \exists g \in V_{i-1}$ such that $g(x) = \frac{1}{\sqrt{2}} f(2x)$

With a unit of dilation equal to 2, for any function in a (relatively) rough subspace, there is a smoother function in a (relatively) smooth subspace, where the

smooth function is a dilation of the rough function.

These conditions provide the necessary ingredients to define wavelet (or detail) functions and scaling functions, also known as "mother" and "father" wavelets. For a given level of resolution l, the subspaces V_l and W_l are the (relatively) smoother and rougher orthogonal components which comprise V_{l-1}. That is, the orthogonal complement of W_l in V_{l-1} is V_l. Both father and mother wavelets have norm one, but the integral of father wavelets is equal to one, while the integral of mother wavelets is equal to zero. We consider in detail one class of wavelets in order to make the cascading resolution of wavelets abundantly clear.

2.2.2 Example: Haar Wavelets

Alfred Haar proposed the simple class of piecewise constant functions for the construction of an orthonormal set of easy-to-integrate functions. With the development of MRA, the class of functions defined by the eponymous Haar has become known as the simplest form of wavelet. While, on the one hand, these functions are not continuous, dealing a blow to the development of theoretical properties for this wavelet class (many subsequently developed wavelet types address this), it is nonetheless uniquely suited to identify discontinuities that may exist in signals.[129]

For a baseline resolution level $l = 0$, we may define the "mother" detail function of the Haar wavelet basis as:

$$\psi_0^{(0)}(x) = \mathbb{I}\left(x \in [0, \frac{1}{2})\right) - \mathbb{I}\left(x \in [-\frac{1}{2}, 0)\right)$$

and the "father" scaling function as:

$$\phi_0^{(0)}(x) = \mathbb{I}\left(x \in [0, 1)\right)$$

The superscript denotes the reference level of resolution and the subscript denotes the trans-

lation. It is evident that, the inner product $\langle \psi_0^{(0)}, \phi_0^{(0)} \rangle = 0$. Furthermore, any two distinct functions from among all the functions that are integer translations of the mother and father:

$$\{\psi_k^{(0)} \mid k \in \mathbb{Z}\} \cup \{\phi_k^{(0)} \mid k \in \mathbb{Z}\}$$

will also have an inner product equal to zero. To move from this level $l = 0$ of resolution to a smoother one (i.e. $l = -1$), the collection of detail functions $\{\psi_k^{(0)} \mid k \in \mathbb{Z}\}$ are discarded and the collection of scaling functions $\{\phi_k^{(0)} \mid k \in \mathbb{Z}\}$ are used to produce detail functions $\{\psi_{2k}^{(-1)} \mid k \in \mathbb{Z}\}$ and scaling functions $\{\phi_{2k}^{(-1)} \mid k \in \mathbb{Z}\}$. Since the scaling functions of $l = 0$ were constant over any intervals between integers, the new detail functions will also only have discontinuities at integer values. The scaling functions are similarly smoothed, now having support between any two even integers. In order for basis functions to have norm equal to one, basis functions must be normalized by $\sqrt{2}$.

To move from V_0 to V_1, the father wavelet and mother wavelet at the previous level must be un-dilated. For the mother wavelet this corresponds to $\psi_0^{(1)}(x) = \psi_0^{(0)}\left(\frac{x}{2}\right)$ and similarly for the father wavelet $\phi_0^{(1)}(x) = \phi_0^{(0)}\left(\frac{x}{2}\right)$. In this case, in order for the level $l = 1$ functions to integrate to one, all functions must be normalized by $\frac{1}{\sqrt{2}}$. To move between any other two resolution-adjacent subspaces, this same process is executed.

2.2.3 Discrete Wavelet Transforms of Signals

Practically and computationally, the basis functions considered are not $\{\psi_k^l \mid k, l \in \mathbb{Z}\} \cup \{\phi_k^l \mid k, l \in \mathbb{Z}\}$, as we are dealing with realizeable wavelet approximations and not a limit using countably many basis functions. For that reason, in applications, the wavelet basis is defined within a finite interval (consisting of the range of values at which a signal is measured) and for a finite number of levels of resolution. This means that the class of functions that may be reproduced exactly is not really $L^2(\mathbb{R})$ but a subspace consisting of the span of $B = \{\psi_{ij} \mid i \in I,\ j \in J\} \cup \{\phi_{ij} \mid i \in I,\ j \in J\}$ (where I and J index levels of resolution and

translations).

A signal s that is received may then be projected onto B and represented using the coefficients $(\alpha_{ij})_{i\in I,j\in J}$ and $(\beta_{ij})_{i\in I,j\in J}$ where $\alpha_{ij} = \langle f, \psi_{ij}\rangle$ and $\beta_{ij} = \langle f, \phi_{ij}\rangle$. A signal reconstruction given the coefficients $(\alpha_{ij})_{i\in I,j\in J}$ and $(\beta_{ij})_{i\in I,j\in J}$ is

$$f(x) = \sum_{i\in I}\left(\sum_{j\in J}\alpha_{ij}\psi_{ij}(x) + \beta_{ij}\phi_{ij}(x)\right)$$

We consider this procedure on a practical problem, and note the result alongside that which would be produced by using harmonic functions instead. We choose to focus on functions that defy the typically simple application of a Fourier transform for denoising or compression. As examples, we consider the chirp signal and a multi-year window of a stock's opening price each trading day.[3] A stock's price over time has less predictable patterns and we take notice of how wavelets and Fourier functions provide a sense of overall trends (especially after having thresholded coefficients with sufficiently small values). See figures 2.2.3 and 2.2.3. We observe in both cases, that for a sufficiently large number of basis functions, each approximation method reconstructs the function almost exactly. For smaller numbers of functions, the Fourier representation is least accurate at the boundaries and occasionally introduces periodicities that simply do not exist in the original function. The smoothness of the Fourier approach may be a benefit, as the stock prices represented using wavelets have sharp discontinuities. That being said, the wavelet used in this examples is the Haar wavelet; smoother wavelet functions would be able to capture smooth trends with a fewer number of terms.

3. The chirp signal is so named due to its audial similarity with a bird's chirp and is significant for increasing radar precision. In our case, we are simply interesting in the ability of wavelets and Fourier functions to approximate the function with k terms for the purpose of denoising or compression.

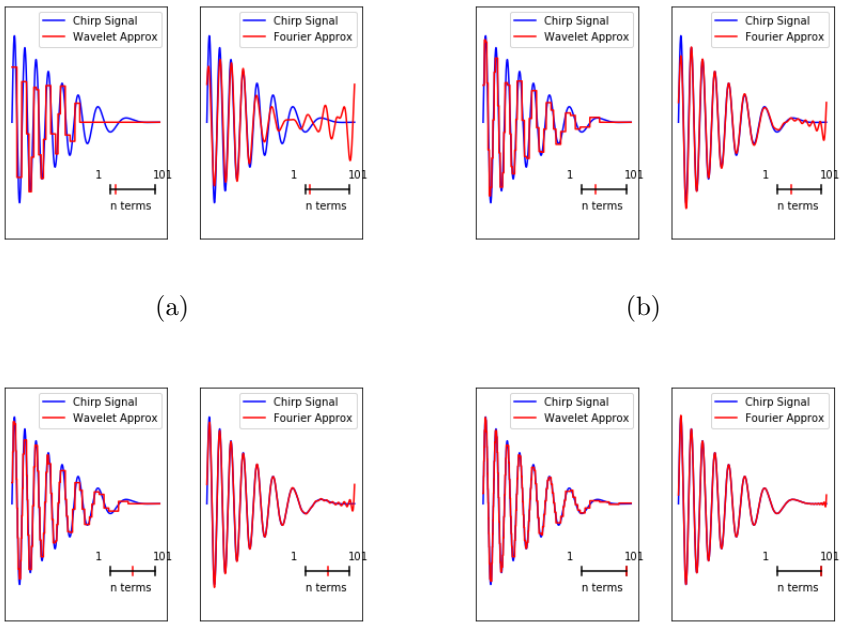


Figure 2.2: Wavelet and Fourier approximations for a chirp signal using 2.2(a): 11, 2.2(b): 21, 2.2(c): 51, and 2.2(d): 101 features.

2.3 Fourier and Multiresolution Analysis in Discrete Spaces

In this section we consider differences that exist between using Fourier and MRA in structured and nonstructured spaces. By a "structured space" we refer to a space whose properties are determined before observing any data or the relations between them. When a signal is observed in $\mathbb{R}$ we do not need to know the signal in order to know that $0, 1 \in \mathbb{R}$ and $d(0, 1) = 1$. We are able to define a complete wavelet basis *a priori* because we can define transforms (DFT or DWT) for the metric space before the data is observed.

In graphs $G = (V, E)$, on the other hand, the observation of the data itself is how we learn the metric space under consideration. Before observing edges, it is impossible to know which observations (i.e. nodes) are closer to each other. Any set of $n = |V|$ orthonormal vectors in $\mathbb{R}$ could be a basis for functions $f : V \to \mathbb{R}^n$, but these vectors will have no

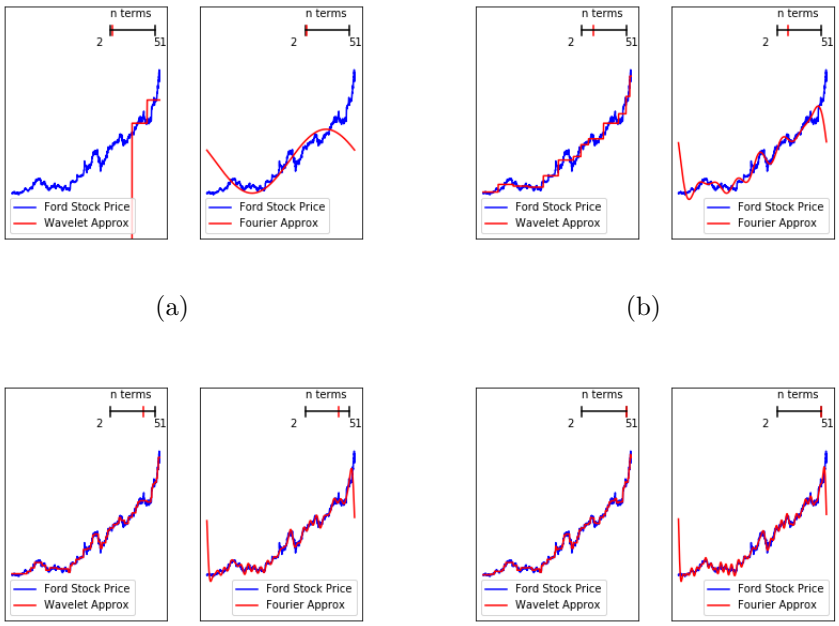


Figure 2.3: Wavelet and Fourier approximations for open prices for Amazon stock using 2.3(a): 2, 2.3(b): 13, 2.3(c): 38, and 2.3(d): 51 features.

correspondence with functions that we are likely to observe on V.[4]

Proposing Fourier or wavelet transforms on graphs is a particularly challenging problem because of the infinite variety of different metric spaces that are realizeable as graphs. Since function bases for graphs must be defined on the fly, the basis set for graphs - whether Fourier or multiresolution - will need to be *data adaptive*. It is necessary to consider the edges between nodes and possibly the functions observed on nodes in order to define sensible basis functions.

A second challenge for the use of MRA on graphs (though not on structured spaces) is the definition of a dilation or translation. As we saw in Mallat's wavelet construction conditions,

4. By *likely* we refer to a simple assumption: when a function is realized on a graph, the function's values at nearby nodes is more similar than the function's values at nodes that are far apart from each other. We do not claim that graphs do not exist where this property is not true, merely that for the sake of problems like denoising graph functions, inference of unknown labels, or clustering, we may only begin to trace a solution if a meaningful metric space may be defined using the graph's nodes and edges. Formulations of this assumption are made rigorously by using regularization operators.

in order to separate wavelets into different levels of resolution and use translations to define all the wavelets at a given level of resolution, we need to know the definition of a dilation and translation across all levels.

Discrete Objects for Consideration Until now we have contrasted graph-based metric spaces with structured metric spaces such as $\mathbb{R}^n$. The challenge of unstructuredness in a graph is also the strength that makes it able to represent a very broad class of observed data, including any finite set of discrete data observations. For instance, if a signal f has measurements at locations $\{k \in \mathbb{Z} \mid 0 \leq k \leq 10\}$, we may instead consider the function domain to be a graph $G = (V, E)$ where $V = \{v_k \mid k \in \mathbb{Z},\ 0 \leq k \leq 10\}$ and $E = \{(v_i, v_j) \mid |i - j| = 1,\ 0 \leq k \leq 10\}$. The fact that predefined bases, such as the Haar wavelet basis described above, already exist means we do not need to redefine the wheel in these cases, but suffice to say that a MRA procedure in general graph-based metric spaces has extremely wide applicability. Additionally, correlation matrices or distance matrices may be understood as graphs whose vertices are matrix indices and whose edge weights are the matrix values at the corresponding index pairs. A discrete wavelet transform on discrete spaces aims to address these cases. More broadly, a discrete wavelet transform may be applied to any linear operator, though many of the motivations we discuss in this thesis apply primarily to linear operators that are symmetric and positive semidefinite.

2.3.1 Connection Between the Fourier Basis and the Eigenvalue Decomposition

Adapting the Fourier transform to the graph setting requires a new perspective on what is so special about the Fourier basis' harmonic functions, specifically their relationship to the Laplace operator $\mathcal{L}$. For $f \in L^2(\mathbb{R}^n)$, $\mathcal{L}f = \nabla \cdot (\nabla f)$, the divergence of the gradient of f. So for a function f, the Laplacian operator applied to f is $\mathcal{L}f = \nabla \cdot (\nabla f) = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2} f$. The eigenvectors of $\mathcal{L}$ are indeed the harmonic functions, of which the Fourier basis functions are

a subset. The gradient of a function gives the directions in which the function changes, and the divergence records the magnitude of those changes. So, one way of thinking about what the Laplacian operator does is it computes how much function values differ locally. In the slightly more manageable case of $n = 2$, given $f : \mathbb{R}^2 \to \mathbb{R}$,

$$\frac{\partial^2}{\partial x^2} f(x,y) = \lim_{h\to 0} \frac{\frac{\partial}{\partial x} f(x+h,y) - \frac{\partial}{\partial x} f(x,y)}{h} = \lim_{h\to 0} \frac{f(x+h,y) + f(x-h,y) - 2f(x,y)}{h^2}$$

$$\frac{\partial^2}{\partial y^2} f(x,y) = \lim_{h\to 0} \frac{\frac{\partial}{\partial y} f(x,y+h) - \frac{\partial}{\partial y} f(x,y)}{h} = \lim_{h\to 0} \frac{f(x,y+h) + f(x,y-h) - 2f(x,y)}{h^2}$$

$$\mathcal{L}f = \lim_{h\to 0} \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y)$$

$$= \lim_{h\to 0} \frac{f(x+h,y) + f(x-h,y) + f(x,y+h) + f(x,y-h) - 4f(x,y)}{h^2}$$

Considered differently, $\mathcal{L}f$ may be approximated over a mesh grid using the finite difference method where, instead of h being a term vanishing to zero, it is the distance between adjacent elements in the grid $h\mathbb{Z} \times h\mathbb{Z}$. This leads us to approximate $\mathcal{L}f$ using the discrete Laplace operator $\mathcal{L}_h$:

$$\mathcal{L}f \approx \mathcal{L}_h f = \frac{f(x+h,y) + f(x-h,y) + f(x,y+h) + f(x,y-h) - 4f(x,y)}{h^2}$$

alternatively, using $(x_1, y_1) \sim (x_2, y_2)$ to denote that (x_1, y_1) and (x_2, y_2) are h units apart,

$$\mathcal{L}f \approx \mathcal{L}_h f = \frac{4}{h^2} \left(\frac{1}{4} \sum_{(x_1,y_1)\sim(x_2,y_2)} f(x_1,y_1) - f(x_2,y_2) \right)$$

Considering the grid as a graph $G = (V, E)$ where $V = h\mathbb{Z} \times h\mathbb{Z}$ and

$$E = \{((x_1,y_1),(x_2,y_2)) \mid |x_1 - x_2| + |y_1 - y_2| = h\}$$

and using $\sim$ as a symbol denoting edge connectedness, this expression of the discrete Laplacian operator is, for a function $f : V \to \mathbb{R}^{|V|}$

$$\mathcal{L}_h f_v = \frac{4}{h^2} \frac{1}{\deg(v)} \sum_{w \in V,\ w \sim v} f_v - f_w$$

We can generalize this form of Laplacian on a mesh grid graph to a general (non-mesh) graph by defining the graph Laplacian similarly. For $G = (V, E)$ and $f : V \to \mathbb{R}^{|V|}$

$$\mathcal{L}_G f_v = \frac{1}{\deg(v)} \sum_{w \in V,\ w \sim v} f_v - f_w$$

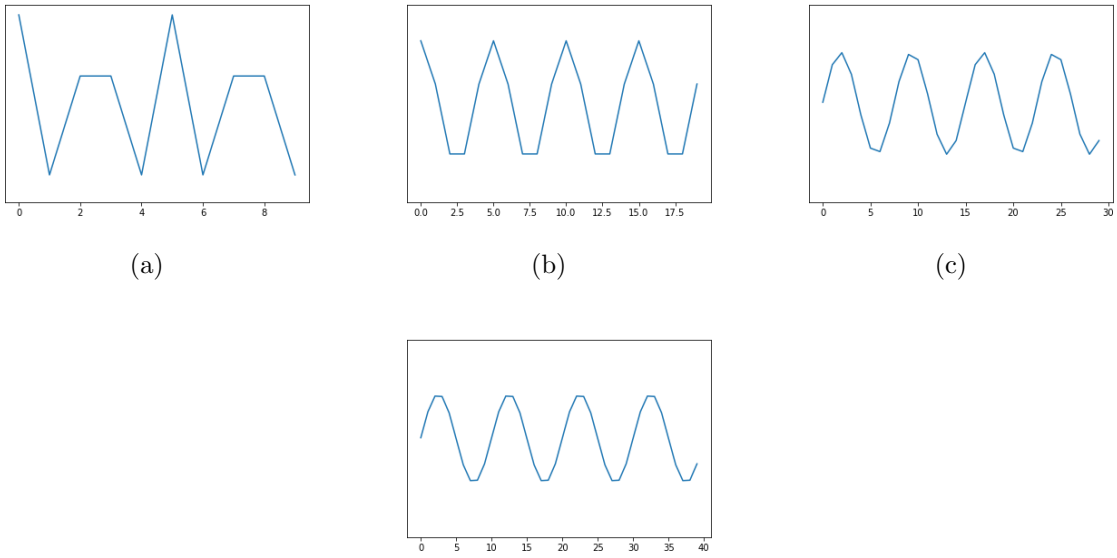
Another way of formulating the $\mathcal{L}_G$ operator is as a matrix. As a simple example, for a cycle graph,

$$C_4 = (V = \{v_1, v_2, v_3, v_4\}, E = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\})$$

the matrix form of the Laplacian operator is:

$$\mathcal{L}_{C_4} = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$

In analogy to the Fourier basis functions being eigenvectors of $\mathcal{L}$, functions f on a graph G may be decomposed into f projected onto each of the eigenvectors of the operator $\mathcal{L}_G$, and represented using the coefficients of the resulting inner products. As a symmetric matrix, the matrix $\mathcal{L}_G$ will have n orthogonal vectors, each with a corresponding eigenvalue. As a real symmetric matrix, all eigenvalues will be nonnegative. To sum up, the graph Laplacian eigenvectors are analogous to the Fourier basis functions in terms of their ability to measure function smoothness. We delve further into quantifications of function smoothness in the section 2.3.2.



(a) (b) (c)

(d)

Figure 2.4: Eigenvectors of the graph Laplacian for a path graph of different lengths: 2.4(a) is length 10; 2.4(b) is length 20, 2.4(c) is length 30, 2.4(d) is length 40

To make the connection between Fourier basis functions and graph Laplacian eigenvectors more visual, we consider a discretization of the time domain as represented by a path graph of length n, P_n. In figure 2.3.1 we can see that, as n grows, the eigenvectors of the $\mathcal{L}_{P_n}$ look more and more similar to Fourier basis functions.

2.3.2 An Operator for Measuring Function Smoothness

In the same manner that the Laplacian operator may be thought of (and approximated) as a measurer of local smoothness, the graph Laplacian $\mathcal{L}_G$ may be used to measure function smoothness for functions on G. Furthermore, the Laplacian may be used to measure a function's smoothness with respect to a manifold. In the case of a graph Laplacian $\mathcal{L}_G$, the operator is measuring function smoothness with respect to a graph G.

We once more gain insight on how function smoothness may be measured by considering the simplest possible graph - the path graph P_n. Just as we noted in Figure 2.3.1, the eigenvectors of $\mathcal{L}_{P_n}$ are discretized harmonic functions. The eigenvalues corresponding to

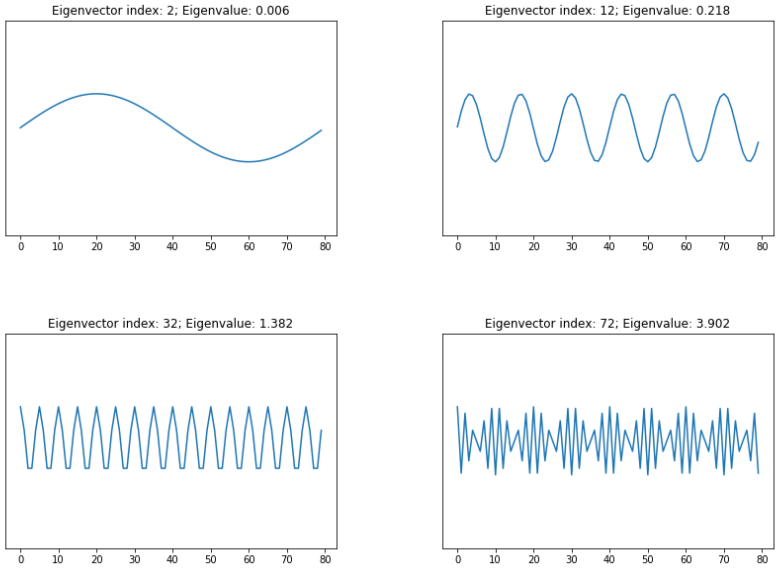


Figure 2.5: Four representative eigenvectors from the path graph of length 80. The higher-valued eigenvectors are "rougher" than the lower-valued ones.

these eigenvectors provide valuable information on how to interpret the action of the graph Laplacian operator. For a path graph of length 80 we show the eigenvectors and eigenvalues for a few representatives in Figure 2.5. It is apparent that the eigenvalues encode the effective frequency of eigenvectors on the path graph - smaller eigenvalues correspond to lower frequencies and higher eigenvalues correspond to higher frequencies.

For a function f, and defining the eigenvectors and eigenvalues as $\{(\lambda_i, v_i) \mid 1 \leq i \leq n\}$ (where eigenvalues are indexed in the order of their magnitude),

$$f^T \mathcal{L}_G f$$

$$= f^T \begin{pmatrix} | & | & | & & | & | \\ v_1 & v_2 & \cdots & v_{n-1} & v_n \\ | & | & | & & | & | \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \lambda_{n-1} & 0 \\ 0 & \cdots & \cdots & 0 & \lambda_n \end{pmatrix} \begin{pmatrix} | & | & | & & | & | \\ v_1 & v_2 & \cdots & v_{n-1} & v_n \\ | & | & | & & | & | \end{pmatrix}^T f$$

$$= \begin{pmatrix} v_1^T f & v_2^T f & \cdots & v_{n-1}^T f & v_n^T f \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & \cdots & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \lambda_{n-1} & 0 \\ 0 & \cdots & \cdots & 0 & \lambda_n \end{pmatrix} \begin{pmatrix} v_1^T f \\ v_2^T f \\ \vdots \\ v_{n-1}^T f \\ v_n^T f \end{pmatrix}^T$$

$$= \sum_{i=1}^{n} \left(\sqrt{\lambda_i} v_i^T f \right)^2$$

The graph Laplacian transform maps the function f to a frequency space whose basis elements are $\{v_i\}_{1 \leq i \leq n}$. This means that $f^T \mathcal{L}_G f$ is equal to $||f||_{\mathcal{L}_G}$, with the magnitude of f in the $\mathcal{L}_G$ frequency space modulated across each dimension by $\sqrt{\lambda_i}$. This results in higher eigenvalue eigenvectors having a higher graph-adaptive notion of frequency. If a function were identical or at least similar in some respect to a low eigenvalue eigenvector (for instance, $f \approx v_2$), the value of $f^T \mathcal{L}_G f \approx v_2^T \mathcal{L}_G v_2 = \lambda_2$. Similarly, if a function is identical or at least similar to a high-eigenvalued eigenvector (for instance, $f \approx v_n$), the value of $f^T \mathcal{L}_G f \approx \lambda_n$. Lower values of this quadratic form will correspond to smoother functions and higher values will correspond to rougher functions.

This face of the graph Laplacian gives it its reputation as a regularization operator. If one is dealing with the objective of denoising observations o on a graph - a graph version of a linear regression problem - the distance of the estimated function f to the observed function $||f - o||$ is balanced with a regularization term: $f^T \mathcal{L}_G f$. In classical signal denoising for functions $f : \mathbb{R}^n \to \mathbb{R}$, a low-pass frequency operator on f smooths the function by ignoring the high frequency regions of the Fourier transform. Specifically, the image of f in the frequency space is projected onto the vector space spanned by the low frequency Fourier basis functions and then re-mapped onto the original vector space. Since a new graph Laplacian is defined for each graph under consideration, the eigenfunctions of the graph Laplacian are adapted to its graph structure and uniquely characterize a definition of smoothness for the graph at hand.

2.3.3 Existing Approaches to Constructing a Wavelet Transforms for Graphs

Having framed the connection between the discrete Fourier transform for signals in $\mathbb{R}^n$, and the discrete Fourier transform for signals on graphs, we now turn to bridging the connection between the wavelet transform for signals in $\mathbb{R}^n$ and the wavelet transform for signals on graphs. Just as in Fourier land we clarified which features of the $\mathbb{R}^n$-based Fourier transform were important to carry over to graph metric spaces, we aim to do the same for wavelet transforms.

In the $\mathbb{R}^n$ case, the wavelets' locality in the time and frequency domains is their most salient feature relative to harmonic functions. As the level of resolution grows finer, the support of wavelets in the time domain becomes smaller. Furthermore, the orthogonality of wavelet bases in $\mathbb{R}^n$ means that they are able to quickly approximate well-behaved signals up to a desired level of precision. Lastly, the construction of wavelets using fixed translations along the real line permitted us to define wavelets with respect to the underlying structure of the metric space - namely, the Cartesian product of real lines. Several proposals exist for constructing wavelets in the graph domain which focus on being:

(a) able to be dilated subject to a desired level of resolution,

(b) orthogonal,

(c) able to capture some inherent graph structure, in turn making the nonzero support of wavelets meaningful.

These proposals carry different advantages and limitations, and we briefly summarize them in order to contrast their approaches with our own.

Vertex-centered Wavelet Construction Methods

Diffusion Wavelets Among the first applications of wavelets to contemporary problems in signal processing on graphs, Coifman *et al.* develop a procedure for defining classes of wavelets on graphs.[36] For a graph $G = (V, E)$, they start by defining a diffusion operator T, central to their wavelet construction, which acts on functions defined on G. The criteria of T are that it smooths graph functions locally, and that as $j \in \mathbb{N}$ grows, $\text{rank}(T^j)$ monotonically shrinks.

The sparsest, finest level of resolution $l = 0$ consists of the Dirac δ-function at each of the graph's vertices $\Phi_0 = \{\delta_v \mid v \in V\}$.[5] Due to the assumption that T is a local graph smoother, the resulting columns of $T\Phi_0$ are still sparse, but less sparse than the matrix whose columns are Φ_0. A sparse QR factorization of $T\Phi_0$[6] produces a sparse orthogonal basis Φ_1 for the space spanned by $T\Phi_0$. Applying the diffusion operator once in this fashion corresponds to a single dilation step in the construction of a multiresolution wavelet basis. Subsequently, the orthogonal basis Φ_j is produced from Φ_{j-1} by applying the operator T^{2^j} to the orthogonal vectors that comprise Φ_{j-1}. As j grows, the operator T^{2^j} has lower numerical rank, leading to a set of orthogonal basis functions which are less sparse, and which are smoother with respect to the graph structure. This process may continue until a resolution smoothness stopping point is achieved or until the columns of Φ_j are effectively constant functions. The general schematic for this algorithm as it appeared in the original publication is reproduced in Figure 2.6.

Diffusion wavelets produce an orthogonal wavelet basis with sparsity that depends on j. The framework of wavelet construction described thus far is vertex-focused. That is, each wavelet is initially defined using an impulse at a single vertex. As the resolution grows coarser, translations are more difficult to define, but in principle, vertices are mapped to

5. We refer to both the set of basis functions and the matrix where these functions are columns as Φ_0.

6. The template for how to compute the sparse factorizations is outlined in pseudocode in the publication of Coifman *et al.* .

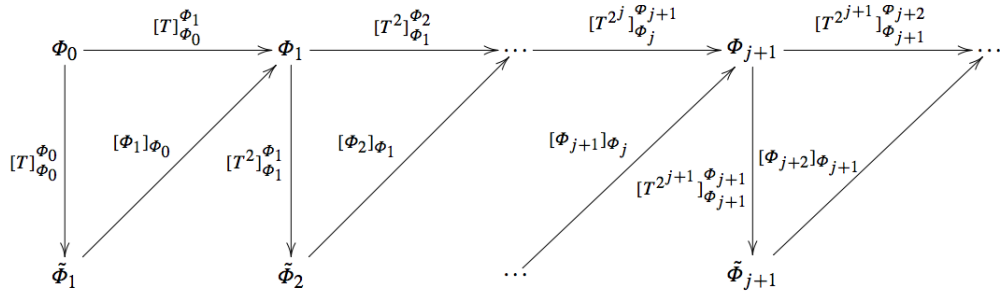


Figure 2.6: Schematic for defining wavelets using diffusion operator T

wavelets. In the clunky world of graph-based MRA, adjacent nodes' wavelets are considered translations of each other.

Diffusion wavelets are appealing in their sensible definition of dilation in terms of a diffusion process and their simple organization based on the diffusion's time scale. Coifman's diffusion wavelets are among the first general wavelet construction methods proposed for multiresolution analysis on graphs, and have found a number of fruitful, recent applications. Diffusion wavelets have been used as the starting point for structural node embeddings,[44] they have provided the first step for developing spectral graph wavelets,[71] and they have opened a number of doors in graph signal processing.[127]

Spectral Graph Wavelets A bit later, spectral graph wavelets (SGW) were developed with many similar qualities to diffusion wavelets.[71] Like diffusion wavelets, SGW construction also uses a diffusion operator - namely the graph Laplacian $\mathcal{L}$. Specifically, they define a wavelet operator

$$T_g = g(\mathcal{L}) = P \begin{pmatrix} g(\lambda_1) & 0 & \cdots & 0 & 0 \\ 0 & g(\lambda_2) & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & g(\lambda_{n-1}) & 0 \\ 0 & 0 & \cdots & 0 & g(\lambda_n) \end{pmatrix} P^T$$

where P is the matrix whose columns are the Laplacian's eigenvectors. If the diffusion operator T of diffusion wavelets is the graph Laplacian, then SGW is equivalent to diffusion wavelets when the function $g(x) = x^2$ is the function that modulates the eigenvalues.[7] By focusing specifically on T_g as parametrized by the graph Laplacian, SGW make the connection between MRA and Fourier analysis explicit, and they focus on the way that g will impact the types of wavelets produced. SGW are also derived from Dirac impulse functions where the wavelet at scale j and vertex v is $\psi_{s,v} = T_g^t \delta_v$ and the corresponding wavelet coefficients for a function f are $c_{s,v} = \langle \psi_{s,v}, f \rangle$. If the definition of $\psi_{s,v}$ is expanded, the wavelet function $\psi_{s,v}$ at each vertex location $w \in V$ is $\psi_{s,v}(w) = \sum_{i=0}^{N-1} g(\lambda_i t) p_i(v) p_i(w)$. This formulation is exactly the inner product of the vectors $(p_i(v))_{0 \leq i \leq N-1}$ and $(p_i(w))_{0 \leq i \leq N-1}$, subject to the weighting of $g_t(x)$. An interpretation of the wavelet function value at entry w generated from impulse function δ_n is the inner product between rows v and w in the matrix P. The values of nodes v, w across all eigenvectors is compared, with their similarity at high frequency eigenvectors being most important. This provides the locality of SGW, since nodes that are close together in a graph will have more similar values across all eigenvectors than nodes that are far apart (and their relative similarity will be greatest, relative to far away nodes, at high frequency eigenvectors).

SGW's straightforward connection tying their interpretation to Fourier eigenvectors and practical large scale implementation using Chebyshev polynomials have led to widespread implementation, likely making them the graph wavelets most frequently in use today. Applications include use as a means of detecting multilevel graph structure,[133, 88] graph embeddings,[43] and image processing.[32]

Limitations of Vertex-centered Wavelet Construction Methods We briefly bring up to some limitations of diffusion wavelets and SGW. For diffusion wavelets, the requirement to choose a specific diffusion operator means that the application of diffusion wavelets

7. In fact, the diffusion operator used by Coifman *et al.* need not be exclusively the graph Laplacian. However, since it is a common choice, SGW may be seen as a generalization of diffusion wavelets.

depends on the operator being well matched to the types of structures the problem aims to identify. Second, the organization of wavelet bases into successive subspaces spanned by sparse orthogonal bases is a convenient representation of graph functions, but less convenient as a representation of graphs themselves. Similarly, for SGW, separating wavelet bases based on scale makes it difficult to learn something specific about graph structure. That is, by decomposing functions f into their inner products with wavelets at each resolution level l as $\{\langle f, \phi_i^{(l)} \rangle \mid 0 \leq i \leq |V| - 1\}$, we are able to identify the wavelets which are most aligned with the function and thus represent functions (or denoise them) using a small number of wavelet coefficients. If we intend, however, to claim the wavelets capture some underlying graph structure, it is necessary to determine the relevance of a resolution level's basis functions. Without a metric of significance, it is impossible to know whether wavelets at resolution l actually carry any meaning. The meaningfulness of any given resolution level is relegated to being outside the procedure's purview. The algorithm simply outputs $j_{\max} N$ wavelets. Additionally, SGW across levels are not necessarily orthogonal, limiting some of the areas in which they may be used. Lastly, SGW depend on the eigenvectors of the graph Laplacian. Though eigendecompositions are convenient for low rank approximations, because power methods can identify the smoothest eigenvectors quickly, they are less adept at capturing local behavior. In practice, eigenvectors are often not sparse, in conflict with the reality that many graphs have strong local properties.

Tree-based (or Almost Tree-Based) Wavelet Constructions

An alternative approach for capturing local patterns that exist in a hierarchical graph structure uses tree approximations. These methods are "data adaptive" in that the wavelet construction depends directly on the graph data - the weight of edges - and makes no smoothness or structural assumptions for the data at hand.

Haar Wavelet Transform The Haar wavelets in $\mathbb{R}^n$ are the simplest building blocks of multiresolution analysis, and the graph Haar wavelet transform (HWT) is a means of constructing Haar wavelets on graphs.[141, 106, 74] Given a graph G, after obtaining a dendrogram or binary rooted tree T (from an as-of-yet unspecified hierarchical clustering scheme), the HWT recursively defines a pairwise averaging at each nonterminal node in the graph (with respect to the root of the tree). So, if the descendants of $v \in T$ are $\text{desc}(v)$, and because of the binary-ness of the tree v has two children nodes $\{u, w\} = \text{ch}(v) \subset V_T$, a wavelet is defined such that it is a positive constant on $\text{desc}(u)$, a negative constant on $\text{desc}(v)$ and zero elsewhere. This wavelet, as discussed in section 2.2.2, will be norm 1 and its entries sum to zero.

Though it is not typically considered, a binary tree could be defined on Euclidean space which would define the same Haar wavelets. This approach to wavelet construction thus defines graph wavelets which reduces to traditional Haar wavelets when applied on a Euclidean domain. The basic idea of using a HWT in the graph context has garnered attention in the graph signal analysis community[29] and semi-supervised learning.[63]

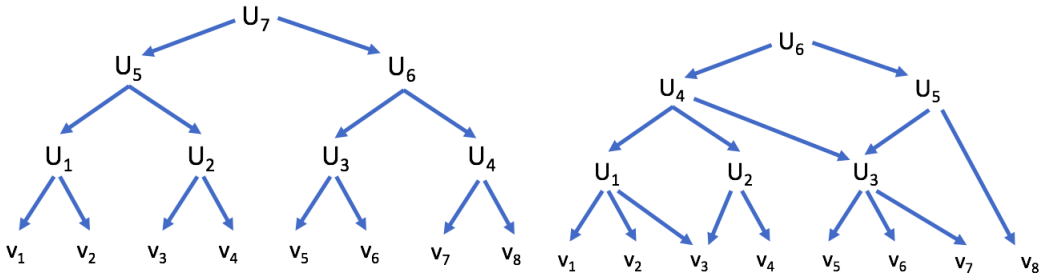
Treelets Using a pre-existing sparse similarity matrix S as the mechanism for identifying node similarity (such as a graph kernel), a tree subgraph T is defined using nodes' pairwise similarity. At iteration $t = 0$, all nodes are available for choosing, and two nodes $u, v \in V$ are selected based on their having the highest off-diagonal value in the similarity matrix $S^{(0)}$. For each iterative step of the transform, the 2×2 submatrix of $S^{(t)}$ corresponding to u, v is diagonalized and the u, v columns and rows of $S^{(t)}$ are rotated according to the rotation matrix resulting from the diagonalization of u, v. One of these nodes is removing from future iterations, and the other remains available for choosing at iteration $t + 1$. Iteration $t + 1$ follows the same process as t except for the correlation matrix is now $S^{(t+1)}$ and the set of available nodes for rotation is of smaller cardinality.[8] The method is remarkably similar to

8. To be clear, the difference between $S^{(t)}$ and $S^{(t+1)}$ is the rotation of $S^{(t)}$. This rotation is affected using a matrix that is the identity except for submatrix of rows and columns corresponding to u, v where it

the Jacobi method for matrix diagonalization, except that treelets converges more quickly because every rotation removes a single index from further rotations (it always converges in less than n rotations). Each iteration may be thought of as the merging of two leaves in a tree, merging leaves progressively until arriving at the tree's root. The output of the algorithm consists of a tree and a sequence of node pair rotations. The rotations may be multiplied together to obtain a matrix whose n columns are the outputted wavelets. In applications, the tree structure may be used for clustering and the wavelets may be used for signal decomposition. The treelets method is differentiated from the simpler HWT by deriving its own hierarchical clustering within the process of wavelet definition, and, more importantly, producing wavelets that are more general than the Haar variety. Criticism of the treelets algorithm, though, primarily focuses on the simplicity of its clustering algorithm. For datasets with more complicated structure than a tree, pairwise relationships are insufficient to decide the sequence of agglomerative clustering.[134]

Multiresolution Matrix Factorization The multiresolution matrix factorization (MMF)[85] was developed after the treelets method, generalizing the treelets algorithm and making that algorithm scalable to larger, not-necessarily-sparse, not-necessarily-tree graph matrices. Similar to treelets, MMF learns a clustering of data on-the-fly through the process of constructing a wavelet basis. Also like treelets, Givens rotations are the mechanism for translating iterations of clustering into wavelet functions. MMF differs from treelets in several important ways. First, MMF directly ties the process of wavelet construction to matrix factorization, making it a wavelet transform that also functions as a numerical approximation tool for myriad large-scale matrix computations. Second, the difference in MMF's objective from treelets is evident in how indices are chosen for rotation. Whereas the treelets method copies the Jacobi method's focus on eliminating the largest off-diagonal entry, MMF's greedy factorization prioritizes minimizing matrix norm error from one iteration to the next. This difference also

is the diagonalization's rotation matrix.



(a) Example of a tree structure making use of k-point rotations where $k = 2$

(b) Example of a tree structure making use of k-point rotations where $k = 2$ or $k = 3$

Figure 2.7: Tree structures based on different rotation schedules

relates to another critical distinction between the methods. Whereas the treelets method (and the earlier HWT) considers Givens k-point rotations where $k = 2$, MMF considers higher-order rotations. In the $k = 2$ case, one node in a rotation is preserved for future rotations, while one is not; in the $k > 2$ case, a proportion of the nodes in a rotation are made "inactive," and a proportion remain "active." This difference leads to MMF summarizing data with tree-*like* structures that are not strictly hierarchical, summarizing more general multi-scale structure in data. This addresses the primary shortcoming of applying treelets to real data. On Figure 2.7 we see the difference between the structure that may be summarized using rotations that are of order 2 or of order greater than 2. Much of the motivation in this thesis relies on the foundation and application of MMF and so further details of motivation, architecture, parameters and explanatory power are delayed to future chapters, primarily chapter 5.

Other Directions

Lastly, we consider two alternative perspectives in multiresolution analysis on graphs bearing direct relevance for our work. While other approaches certainly also exist, we have chosen the particular examples that follow due to the extent that they resemble our own approach.

Fast Factorizations of Symmetric Matrices While fast factorizations are not primarily oriented towards wavelet construction, such as a factorization using structural assumptions

to speed up matrix operations, they offer a useful contrast to the work we do with MMF. Given a hierarchical clustering of matrix indices, the HSS and HODLR matrix formats have been used as models for applications such as matrix compression[11] and Gaussian Process regression.[10, 116] For these methods, a hierarchical clustering method must be used (in practice, the citations in the previous sentence use divisive clustering) and moving from the root of the clustering tree, the submatrix of interactions between nodes in one tree branch and another are summarized with a low rank approximation. The submatrix of a tree branch is recursively defined with low rank approximations between neighboring tree branches. Using the Sherman-Morrison-Woodbury formula and Sylvester determinant theorem, low rank updates to invertible matrices are easily inverted, and the determinant may be computed quickly.

Generalized Haar-Walsh Transform We finally feature one form of generalized HWTs. The generalized tree-based wavelet transform (GTWT)[115] aims to expand the multi-scale wavelet transform.[63] In its simplest version based on a hierarchical clustering tree, the GTWT reduces to the HWT. The GTWT differs, however, both in the types of wavelet filters (higher-order Daubechies wavelets) considered and the way the tree-based wavelets are defined. Instead of a hierarchical clustering algorithm which would could be represented by a tree, GTWT uses a more flexible random walk process that produces a "tree-like" clustering. At the finest level $l = 0$, an ordering of the indices is derived by tracing a nearest neighbor path starting at a random index. An averaging filter produces a smaller number of datapoints at the next coarser level. At that level, the nearest neighbor path produces an ordering of these new points. In figure 2.8, a schematic of the summary of graph nodes being joined for wavelet definition is displayed as it appeared in the original publication. Whereas in the simplest case of Haar wavelets, the GTWT reduces to the HWT, when a broader variety of wavelets is considered, the sequence of rotations becomes a different beast altogether.

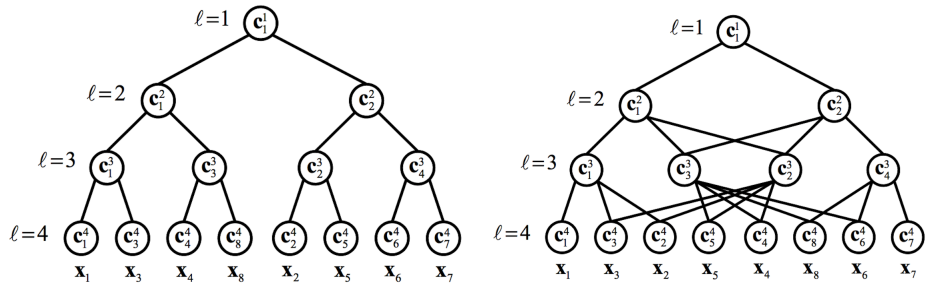


Figure 2.8: Schematic for defining wavelets using GTWT

2.4 Summary

Fourier analysis and MRA provide a framework for signal processing that has revolutionized the way we transmit information. In some ways complementary and in some ways different, these two frameworks are intertwined in both discrete and nondiscrete metric spaces. Their objectives are similar - efficient representation of information - but their different building blocks lead to different results. While the infinitely differentiable basis functions of Fourier analysis translate into an algorithmically simple, easily interpretable tool, time-frequency localized wavelets excel in regimes where there is a sharp discontinuity or strong local behavior. Recent extensions of these frameworks to graph settings sets the stage for expanding the construction and application of Fourier and wavelet transforms. We have shown that the differences between Fourier and MRA carry over into the graph setting, with clear analogies between the behavior of these analytical framework in continuous space and in graphs. The plethora of new approaches to applying wavelet-based methods to large-scale applied problems inspires confidence that MRA has a lot to offer current real-world problems. Subsequent chapters of this thesis focus on the properties and performance of one particular approach - MMF - and describe the novel contribution of this method in light of some ideas from hyperbolic embeddings.

CHAPTER 3

TREES, TREE-LIKENESS, AND HYPERBOLIC METRIC SPACES

There never has been, and until we see it we never shall believe that there can be, a system of geometry worthy of the name, which has any material departures from the plan laid down by Euclid.

Augustus De Morgan (1848)

This thesis focuses predominantly on the application of multiresolution analysis (MRA) to the general class of graph metric spaces. MRA depends on the definition of a mother wavelet that is localized in time and frequency. In the Euclidean domain, the structuredness of the metric space $\mathbb{R}^n$ means that by using translations and dilations a simple locally smooth function such as the Haar wavelet or Mexican hat wavelet is sufficient to define a countable set of basis functions that can approximate square integrable real-valued functions arbitrarily well. A graph metric space is much more general than $\mathbb{R}^n$, and mother wavelets on a graph will inevitably depend on the structure of the graph at hand. Predefined wavelet bases will be unable to characterize function smoothness.

For instance, one trivial wavelet construction method for graphs is to impose an ordering of the graph's nodes $V = \{v_1, \ldots, v_n\}$ and to define a mother wavelet $\psi_t(v_i) = \mathbb{I}(i \in \{t\}) - \mathbb{I}(i \in \{t+s\})$ with t and s defining the location and the width of the support of ψ. Through dilations and translations this wavelet may be used to define a basis on V. These wavelets may qualify as a multiresolution basis, but they do not inspire much confidence in their ability to capture coarse or fine levels of function smoothness on G. In fact, only if G is a path graph, does this wavelet construction make any sense at all. To develop sensible techniques for wavelet construction, we add some color to the characterization of graph domains we are likely to consider. In this chapter we focus on existing research tying real world graphs to hyperbolic space as well as the way that hyperbolic space is a natural habitat for real world graphs.

It seems unavoidable in addressing a massive problem like "what sort of graphs do we want to model for graph-based machine learning?" that we define a large class of graphs. Our aim is to specify conditions on graphs or classes of graphs that will provide a starting point for any wavelet construction method. This starting point must be specific enough that it provides direction on which method to use, but general enough to describe the unknown graphs to be encountered. Our decision is ultimately to model graphs using hyperbolic metric spaces[1] and their "tree-like" properties. To justify this decision, we consider research on the hyperbolicity and tree-likeness of real world graphs. We chart this path by:

1. introducing properties of hyperbolic spaces,
2. surveying several existing approaches to graph embeddings in hyperbolic metric spaces,
3. considering several tree representations and approximations of graphs, and
4. expanding the class of tree-based methods to include methods making use of products, sums, or collections of trees.

3.1 Graphs and their Hyperbolicity

3.1.1 Some Properties of Hyperbolic Geometry

The continuous hyperbolic metric space is a continuous metric space similar to Euclidean space with one notable exception: the removal of Euclid's "parallel postulate," an axiom of Euclid's geometry:

> *If a line segment intersects two straight lines forming two interior angles on the same side that sum to less than two right angles, then the two lines, if extended indefinitely, meet on that side on which the angles sum to less than two right angles.*[72]

1. We intend the term *hyperbolic metric space* to include both discrete and continuous metric spaces. By a *discrete* metric space we refer to a metric space whose elements are finite or countable.

A more recent proposition equivalent to this postulate is known as Playfair's axiom:

> *For any line l and a point p not on this line, there exists exactly one line l_p may be drawn that goes through this point and never intersects the line l.*[114]

In hyperbolic geometry we instead consider an alternative parallel axiom

> *For any line l and a point p not on this line, there exist at least two distinct lines l_p^1, l_p^2 that may be drawn, which go through this point and never intersects the line l.*[95]

In the world of non-Euclidean geometries, hyperbolic geometry can be considered a counterpart to elliptic geometry. For any line l and a point p not on l, define s_p to be the set of lines that contain p and do not intersect l. In hyperbolic geometry, the cardinality $|s_p| > 1$, in Euclidean geometry $|s_p| = 1$, and in elliptic geometry $|s_p| = 0$. Together with Euclid's other first principles, non-Euclidean parallel postulates generate a non-Euclidean geometry with a number of properties unlike those we expect to see in the real world.[2] The parallel postulates for these spaces are equivalent to a few other identifying properties detailed in table 3.1.[3,4] Interestingly, while the circumference of a circle in hyperbolic space is indeed recorded as "¿2π" in table 3.1, the actual relationship between circumference and diameter is notably exponential.

2. Hyperbolic geometry is actually more familiar than many realize. For instance, hyperbolic models of metric spaces are central to local representations of Minkowski space for special relativity. A more immediately recognizeable appearance of hyperbolic geometry is the surface of real-world coral reefs.

3. The difference in a circle's area in hyperbolic and Euclidean space is a particularly relevant point of comparison. For a rooted tree with an exponentially growing number of leaves as a function of tree depth, an embedding will be possible only if the embedding space is able to preserve the relative distance between tree leaves and the root, and *also* the relative spacing between leaves. Whereas the area in a circle of (2-dimensional) Euclidean space grows quadratically, the same circle's volume in hyperbolic space grows at least exponentially.

4. In the formula for area of a circle in an elliptic metric space: r denotes the radius of the circle and R denotes the radius of the sphere which constitutes the manifold that the circle lies upon - if the model of hyperbolic space were something other than a sphere, we would need a different formula

Property	Hyperbolic	Euclidean	Elliptic
Sectional curvature (κ)	$\kappa < 0$	$\kappa = 0$	$\kappa > 0$
Triangle angles (x, y, z)	$x + y + z < 180^o$	$x + y + z = 180^o$	$x + y + z > 180^o$
Circumf.::diam. of a circle	$> 2\pi$	2π	$< 2\pi$
Area of a circle	$2\pi\cosh(r\sqrt{-\kappa}) - 1$	πr^2	$4\pi R\sin\left(\frac{r}{R}\right)$

Table 3.1: Comparing properties of spaces based on their respective parallel postulates

3.1.2 Gromov δ-Hyperbolicity

A hyperbolic metric space is a metric space incorporating the basic properties of hyperbolic geometry. To describe this geometry further, and hyperbolic metric spaces in particular, we delve deeper into ways of describing hyperbolic metric spaces. While some might simply consider whether or not a space is hyperbolic - whether the relevant parallel postulate will hold true or not - we use a property of the space's triangles to characterize metric properties with more finesse. The property described in table 3.1, where the sum triangles' angles are less than 180 degrees, leads to a "slim" triangle. Mikhail Gromov used this triangle property to not only characterize whether or not a space is hyperbolic, but the degree to which a space is hyperbolic.[67] We first consider a convenient definition relating to triangle slimness, and then consider an equivalent definition of hyperbolicity known as the *four point condition* (4PC).

Definitions of δ-hyperbolicity

Definition 3.1.1. δ-slimness Consider a geodesic triangle T_{xyz} with corners x, y, z and line segments $s_1 = [x, y]$, $s_2 = [x, z]$, and $s_3 = [x, z]$ as the sides. If $\exists\delta > 0$ such that $\forall a \in s_1$, $\exists b \in s_2 \cup s_3$ such that $d(a, b) < \delta$, and the same property is true $\forall a \in s_2$ and $\forall a \in s_3$, then T_{xyz} is δ-slim.[5]

This concept of δ-slimness provides us with the necessary material to define δ-hyperbolicity.

Definition 3.1.2. δ-hyperbolic Given a metric space (M, d), if $\exists\delta > 0$ such that $\forall x, y, z \in$

5. Stated differently, if for any two sides of a triangle, the union of δ-balls around all the points on those sides contains the third side, then the triangle is δ-slim. See figure 3.1.

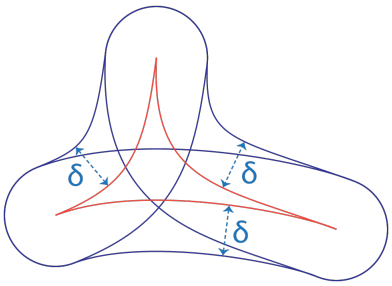


Figure 3.1: A δ-slim triangle

M, the triangle T_{xyz} is δ-slim, and if there is no $\delta_o < \delta$ such that this is true, then (M, d) is a δ-hyperbolic metric space.

A space's δ-hyperbolicity may also be defined using the 4PC. First, it is necessary to define the Gromov product between two elements with respect to a fixed point.

Definition 3.1.3. For a metric space (M, d), $\forall z \in M$, $\forall x, y \in M$ the Gromov product between x and y with respect to fixed point z is

$$(x, y)_z = \frac{1}{2}\left(d(x, z) + d(y, z) - d(x, y)\right)$$

Definition 3.1.4. A metric space (M, d) is δ-hyperbolic if $\forall w, x, y, z \in M$,

$$(x, z)_w \geq \min\left((x, y)_w, (y, z)_w\right) - \delta$$

or equivalently,

$$d(w, x) + d(y, z) \leq \max\left(d(w, y) + d(x, z), d(w, z) + d(x, y)\right) + 2\delta$$

The 4PC may also be formulated differently. A picture to help visualize definition 3.1.5 is in figure 3.1.2.

Definition 3.1.5. A metric space (M, d) is δ-hyperbolic if $\forall w, x, y, z \in M$ it is the case that the largest difference between any two elements among $\{d(w, x) + d(y, z), d(w, y) +$

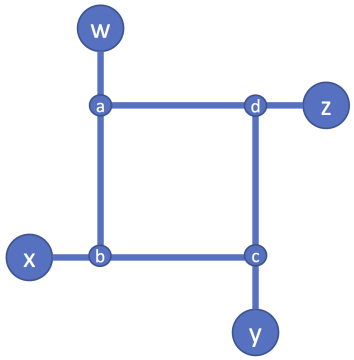


Figure 3.2: Layout for visualizing the four point condition using node $w, x, y, z \in V$

$d(x, z), d(w, z) + d(x, y)\}$ is no greater than 2δ.

Using figure 3.1.2 it is possible to see how cycles make spaces less hyperbolic. Let $l_{ab}, l_{bc}, l_{cd}, l_{da}$ be the distances between the corners of the loop made by a, b, c, d in figure 3.1.2, and let those lengths all be equal to l. Using the 4PC, the hyperbolicity of this space will then be at least $2l$, the maximum difference between two of the elements in

$$\{l_{ab} + l_{cd}, l_{ac} + l_{bd}, l_{ad} + l_{bc}\} = \{l + l, (l + l) + (l + l), l + l\} = \{2l, 4l, 2l\}$$

This provides the intuition that a 0-hyperbolic space is a tree, because no loops exist.

Furthermore, we can see that it is the length of the cycle itself and not the length from say, w to a which makes the difference. If we consider the difference between the two maximum pairwise sums among w, b, c, d, for example, where $l_{wa} = k$, the resulting set of pair sums is

$$\{d(w, b) + d(c, d), d(w, c) + d(b, d), d(w, d) + d(bc)\}$$

$$= \{d(w, a) + d(a, b) + d(c, d), d(w, a) + d(a, c) + d(b, d), d(w, a) + d(a, d) + d(bc)\}$$

Since each term in the set is simply augmented by $d(w, a)$, the maximum difference among pair sums does not change. This means that for any four vertices, the evaluation of their 4PC may be reduced to a smaller problem. Specifically, for w, x, y, z, we can substitute a, b, c, d,

which are the corners of the loop that is formed by using the geodesics $g_{wx}, g_{xy}, g_{yz}, g_{zw}$ whose union results in the smallest length loop. The "corners" of the loop are the closest paths of w, x, y, z to an element of the loop. This reduction of the problem of defining hyperbolicity is found in the work of Chepoi *et al.*[31] The set of four nodes w, x, y, z having geodesics that result in the shortest length loop and also the longest length loop are we call the loop's potential corners: a, b, c, d in figure 3.1.2. A loop of this type defines an equivalence class of four node sets all of whom will have the same maximum difference among node pair sums.

Examples of hyperbolic metric spaces and their degree of hyperbolicity

We focus first on interpreting the δ-slim definition. In an unbounded Euclidean space, $\forall \delta > 0$ it will always be possible to find a triangle that is not δ-slim. Consider, for example that any triangle whose height[6] is greater than δ will not be δ-slim. The constant negative curvature of hyperbolic space, however, means that no matter how large the size of a triangle or length of its sides grow, the triangle remains δ-slim.

We consider some examples of metric spaces and their associated δ-hyperbolicity:

- A bounded metric space, such as $(0, 1) \times (0, 1)$: Since this a bounded metric space, and the distance between any two points is at most $\sqrt{2}$, every triangle in this metric space is at least $\sqrt{2}$-slim. A similar argument may be made for any bounded metric space that the diameter of the metric space can serve as a δ for which every triangle will be δ-slim. This trivial example leads to a bit of confusion on whether δ-hyperbolicity actually implies the variety of hyperbolicity we expect based on intuition. For bounded metric spaces, it is worth considering the relationship between δ and a metric space's diameter to make sure that the constant δ conveys a meaningful characterization of the space.

6. As defined by the minimum distance between the intersection of two sides and the third side.

- The metric space $\mathbb{R}^2$: This metric space has Euclidean geometry, and as such, is not hyperbolic. How is this reflected in the hyperbolicity constant δ? Suppose that there exists a $\delta > 0$ such that every triangle in $\mathbb{R}^2$ is δ-slim. Consider the equilateral triangle with corners $x, y, z \in \mathbb{R}^2$, where the length of the sides are 2δ. Then the distance from x to the nearest point on geodesic g_{xy} (or by symmetry g_{xz}) is $\frac{2\delta\sqrt{3}}{2} = \delta\sqrt{3}$. This triangle is $\delta\sqrt{3}$-slim and $\delta\sqrt{3} > \delta$, therefore no such δ exists and the hyperbolicity of $\mathbb{R}^2$ is $\delta \to \infty$.

- A tree graph: A tree $G_T = (V, E)$ has the property that there are no loops - the intersection of all paths between two nodes $u, v \in V$ in the tree is the shortest path between the two nodes. Consider any three points u, v, w and the geodesic triangle T_{uvw} that they form. Let s_1 be one of the triangle's sides and let s_2, s_3 be the other two sides. $\forall x \in s_1$, let $\gamma_x = \min_{y \in s_2 \cup s_3} d(x, y)$. Suppose that $\exists x$ such that $\gamma_x > 0$. Then it must be the case that $x \notin s_2 \cup s_3$, or else $\gamma_x = 0$. Define x_{s_2} and x_{s_3} to be the nodes on s_1 closest to x for which $\exists y_2 \in s_2$, such that $d(x_{s_2}, y_2) = 0$ and $\exists y_3 \in s_3$, such that $d(x_{s_3}, y_3) = 0$. By construction, these points must exist, because in the extreme case they will just be two elements out of $\{u, v, w\}$, and they must both be at least γ away from x. Denote t as the point connecting s_2 and s_3 which is closest to the corner of s_1 and s_2. Now we consider the triangle $T_{tx_{s_2}x_{s_3}}$. Two paths of edges exist from t to x_{s_2} whose intersection is empty. The two paths are (1) the path $g_{tx_{s_3}}$ together with the path $g_{x_{s_2}x_{s_3}}$, and (2) the path $g_{tx_{s_2}}$. Since these edge set of these paths has an empty intersection, they constitute a loop and T is not a tree. Therefore $\gamma_x > 0$ results in a contradiction and we conclude $\gamma_x = 0$. This means that all triangles in G_T are δ-slim with $\delta = 0$.

- A cycle graph: Let C_n denote the graph that is a cycle of length n, and define a triangle T_{abc} where $a, b, c \in V(C_n)$. Since the sides of the triangle are geodesics, the length of any of the sides is no more than $\frac{n}{2}$. We consider a triangle in figure 3.3 that

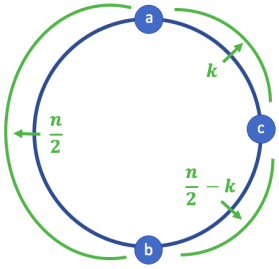


Figure 3.3: The cycle graph C_n and triangle abc

consists of the entire cycle and the geodesic $\bar{ab}$ has length $\frac{n}{2}$. Let $m_{\bar{ab}}$ be the midpoint of $\bar{ab}$ and we assume that $d(m_{\bar{ab}}, a) = d(m_{\bar{ab}}, b) = \frac{n}{2}$. This means the path connecting $m_{\bar{ab}}$ to $\bar{ac} \cup \bar{bc}$ must be of length at least $\frac{n}{4}$. We conclude that the cycle graph C_n is $\frac{n}{4}$-hyperbolic.

- Chordal graphs: A graph G is called *chordal* if there are no induced cycle subgraphs of length greater than 3. A k-chordal graph is a graph with no induced cycle subgraphs of length greater than k. A chordal graph may be related to a tree graph by noting that a k-chordal graph with $k = 2$ is simply a tree. Similar to the difference in k between trees and 3-chordal graphs, there is also a similar progression in δ-hyperbolicity. Whereas a tree is 0-hyperbolic, a 3-chordal graph is 1-hyperbolic, and a k-chordal graph is $\left\lfloor \frac{k}{2} \right\rfloor$-hyperbolic.[22, 16, 142]

To sum up what we infer from these examples, graphs that are trees are extremely hyperbolic, graphs that are 3-chordal (essentially clusters of nodes, or combinations of trees and clusters of nodes like core-periphery graphs) are slightly less hyperbolic, and the longer an induced cycle in a graph, the less hyperbolic its metric space. The most extreme case of a metric space with induced cycles of any size is an unbounded Euclidean space. These examples provide a sliding scale for the tree-likeness of a graph.

3.1.3 A Sample Case in the Tree-likeness of Hyperbolic Space: $\mathbb{R}$-trees

Until now we have considered primarily metric spaces with a finite number of elements such as a graph with a metric defined by its edge set. A simple way of making the uncountable $\mathbb{R}^2$ into a hyperbolic metric space is, instead of using the Euclidean shortest path metric d, to use the Paris metric d_o^{Paris}.[7] This metric space is an example of an $\mathbb{R}$-tree. A 0-hyperbolic metric space X is a $\mathbb{R}$-tree if $\forall x, y, z \in X \ \exists t \in g_{xy}$ such that $g_{xz} \cap g_{yz} = g_{tz}$.[8] As a 0-hyperbolic space, $\mathbb{R}$-trees are called "continuous trees" and are the simplest example of a continuous hyperbolic space, but we will consider that there are non-0-hyperbolic spaces that also have tree-like properties.[7]

The ways in which hyperbolic space has properties that are similar to trees may be observed from several different directions,[70] but we focus particularly on embeddings. Gromov's development of a method for quantitatively measuring a space's hyperbolicity inspired a field of research attempting to find embeddings of general hyperbolic spaces in $\mathbb{R}$-trees. Some methods find $\mathbb{R}$-trees which locally approximate a hyperbolic space's local structure.[38] Similarly, fast algorithms exist for measuring distances in a δ-hyperbolic space using tree approximations of the space.[31] Other approaches aim to approximate the hyperbolic space's boundary by defining a continuous function from the hyperbolic space's boundary to the $\mathbb{R}$-tree's.[67, 69] The ability of $\mathbb{R}$-trees to well-approximate different aspects of general hyperbolic spaces has led to advances in a number of fields related to search and routing.[124] A necessary first step in several applications of hyperbolic embeddings of data has indeed relied on a close connection between finding approximating trees and an associated hyperbolic metric space.[83, 136] Empirically, the δ-hyperbolicity is a reliable way of identifying whether or not a graph has a core-periphery structure; the δ constant will describe whether

7. The metric d_o^{Paris} is defined as follows: for $x, y \in \mathbb{R}^2$, if there exists a ray r from $o \in \mathbb{R}^2$ such that $x, y \in r$, then $d_o^{\text{Paris}}(x, y) = d(x, y)$. Otherwise, $d_o^{\text{Paris}}(x, y) = d(o, x) + d(o, y)$. The metric is an abstraction of the city streets of Paris which, in several locations, have the appearance of all shooting out from a single central point - the Arc de Triomphe, *pour example*.

8. This is equivalent to saying that any triangle is a tripod.

or not there are many paths that are notably different between nodes that are far apart. If there are not, then the graph has a character that is tree-like.[6]

3.2 Hyperbolic Embeddings of Data

We next turn to the ability of hyperbolic space to serve as the embedding space of general metric spaces, and under which conditions. In contrast to embeddings of metric spaces in metric trees, embedding metric spaces in hyperbolic space is a mapping of continuous space to continuous space *without any discretizations*. Even in the case of a discretized graph metric, embedding a graph with a presumed tree-like structure in an $\mathbb{R}$-tree metric may be considered a relaxation to the problem of finding an optimal embedding of a graph into a hyperbolic space. In this section we review approaches to hyperbolic embeddings, the manner in which these approaches have generated novel solutions to persistent problems, and under which conditions these embeddings are possible.

In large networks, particularly large social networks or knowledge graphs, a constant challenge is estimating pairwise distances between vertices for problems related to community detection or recommendation systems. It has been both theoretically and empirically defended that low-dimensional hyperbolic (relative to Euclidean) embeddings radically speed up the time to accurately calculate *most* pairwise distances with low distortion.[30, 66] One early contribution in the field of hyperbolic embeddings is Kleinberg *et al.*'s algorithm for embedding finite graphs into hyperbolic space using a "greedy" embedding algorithm.[83][9] Later theoretical work has attempted to characterize the types of data where hyperbolic embeddings are likely to have the lowest distortion, and thus are most likely to accurately capture the metric space's structure. Some of these characteristics include quasi-cyclicity,[136][10]

9. A *greedy embedding* is a variety of embedding that avoids a particularly damaging form of distortion. The metric embedding $f : G \to M$ is a greedy embedding if $\forall u, v \in V, \exists w \in \text{nbhd}(v)$ such that $d_M(f(v), f(w)) \leq d_M(f(u), f(v))$. That is, there is no pair of vertices that is embedded such that the distance between the two vertices is made closer than the distance between a vertex and one of its closest adjacent vertex.

10. A metric space (V, d) is quasi-cyclic q if the largest induced α-cycle is smaller than q. An α-cycle is a

node degrees being distributed according to an exponential distribution,[125], or presence of a core periphery structure.[125]

Three notable approaches to implementing graph metric embeddings in continuous hyperbolic spaces are: neural embeddings, Lorentzian distance embeddings, and embeddings using the Poincare Disk model.

Neural network embeddings have found wide application in natural language processing (NLP), forming word representations for tasks like machine translation. The success of neural embeddings for language has motivated the use of neural networks for representing other unstructured data such as graphs. Instead of using the Skipgram architecture as is commonly done in NLP, random walks are used to abstract some of the contextual information associated with vertices. The network architecture is then learned using backpropagation as it would be in general neural embedding systems.[25]

The Poincaré ball is one of many models for a metric space with negative curvature. Nickel and Kiela use Riemannian optimization to learn metric embeddings that minimize a loss that depends on a hyperbolic distance function. This leads to results that suggest that for many text or graph-based datasets the Poincaré model is a more accurate representation than a Euclidean model. Furthermore, their path towards learning the embedding avoids the computational complexity involved in fitting neural embeddings for large networks.[110, 131] Recent work has suggested that the Lorentzian distance metric is more appropriate than the Poincaré metric based on complexity considerations.[92]

Thinking of hyperbolic space as a smooth version of trees, as approximating trees, or as "tree-like" gives a rationale for using hyperbolic space as the domain to model hierarchical clustering structures. Many complex networks exhibit hierarchies that may be summarized using a hierarchical tree structure[33] admit a smooth representation if a hyperbolic metric structure is used instead.[87, 30] Non-discrete representations are frequently beneficial

set of nodes $U \subset V$ such that the induced subgraph of those nodes has similar metric properties to a cycle graph metric space (C, d_C). Specifically $\exists \alpha$ such that $\forall u, v \in U$, $\alpha d_C(u, v) \leq d(u, v) \leq d_C(u, v)$

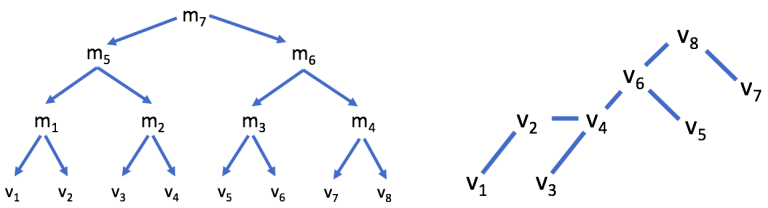
in machine learning applications when discrete, tree representations can be overly simplifying. Still, we turn next to tree representations, for their simplicity, generality, and frequent adoption as the backbone of techniques for multiresolution analysis.

3.3 Trees: Models of Real-World Data and Graph Approximations

Empirically and practically, trees and hyperbolic metric spaces are convenient representations for real-world data. Empirically - we observe many real-world networks or data-derived networks bearing a close similarity to hyperbolic or $\mathbb{R}$-tree metric spaces.[87] Practically - trees underly many fast algorithms, such as binary search. For instance, R-trees (distinguished from the $\mathbb{R}$-tree) are a tree data structure used for organizing data lying in multidimensional spaces. Constructed using a recursive partitioning algorithm, the data structure can speed up spatial queries such as "find the nearest supermarket to my location."

By organizing a dataset of spatial data using a tree structure, we pair a graphical model with our intuition that data has clustering at multiple levels of resolution. This model also corresponds to drawing a dendrogram summarizing the clustering structure of a dataset. We note as an aside that dendrograms (or decision trees) are trees whose leaves correspond to datapoints and whose branch splittings correspond to the merging of clusters. Dendrograms may be used to define equivalence classes of rooted tree graphs to represent multi-layer clustering. The correspondence between dendrograms and rooted tree graphs is important to our subsequent discussions and we present an example of a dendrogram-tree pair in Figure 3.4.

In the next section we consider existing approaches to use trees to represent and approximate real-world graphs.



(a) Dendrogram: v_i nodes are the nodes representing datapoints, and m_i represent the sequential merges of datapoints

(b) Tree graph recording the merging information in the dendrogram

Figure 3.4: This figure shows the correspondence between dendrogram and tree graph. One important qualification must be noted. Whereas a dendrogram is agnostic to within-cluster ordering, in a tree graph there is a strict hierarchy at each merge. This may result from choosing one of the nodes being merged to be the "father" and the others to be "descendents," or there may be some structural reason why one node belongs higher in a hierarchy than others.

3.3.1 Ultrametric Spaces

An ultrametric tree (U, d_u) is a rooted tree metric space with a modification to the traditional triangle inequality. Instead of $\forall x, y, z \in U,\ d(x, z) \leq d(x, y) + d(x, z)$, we have $\forall x, y, z \in U,\ d_u(x, z) \leq \max(d_u(x, y), d_u(y, z))$. This results in a metric space that is more general than $\mathbb{R}^n$ with the Paris metric and results in some interesting properties:

- Intersecting balls: For $a, b \in U,\ r_a, r_b \in \mathbb{R}^+$, if $B(a, r_a) \cap B(b, r_b) \neq \emptyset$ then either $B(a, r_a) \subseteq B(b, r_b)$ or $B(b, r_b) \subseteq B(a, r_a)$.
- Ball centers: For a ball $B(x, r)$ in the metric space (U, d_u), $\forall a, b \in B(x, r)$, $B(a, r) = B(b, r)$.

An example of an ultrametric space is $(\mathbb{Q}, d_2)$ where d_2 is defined using the diadic norm. For $x \in \mathbb{Q}$, since x is rational, there is a representation of x in terms of its prime factors $x = p_1^{n_{p_1}} p_2^{n_{p_2}} \cdots$ where $p_1, p_2, \ldots$ are distinct prime numbers and exponents $n_{p_1}, n_{p_2}, \cdots \in \mathbb{Z}$. In this ultrametric space we define $x = \frac{p2^n}{q}$ where $n \in \mathbb{Z}$ and where $p, q \in \mathbb{Z}$ are the product of all prime terms in the prime factor representation other than two. The diadic norm is $\|x\|_{\text{diadic}} = \frac{1}{2^{\text{order}_2(x)}}$ where order(x) is equal to n for the diadic representation $x = \frac{p2^n}{q}$. The

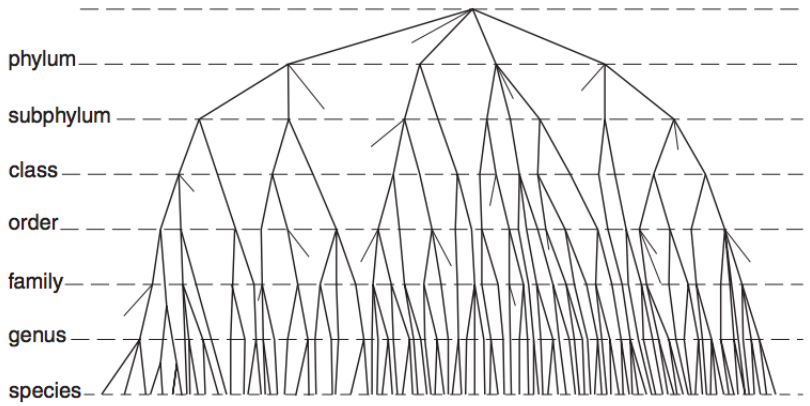


Figure 3.5: This visual of Linnaean taxonomy helps to explain how distance is measured in an ultrametric space. Leaves of the tree are more closely related the more recently they share a common ancestor.

ultrametric space's distance for $a, b \in \mathbb{Q}$ is $d_{\text{diadic}}(a, b) = \|a - b\|_{\text{diadic}}$.[47] Another picture of an intuitive ultrametric space is in Figure 3.5.

The hierarchy natural to an ultrametric space makes it a convenient model for modeling hierarchical structure in real world settings. One path towards uncovering said structure is to embed a general metric space in an ultrametric space. Abraham *et al.* develop an algorithm for such an embedding by developing a novel usage of partitioning tools used for metric Ramsey problems.[4] This metric embedding of (M, d) achieves an ϵ-distortion for at least $100(1 - \delta(\epsilon))\%$ of the pairwise distances between elements in M. The remaining $100\delta(\epsilon)\%$ pairwise distances may have larger distortion.

Another approach to embedding metric spaces in ultrametric spaces by Di Summa *et al.* assume an optimal solution to the embedding problem exists where all pairwise distances come from a discrete set.[41] This allows the embedding problem to be framed as an integer linear program, with the objective to find an embedding with a metric as close as possible to the optimal ultrametic's.

3.3.2 Spanning Trees and Uniform Spanning Trees

As noted in Figure 3.4, there is an equivalence between dendrograms of datapoints and trees connecting the datapoints. Similarly, there is a correspondence between ultrametric spaces and tree representations of graph metric spaces. The entry point to tree representations of graphs is thus the spanning tree. To effectively represent a dataset, it is necessary to ask: for a given graph, are all trees created equal? Do some trees represent the graph better than others? Secondly, supposing that a tree structure is basically a good model for a graph, but individual spanning trees oversimplify graph structure, how may the abstract idea of tree-structuredness by formalized? These two questions are answered in the next two sections by reviewing research into minimum spanning trees, uniform spanning trees, and measures dealing with collections of trees.

Spanning Tree Properties A spanning tree for a connected graph $G = (V, E)$ is $T = (V, E_T)$ where $E_T \subset E$ and there exists exactly one path in E_T between any two nodes. A spanning tree may thus be obtained simply by removing an edge whenever a loop in the graph is encountered. In fact, any connected subgraph of G with $n - 1$ edges will be a spanning tree. It is positively dizzying, however, to consider how many spanning trees might exist for a large graph. Conveniently, Kirchhoff's theorem provides an insightful way of calculating this number using the graph Laplacian $\mathcal{L}$. If the set of eigenvalues of $\lambda(\mathcal{L}) = \{\lambda_0, \ldots, \lambda_{n-1}\}$, in increasing order, then the number of spanning trees is

$$\mid ST(G) \mid = \frac{1}{n} \prod_{i=1}^{n-1} \lambda_i$$

which is also equivalent to a cofactor of the Laplacian matrix. We refer to the set of all spanning trees of G as $ST(G)$. This quantification of the number of spanning trees is relevant for the use of effective resistance in chapter 6.

Minimum Spanning Tree If the graph is weighted, two types of spanning trees garner special attention: the minimum spanning tree (MST) and the maximal spanning tree. The MST of $G = (V, W)$ is

$$S_{\min} = \text{argmin}_{T \in ST(G)} \sum_{w \in W_T} w$$

and the maximal spanning tree is

$$S_{\max} = \text{argmax}_{T \in ST(G)} \sum_{w \in W_T} w$$

Kruskal's MST algorithm is a simple approach to finding the MST and can be used to find the maximal spanning tree by taking the inverse of the edge weights. Identifying an MST is a critical step in certain clustering algorithms. For instance, Yu *et al.* propose a method for agglomerative hierarchical clustering in social networks by finding the MST of a network.[149]

In the case when G is unweighted, the sum of minimum and maximum spanning trees' weights are identical, and so the "best" spanning tree can be defined differently. Elkin *et al.* propose a method to find a *low stretch* spanning tree.[50] The average stretch s of edges in a tree is

$$s(T) = \frac{1}{|E_T|} \sum_{(u,v) \in E} \frac{d_T(u, v)}{d(u, v)}$$

A tree with low stretch thus preserves distances locally by defining a cost function only using the distortion between nodes that are nearby in the original graph. Their low-stretch method has the advantage of both being fast

$$\mathcal{O}\left(|E| \log(|V|) + |V| \log^2(|V|)\right)$$

and providing a simple graph representation to speed up many expensive graph operations, such as solving systems of equations with respect to the graph Laplacian.

Sampling Uniform Spanning Trees While approaches exist to find spanning trees that are, in some sense, representative of a graph's structure, it is evident that the benefit of using only a tree to find local and global properties is limited. Instead of asking how well a single spanning tree can represent a graph, one may ask how well general spanning trees do the same task (covered in the next section, section 3.3.3)? Or more broadly, how well do all spanning trees do the same task? To approach these questions, it is necessary to know how to sample spanning trees and then to pool together the inferences from these trees. A large, finite sample of spanning trees may provide a richer picture than any individual tree of what is produced using the tree-like assumption of a graph.

There are a large number of algorithms for sampling spanning trees uniformly [11] that may be pooled into two categories. One group of methods use random walks,[8, 23][12] and another depends on effective resistance.[46] These approaches to uniformly sampling spanning trees have provided a convenient path to approximate methods when exact methods are intractable at a massive scale.

3.3.3 Collections of Trees

Our last summary of research in tree-based graph representations of graphs focuses on methods that make use of collections of metric trees. Considering each tree a weak expert, the large collection, or ensemble, of trees can provide a much richer picture of what results from an assumption that a graph is tree-like. This simple idea is the topic of chapter 6 and the inspiration to our approach to community detection in chapter 7. While there are many machine learning applications where collections of trees are used to estimate vertex labelings, in this section we primarily consider collections of trees to support metric embeddings. We delay discussing ways of using trees as graph models in applied machine learning problems

11. The *uniformly* is important because we must avoid sampling too many similar trees, or in the degenerate case, the same tree over and over again.

12. For instance, one nice, easily implementable algorithm for uniform sampling is to take a random walk on a graph and make a new graph's edge set any edge on the random walk that visits a node for the first time (other than the first node in the walk). When every node has been visited, the new graph is a tree.

until chapters 5, 6, and 7.

There are three primary ways that distinct trees may be combined in a metric space: tensor products, Cartesian products and strong products. For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, let $E_1 \Box E_2 = A_1 \cup A_2$ where

$$A_1 = \{(v_{1a}, v_{2a}), (v_{1b}, v_{2b}) \mid (v_{1a} = v_{1b}) \wedge (v_{2a}, v_{2b}) \in E_2)\}$$

and

$$A_2 = \{(v_{1a}, v_{2a}), (v_{1b}, v_{2b}) \mid (v_{2a} = v_{2b}) \wedge (v_{1a}, v_{1b}) \in E_1)$$

and let

$$E_1 \otimes E_2 = \{((v_{1a}, v_{2a}), (v_{1b}, v_{2b})) \mid ((v_{1a}, v_{1b}) \in E_1 \wedge (v_{2a}, v_{2b}) \in E_2)\}$$

The tensor,[13] Cartesian,[14] and strong products[15] are defined, respectively, as:

13. In order to contextualize the structure of these different product spaces, we also consider now the relationship between d_{T^A} and d_{T^B}, and $d_{T^A \otimes T^B}$, $d_{T^A \Box T^B}$ and $d_{T^A \boxtimes T^B}$. For any two path-connected vertices $v, w \in V(T^A \otimes T^B)$, the distance $d_{T^A \otimes T^B}(v, w)$ is the smallest number of edges that path connect v and w. Suppose that the shortest path between v^A to w^A in T^A is p^A with $l^A = |p^A|$ and the shortest path between v^B to w^B in T^B is p^B with $l^B = |p^B|$. If $l^A - l^B = 1$ (mod2) then there will be no path beginning at v and ending at w after k steps for some $k \in \mathbb{N}$. So it must be the case that $l^A - l^B = 0$ (mod2). In this case, the shortest path p from v to w is the following (without loss of generality assume that $l^A \geq l^B$): $(p_1^A, p_1^B), \ldots, (p_{l^B}^A, p_{l^B}^B), (p_{l^B+1}^A, p_{l^B-1}^B), (p_{l^B+2}^A, p_{l^B}^B), \ldots, (p_{l^A}^A, p_{l^B}^B)$. The length of this path is l^B so $d_{T^A \otimes T^B}(v, w) = l^A = \max(d_{T^A}(v^A, w^A), d_{T^B}(v^B, w^B))$.

14. We next consider the interpretation of $d_{T^A \Box T^B}$. We consider the same graph as in footnote 13, but we now consider the Cartesian instead of tensor product. To find $d_{T^A \Box T^B}(v, w)$ we again consider the paths p^A and p^B. Unlike the tensor product graph, for any two tuples of nodes, if there exist a path between the nodes in index one in T^A and a path between the nodes in index two in T^B, there is a path in $T^A \Box T^B$ as well. This path is the following: $(p_1^A, p_1^B), (p_2^A, p_1^B), \ldots, (p_{l^A}^A, p_1^B), (p_{l^A}^A, p_2^B), \ldots, (p_{l^A}^A, p_{l^B}^B)$. A symmetric path exists by first moving along p^B and then p^A, or some combination thereof. In either case, $d_{T^A \Box T^B}(v, w) = l^A + l^B = d_{T^A}(v^A, w^A) + d_{T^B}(v^B, w^B)$.

15. Again we consider the path between $v, w \in T^A \boxtimes T^B$ using $d_{T^A \boxtimes T^B}$ where $d_{T^A \boxtimes T^B}$ is the length of the shortest path in $T^A \boxtimes T^B$ between v and w. Similar to $T^A \Box T^B$, there is a path between v and w as long as p^A and p^B exist in T^A and T^B. Since edges in $T^A \boxtimes T^B$ may connect nodes whose entries differ in both index one and index two, the path between v and w in $T^A \boxtimes T^B$ is shorter than in $T^A \Box T^B$. Specifically, the path will be (without loss of generality assume that $l^A \geq l^B$): $(p_1^A, p_1^B), \ldots, (p_{l^B}^A, p_{l^B}^B), (p_{l^B+1}^A, p_{l^B}^B), \ldots, (p_{l^A}^A, p_{l^B}^B)$. This implies that $d_{T^A \boxtimes T^B}(v, w) = \max(d_{T^A}(v^A, w^A), d_{T^B}(v^B, w^B))$.

$$G_1 \otimes G_2 = (V_1 \times V_2, E_1 \otimes E_2)$$

$$G_1 \square G_2 = (V_1 \times V_2, E_1 \square E_2)$$

and

$$G_1 \boxtimes G_2 = (V_1 \times V_2, (E_1 \square E_2) \cup (E_1 \otimes E_2))$$

While there is extensive literature on the very general characteristics and hyperbolicity of $G_1 \otimes G_2$, $G_1 \square G_2$ or $G_1 \boxtimes G_2$ based on $\delta(G_1)$ and $\delta(G_2)$,[16] we have not found any accessible treatment of the hyperbolicity of these products in the simpler case when the graphs are trees. This particular scenario is of interest since trees may be considered the experts comprising an ensemble of weak learners, and seeing the ensemble as a product of trees provides a heuristic foundation for practically using collections of trees, as well as providing a graph model to study the properties of the collections. We demonstrate some properties of tree products now.

The Cartesian product is the most relevant graph product for applications because, as mentioned in footnotes 13, 14, and 15, for two vertices $(u, u), (v, v) \in V \times V$, $d_{\square}((u, u), (v, v)) = d_{T_1}(u, v) + d_{T_2}(u, v)$, i.e. a constant multiple of the average distance between $u, v \in V$ for both trees (more specifically, the sum of those distances). The other products - which boil down to defining the distance $d_{\otimes}(u, v)$ or $d_{\boxtimes}(u, v)$ in terms of the largest distance between u and v across all trees - is considering only the tree that introduces the largest amount of distortion between u, v, but not the case when borrowing information from a set of trees makes stronger estimates about pairwise relationships within the graph.

First, we demonstrate that the rate at which the surface area of balls grow for $T \square T$. Our aim is to show that they have the same exponential growth as standard hyperbolic graphs.

16. These characterizations are typically in terms of the worst case scenario, the least hyperbolic product possible. This makes sense since a general graph class is being considered. The worst case in the tree context, however, is quite different and generally more hyperbolic.

Remark. As the surface area of trees is being discussed, it is worth first reviewing the formulas for volume and surface area of a single tree T. Let $B(v,r) = \{u \in V \mid d_T(u,v) \leq r\}$ and let $SA(v,r) = \{u \in B(v,r) \mid \exists w \in V - B(v,r), \text{ s.t. } (u,w) \in E(T)\}$. These two quantities (volume and surface area) are related, as it is the case that $|B(v,r)| = \sum_{i=0}^{r} |SA(v,i)|$. In the binary tree T, under consideration for this example, $\forall v \in V(T),\ \deg(v) = 3$. For the center of the ball v, there are three neighbors. Those neighbors also have three neighbors, but one of them is v, so they have only two *new* neighbors. Because there are no loops, every subsequent element of the ball will similarly have two new neighbors, to be included in the next ball with radius increased by 1. Therefore there are $3(2^{r-1})$ nodes along the surface area of the ball $B(v,r)$ (for $r \geq 1$). What is the volume of that tree? For radius $r \geq 1$, the three descendants of the initial node v each have a total of $\sum_{i=1}^{r-1} 2^{i-1} = 2^r - 1$. All together, this means the ball $B(v,r)$ has volume $3(2^r - 1) + 3 = 3(2^r)$.

Theorem 1. Given a binary tree T, $\forall v \in V(T \square T)$, the cardinality of the set

$$\{u \in V \mid d_{\square}(u,v) = r\}$$

is defined by the integer sequence:

$$A(r) = 2B(r) + (r-1)B(2)B(r-2)$$

where $B(r)$ is the sequence of surface areas for balls in T (i.e. $B(r) = 3(2^{r-1})$) and with the convention that $A(0) = 1$ and $\forall k < 0,\ B(k) = 0$ (or in other words, $A(1) = 6$).

Proof. The reason for the specific form of the equation defining the surface area of balls in $T \square T$ may be understood using mathematical induction. For the Cartesian product, the induced subgraph of $T \square T$ using the vertex set $v \times V$ or $V \times v$ is itself a structure preserving representation of T. This will make it easier to extrapolate properties of subgraphs of $T \square T$. Given a center point v and a radius r, the surface area of a ball in $T \square T$ is defined as

$SA(B_{T\square T}(v,r))$. This collection of nodes may be partitioned according to their first index. Each $u \in V$ defines a distinct subgraph $u \times V$, and we define the component of the surface area on each of these subgraphs separately.

Consider two simple examples of these subgraphs: $\{u \in v \times V \mid d(u,v) = r\}$ and $\cup_{\{u \in V \times v | d(u,v)=r\}} \{w \in u \times V \mid d(u,w) = 0\}$. The former set defines the set of nodes along the surface of the radius-r ball in $v \times V$ and the latter set defines the set of nodes along the surface of the radius-r ball in $V \times v$. These two sets expand according to the rate that volume expands in a ball in T, and these comprise the first summand in theorem 1.

The second summand describes the collection of sets which are expanding within each subgraph $u \times V$ where $u \in B_{V \times v}(v,r)$. Defining these sets more specifically, for each integer $1 < s < r$, the set of elements $\{u \in B_{V \times v}(v,r) \mid d(u,v) = s\}$ each contain a subset of the nodes on the surface of $B_{T\square T}(v,r)$. These sets of nodes will be

$$A_s = \cup_{\{u \in B_{V \times v}(v,r) | d(u,v)=s\}} \{w \in u \times V \mid d(w,u) = r - s\}$$

A helpful property that makes enumerating the collection of these surface-lying points is that $\forall s$ s.t. $1 < s < r$, $|A_s| = f(s)$. That is, for any integer s, given a fixed r, the cardinality of A_s is the same. This may be easily demonstrated by multiplying the cardinalities of the set over which the union is taken together with the set of elements in the subgraph's one-dimensional ball. This means that it is sufficient to take the cardinality of A_s for a single s, say $s = 2$ and multiply by the number of possibly values s can assume, which is $r - 1$.

The cardinality of A_2 is the product of the number of elements that are of distance 2 away from v times the number of elements that are radius $r - 2$ away from those elements. All together this leaves us with $B(2)B(r-2)$. $\square$

When we consider the complexity of the first term, it just an integer multiple of $B(r)$. The complexity of the second term is an linearly increasing integer times a quadratic function of $B(r)$. So the rate at which this ball grows will remain exponential, preserving a relevant

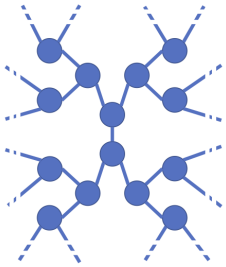


Figure 3.6: Binary tree used in Theorem 1

property found in hyperbolic spaces.

The surface area's cardinality is of interest because the exponentially growing surface area of spheres in hyperbolic space is one of the key properties that leads to their serving as embedding spaces for trees. The theorem is stated regarding the product of the binary tree T with itself for simplicity. It is easier to come up with an exact formula for surface area and volume for the case of a binary tree, though the product of any two trees will have volume of the same order, so long as for each tree's volume grows exponentially as a function of the radius around any vertex in the tree. This excludes, for instance, a path graph, which is a tree, but whose volume grows linearly as a function of radius.

A snapshot of one region of the T being considered is presented in figure 3.6.

The surface area may be a relevant feature of hyperbolic spaces, but it is not a proof of their hyperbolicity. The two ways of defining hyperbolicity we have covered are the triangle slimness of any three elements of the space and the 4PC. In the metric space which is the product of two trees, it is difficult to define the geodesics which comprise the sides of a triangle, since so many different equal-length paths exist between any two points. For that reason, we choose to use the 4PC to characterize $T \square T$'s hyperbolicity.

Before considering how the 4PC can be used for $T \square T$, we first consider just T. For a general tree graph, it is straightforward to see why the 4PC leads to a hyperbolicity of $\delta = 0$.

Let w, x, y, z be any four nodes in T. First suppose that, without loss of generality x lies on the shortest length path between w and y. Then $d(w, y) = d(w, x) + d(x, y)$. This allows

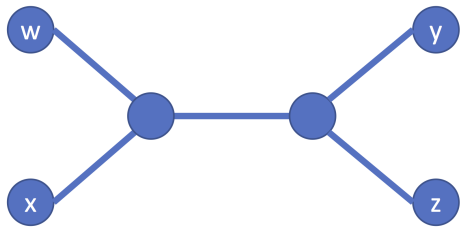


Figure 3.7: Simplest nontrivial layout of four nodes in a tree

a substitution into the 4PC to make:

$$d(w,x) + d(y,z) \leq \max\left(d(w,x) + d(x,y) + d(x,z), d(w,z) + d(x,y)\right) + 2\delta \tag{3.1}$$

$$\leq d(x,y) + \max\left(d(w,x) + d(x,z), d(w,z)\right) + 2\delta = d(x,y) + d(w,x) + d(x,z) + 2\delta \tag{3.2}$$

$$d(y,z) \leq d(x,y) + d(x,z) + 2\delta \tag{3.3}$$

The equality in line 3.2 is due to the triangle inequality. $\forall x, y, z$ line 3.3 will be true, also due to the triangle inequality. This means that the nontrivial case of evaluating the 4PC will occur when $\forall v_0 \in \{w, x, y, z\}$ it is the case that $v_0 \notin [v_1, v_2]$ where $v_1, v_2 \in \{w, x, y, z\} - \{v_0\}$.

This reduces evaluating the 4PC for a tree to the case when the induced subgraph of geodesics connecting the four points looks like the one in figure 3.7. In this case, the pairing of nodes that leads to the largest sums are $(w, y), (x, z)$ and $(w, z), (x, y)$, where their summed distances are both the same.[17] This means that the tree has $\delta = 0$, as there is zero difference between the two maximum summed distances.

With this in hand, we next turn to the case of $T \square T$. We first show the the δ-hyperbolicity of $T \square T$ is bounded below, and then show a particular restriction of $T \square T$ to a smaller metric space has stronger hyperbolicity.

Theorem 2. *$T \square T$ is δ-hyperbolic with $\delta \geq 2\text{diam}(T)$.*

17. There are three possible ways of splitting $\{w, x, y, z\}$ into two pairs: $(w, y), (x, z)$, $(w, z), (x, y)$, and $(w, x), (y, z)$. In the first two cases, the summed distances are $d(w, y) + d(x, z) = d(w, z) + d(x, y)$. In the last case, the summed distance is $d(w, x) + d(y, z)$.

Proof. Let $u, v \in V$ be the two vertices for which $d_T(u,v) = \text{diam}(T)$. Then consider the union of all vertices lying on the shortest paths between the vertices $(u,u),(v,v) \in V \times V$. Note that the length of the shortest path between $(u,u),(v,v)$ in $T\square T$ is $2d_T(u,v)$. Define the set of vertices along shortest length path p_{uv} between u and v on T as $V(p_{uv})$ and consider $w \in V(p_{uv}) \times V(p_{uv})$. By definition of the box product's distance metric, $d_{T\square T}(u,w) = 2d_T(u,w)$ and $d_{T\square T}(w,v) = 2d_T(w,v)$, and so $d_{T\square T}(u,w) + d_{T\square T}(w,v) = 2d_T(u,w) + 2d_T(w,v) = 2d_T(u,v)$. Therefore, every one of the elements in $V(p_{uv}) \times V(p_{uv})$ will lie along a shortest length path between (u,u) and (v,v). The subgraph induced by $V(p_{uv}) \times V(p_{uv})$ is a grid. This is because it is the box product of two identical path graphs between u and v. We have already shown in section 3.1.2 that the hyperbolicity of a grid is equal to its diameter. Therefore δ is lower bounded by $\text{diam}(T\square T) = 2\text{diam}(T)$. □

Verbeek *et al.* demonstrate that any graph with quasicyclicity m has a multiplicative distortion of $\mathcal{O}\left(\frac{m}{\log(n)}\right)$ in hyperbolic space.[136] Theorem 3.5 in their paper, which discusses the distortion of quasi-cyclic graphs specific deals with the case of (α,β)-quasi-cyclic graphs. This variety of quasi-cyclic graphs are cycle graphs where shortcuts[18] along the cycle are permitted, specifically shortcuts that do not reduce the distance by more than α when shortcuts are longer than β. When shortcuts are longer than β, the distortion is not constrained by α. A grid is an example of a quasicyclic graph with m equal to the diameter. Therefore the multiplicative distortion of embedding $T\square T$ in a hyperbolic space is $O(1)$.

Multiplicative distortion does not indicate a space is "very" hyperbolic (relative to a tree, at least), but the metric space under consideration, $T\square T$, is not actually the metric space that will be used for assessing pairwise similarities between nodes or global graph properties. The only distances that will ever be measured using a combination of trees are in fact from the subset of $V \times V$ equal to $(V \times V)_{\text{diag}} = \{(v,v) \mid v \in V\}$. The whole purpose of considering constructions like products of trees is to gain the ability to consider properties like distance

18. A shortcut is a path between two nodes whose length is shorter than the path between the nodes along the cycle.

between elements of $(V \times V)_{\text{diag}}$. For example, for two trees $T_1, T_2 \subset G$, when the graph $T_1 \square T_2$ is considered, $\forall u, v, w, x \in V$ where $u \neq v$ and $w \neq x$, the distance $d((u, v), (w, x))$ has a meaningful interpretation in $T_1 \square T_2$, but not in the original graph G. After all - what is the relevance of knowing the value of $d((u, v), (w, x))$ when investigating properties of the relationships between u, v, w, x?

For this reason, we consider the smallest δ for which the 4PC holds when we only consider $w, x, y, z \in (V \times V)_{\text{diag}}$. The answer is that this δ will depend on the properties of T_1 and T_2.

Let $w, x, y, z \in (V \times V)_{\text{diag}}$ be the four points for use in the 4PC. As oppose to the assumption in theorem 1, where the box product consists of a tree with itself, here two distinct trees are considered. We work with the three considered pairwise sums:

$$\{d_{T\square U}(w, x) + d_{T\square U}(y, z), d_{T\square U}(w, z) + d_{T\square U}(x, y), d_{T\square U}(w, y) + d_{T\square U}(x, z)\}$$

Any four points in a graph may be depicted as in figure 3.1.2. In the case of a graph which is a tree, the relationships between any four points may be summarized using one of the models in figure 3.7. This means that, though there will inevitably be some differences between T and U, they boil down to very similar representations for four-way relationships. There are three nontrivial possibilities - either they are both of the form (a), both of the form (b), or one is of the form (a) and one is of the form (b). We consider the first and third possibility the second results from symmetry.

Suppose that $d_T(w, x) = l_1^T + l_2^T$, $d_T(w, y) = l_1^T + l_3^T + l_4^T$, etc., and $d_U(w, x) = l_1^U + l_2^U$, $d_U(w, y) = l_1^U + l_3^U + l_4^U$, etc. Then the pairwise sums are:[19]

19. To make notation more compact, the use of $l_i^{T+U} = l_i^T + l_i^U$.

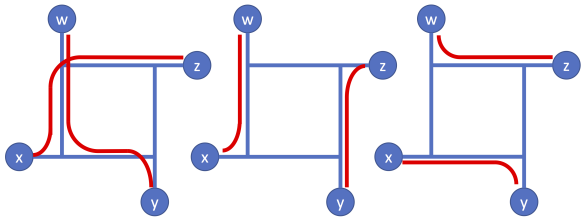
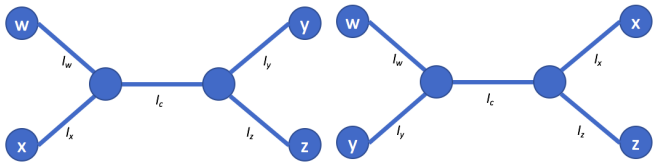


Figure 3.8: The three pair sums used to evaluate the four point condition

		Version 1: Both of form (a)
1)	$wx + yz$	$(l_w^{T+U} + l_x^{T+U}) + (l_y^{T+U} + l_z^{T+U})$
2)	$wy + xz$	$(l_w^{T+U} + l_c^{T+U} + l_y^{T+U}) + (l_x^{T+U} + l_c^{T+U} + l_z^{T+U})$
3)	$wz + xy$	$(l_w^{T+U} + l_c^{T+U} + l_z^{T+U}) + (l_x^{T+U} + l_c^{T+U} + l_y^{T+U})$
		Version 2: One of form (a), one of form (b)
1)	$wx + yz$	$(l_w^{T+U} + l_c^{U} + l_x^{T+U}) + (l_y^{T+U} + l_c^{U} + l_z^{T+U})$
2)	$wy + xz$	$(l_w^{T+U} + l_c^{T} + l_y^{T+U}) + (l_x^{T+U} + l_c^{T} + l_z^{T+U})$
3)	$wz + xy$	$(l_w^{T+U} + l_c^{T+U} + l_z^{T+U}) + (l_x^{T+U} + l_c^{T+U} + l_y^{T+U})$

In version 1, the two largest pair sums are lines 2 and 3 which are equal. By contrast, in version 2, line 3 is largest, with line 1 or 2 being next largest depending on the comparison between l_c^T and l_c^U. The hyperbolicity will boil down then to $l_c^{T+U} - \max(l_c^T, l_c^U) = \min(l_c^T, l_c^U)$. This definition of the hyperbolicity of the space may be interpreted in terms of the properties of T and U. If T and U are very similar, then there are few examples of four points (w, x, y, z) which are described very differently in T versus U. For instance, if w, x and y, z are each a couple of close together nodes, the distance between the couples is not going to be very large in T, unless it is also large in U. For example, if in T w, x are close together and y, z are close together, and in U, if we find that w, y and x, z are similarly close together, then the four points are likely to belong to a cluster.

We use the above theorems to extrapolate what the hyperbolicity of a product graph is with more terms, such as $T^1 \otimes T^2 \otimes \cdots \otimes T^k$,$T^1 \square T^2 \square \ldots \square T^k$, or $T^1 \boxtimes T^2 \boxtimes \cdots \boxtimes T^k$. Considering the discussion of distances in footnotes 13, 14, and 15, these finite products are relevant when we consider estimating pairwise distances using a function of the pairwise



(a) (b)

Figure 3.9: Two simplified models for the layout of four vertices in a tree graph

distances found in a finite sample of spanning trees; whereas for the Cartesian or strong product, this distance would be the maximum across all trees, the tensor product corresponds to the average distance.

The hyperbolicity of graphs is weaker the longer the chain of tree products that it involves, but hyperbolicity is preserved at each stage. It has been proven that an upper bound of $\delta(G \times T) \leq \text{diam}(G) + \frac{1}{2}$ exists for the strong product.[24] On the one hand this result demonstrates that hyperbolicity is not lost in the strong product operation, but on the other hand, the result only provides a very conservative bound on how un-hyperbolic a product can be, a bound that grows very quickly for a finite number of products. We aim to produce a tighter result where it is known that G is a product of a finite number of trees, but have not been able to produce such an argument as of yet. We will discuss some motivating directions in the conclusion in chapter 9.

While products of trees provide a convenient geometric model for how to combine information from different tree representations of a graph, a practical approach in machine learning is to consider collections or distributions of trees, not products. In this case, for instance, a distance between $v, w \in G$ is not measured as the distance from $(v, v, \ldots, v)$ to $(w, w, \ldots, w)$, but as $\int d_T(v, w) d\mu_T$. Two primary results in this area deal with approximating metric spaces using tree distributions: Bartal and Fakcharoenphol use hierarchically well-separated tree metric spaces to probabilistically approximate arbitrary metrics, with the quality of the approximation characterized as a function of the number of tree metrics that comprise it. Bartal's result is sharpened by Charikar to apply even in the case when there is a "small" number of trees. In chapter 5 of this thesis we consider a number of ways that tree

collections or distributions can simplify challenging machine learning problems, and ways that tree collections' descriptive power overlaps with what is produced from MRA.

3.4 Summary

The topics covered in this chapter serve as a gentle introduction to hyperbolic geometry, hyperbolic metric spaces, and the characterization of tree-likeness for real-world datasets. Long valued as a foundation for relativity theory, the renaissance of hyperbolic metric spaces takes root in approximating tree-like graphs. Whether by means of embeddings in discrete or non-discrete metric spaces, hyperbolicity is an assumption that can guide approximate representations for large-scale analysis.

Our focus on products of trees is due to the close relationship between many MRA frameworks and tree approximations, as well as the use of collections of trees to boost performance of otherwise overly-reductionist approaches. By combining the scalability of tree-based inference together the descriptive power of low-distortion metric embeddings, products of trees provide an engine for analyzing large-scale network data. Additional argumentation from a regularization perspective for using hyperbolic representations for applied problems is found in section 5.4.3.

CHAPTER 4

OVERVIEW OF CONTEMPORARY PROBLEMS IN NETWORK ANALYSIS

To present methods that exploit graph structure, some foundational areas of network analysis are first introduced[1] to show the way in which our work delivers a novel contribution. In section 4.1 we discuss how graphs can be used for inference. When signals are defined on a portion of the vertices and unknown on the rest, i.e. semi-supervised learning (SSL), the edges of a graph may be used to infer the value of the signal at the unlabeled positions. The primary assumption in using a graph to estimate signals on a graph are that signals will be smooth with respect to the graph structure. The same technique that may be used to estimate smooth functions on graphs is also the foundation of Gaussian Process regression. SSL techniques provide a valuable framework for approaching problems with a lot of unknown information. All the more so today, when data collection has never been easier, the labeling of data remains expensive, we confront a demand for methods that function in this demanding context as well. In the setting when no labeling information is present, i.e. unsupervised learning, a common challenge is to produce a labeling of nodes that respects some division of *community* structure. A community in a graph, corresponding to a cluster in $\mathbb{R}^n$, is a set of nodes where the interconnectedness among the nodes is much stronger than the connections between the elements of the set of nodes and the elements of the set's complement. The task of *community detection* is to discover those communities without any features other than the vertex and edge set. The novel contributions we present to community detection are contained in chapters 6 and 7.

1. This chapter's title of "network analysis" was chosen in lieu of something along the lines of "graph analysis" to distinguish our work from the abstract field of graph theory. Our aim is to address applied problems in real-world graphs using techniques arising from multiresolution analysis. In our discussion, we will refer to graphs as models for data, whereas networks are the realizations of those models. A social network is comprised of data consisting of entities and the connections between them. The corresponding graph is a model of these interactions, either by directly copying it or by abstracting some of the information contained in it. The difference is subtle and we apologize for any confusion in this matter.

Notation All graphs $G = (V, E)$ in this chapter have $n = |V|$ and $m = |E|$, and are undirected, unless specified otherwise. The vector $d_G = (d_v)_{v \in V} \in \mathbb{N}^{|V|}$ is the vector of nodes' degrees. For convenience, a vertex or edge set of a graph G may also be denoted $V(G)$ or $E(G)$, respectively. Similarly, a graph's weight matrix is denoted W_G or $W(G)$. The $n \times n$ adjacency matrix of a graph is denoted A_G and the graph Laplacian of the graph is $L_G = \text{diag}(d_G) - A_G$.

4.1 Semi-supervised learning on graphs

While graphs are not the only model for semi-supervised learning, graph-based SSL remains an exciting area of research due to the way that graphs are able to represent a complete dependence structure among datapoints. Given a dataset $D = \{x_i \mid 1 \leq i \leq n\}$ and a positive semidefinite function $f : V \times V \to \mathbb{R}^+$ measuring the relative similarity between datapoints, a weight matrix W_G may be defined such that $W_G(i, j) = f(v_i, v_j)$. A typical way of defining these similarity-based weights from D is by choosing a kernel k combined with a thresholding function. For instance, the l-parametrized radial basis function (RBF) kernel may be used to calculate similarities between observations, and then a k-nearest neighbor (KNN) or τ-parametrized rectified linear unit (ReLU) may be used for thresholding. The thresholding step is critical for large datasets in order to work with a sparse graph. For two datapoints d_i, d_j this would correspond to

$$t_\tau \circ k_{\text{RBF}}(d_i, d_j) = t_\tau \left(e^{-\frac{\|d_i - d_j\|}{l}} \right) = e^{-\frac{\|d_i - d_j\|}{l}} \mathbb{I}\left(\tau \leq e^{-\frac{\|d_i - d_j\|}{l}} \right)$$

or alternatively,

$$t_K = \mathbb{I}\left(j \in \text{nbrs}_k(i)\right) \Longrightarrow t_K \circ k_{\text{RBF}}(d_i, d_j) = e^{-\frac{\|d_i - d_j\|}{l}} \mathbb{I}\left(j \in \text{nbrs}_k(i) \wedge i \in \text{nbrs}_k(j)\right)$$

Given a matrix weighting, the SSL problem is cast with a partition of $D = \{x_i \mid i \in$

$U\} \cup \{x_i \mid i \in L\}$ where $U \cup L = \{i \mid 1 \leq i \leq n\}$ and $U \cap L = \emptyset$. The sets U and L identify unlabeled and labeled observations, respectively. So, in full, the problem starts with D, y where $|D| = |L| + |U| = n$ and $|y| = |L|$. Our goal is to estimate the function y at the vertices of U. We construct a model of the dataset, make assumptions about the types of functions that are most favorable, and provide estimates at the unlabeled positions. This type of learning is called *transductive learning* - using information from one collection of objects to teach us about another set of objects, both of which are present at the training stage. This is contrasted with *inductive learning* where a function is learned using information available at the training stage, but testing takes place on data unavailable at training. The fact that graph-based SSL is transductive might appear as a weak point for this approach - supervised learning, for example, typically learns a function using training data and may then be applied to any future test data. The supervised learning framework for our setting, though, would waste a (possibly) huge amount of information. The supervised learning approach would use $\{x_i \mid i \in L\}, y$ for training and ignore $\{x_i \mid i \in U\}$. When $|L| << |U|$ this decision may be catastrophic. Nonetheless, there is undoubtedly a tradeoff between transductive semi-supervised learning and inductive supervised learning, depending on contextual factors. It must be noted that since its initial introduction, inductive forms of semi-supervised learning have been developed, evidencing the necessary contribution of an SSL framework in many problems where labeled data is expensive.

4.1.1 Estimating Unknown Labels Using Smooth Functions on Graphs

In the seminal paper introducing graph-based SSL, Zhu *et al.* provide a clear algorithm for estimating labels on $(y_i)_{i \in U}$ using the regularization properties of the graph Laplacian $\mathcal{L}$.[156] The first step is to choose a metric that the estimated functions should satisfy. In the Euclidean setting, this would be analogous to choosing a function interpolation method for estimating a function where its values are not known. Invariably, the interpolation method in that context would penalize how much a function value changes from one location's

known function value to a neighbor's. The more complicated structure of a graph defines interpolation by penalizing sharp changes of the estimated function with respect to the graph's edge set. One metric for defining this penalty is the quadratic energy function:

$$\mathcal{E}(f) = \sum_{(i,j)\in E} w_{ij}(f_i - f_j)^2$$

The resulting objective function is

$$\hat{f} = \text{argmin}_{f\in\mathbb{R}^{|V|}, f|P_L=y}\mathcal{E}(f) \tag{4.1}$$

The unique minimizer of this objective is also the function for which $\mathcal{L}f = 0$. The solution to this system of equations is the function

$$f_U = \mathcal{L}_{UU}^{\dagger}\mathcal{L}_{UL}f_L$$

The estimation of the unlabeled portion of f using this formula connects the solution to this problem together with the solutions to several related problems. For instance, if random variables X, Y are jointly Gaussian, with mean zero and variance matrix Σ, the distribution of the random variable $X \mid Y$ is also Gaussian with mean

$$\mu_{X|Y=y} = \Sigma_{XY}\Sigma_{YY}^{\frac{-1}{y}}$$

Using the Schur complement's expression of $T = \Sigma^{-1}$

$$\mu_{X|Y=y} = \Sigma_{XY}\Sigma_{YY}^{\frac{-1}{y}}$$

$$= -T_{XX}^{-1}T_{XY}(T_{YY} - T_{YX}T_{XX}^{-1}T_{XY})^{-1}\left((T_{YY} - T_{YX}T_{XX}^{-1}T_{XY})^{-1}\right)^{-1}y = -T_{XX}^{-1}T_{XY}y$$

The kernel matrix, as a measure of similarity, bears close resemblance to the variance of a

dataset, showing that this expression of the mean of the conditional Gaussian is expressed exactly like the harmonic function on G.

Alternatively, for a Markov process whose time-independent transition matrix has entries $P_{ij} = \frac{-\mathcal{L}_{ij}}{\sum_{k \neq i} \mathcal{L}_{ik}} \mathbb{I}(i \neq j)$, initialized with a random vector $f^{(0)} \in \mathbb{R}^{+|V|}$, defining the transition as $f_U^{(t+1)} = (Pf^{(t)})_U$ and $f_L^{(t+1)} = f_L$, then, as in the case of the conditional Gaussian RV, we have[2]

$$f_U^{(t)} \rightarrow^P P_{UL} P_{LL}^{-1} f_L$$

In a subsequent article, Zhu *et al.* describe their estimated harmonic function as the mean of a Gaussian random field, where the field is defined by the density $\text{Prob}_\beta(f) \propto e^{-\beta \mathcal{E}(f)}$.[155] Lastly, the estimated harmonic function may be seen as the solution to a kernel-learning problem. Specifically:[3]

$$\hat{f} = \text{argmin}_{f \in \mathbb{R}^{|L|+|U|}} \sum_{i \in L} l(f_i - y_i) + \lambda f^T K^{-1} f \tag{4.2}$$

The pseudoinverse of the graph Laplacian is a positive semidefinite matrix and may be considered a kernel matrix for the graph's nodes. The means the clamped version of graph-based SSL (where $\forall i \in L, \hat{f}_i = y_i$) makes the formulations in lines 4.1 and 4.2 identical. If, however, the kernel considered is not the general graph kernel $\mathcal{L}^\dagger$ but simply the kernel whose pairwise similarities have defined the graph's edge weights, then the two formulations are not the same. In fact, by formulating the SSL problem in terms of learning a function in the reproducing kernel Hilbert space of K, the ML practitioner has more control over how function smoothness is described. Through transformations of the kernel matrix's eigenspectrum, the feature space in which the regularization of function smoothness takes place may be chosen more deliberately.[80]

2. Normalizing each row of P using the degrees along $\mathcal{L}$'s diagonal will not change any results, since both sides of the iterative update contain the same normalization.

3. l refers to a loss function.

4.1.2 Limitations

Modeling unknown labels as harmonic functions is an insightful, simple assumption to guide inference. In the nearly two decades since its introduction, graph-based SSL has faced a broad set of challenges. First, and simplest, is the scalability of the method. Either through solving a system of equations or inverting a graph Laplacian, finding the harmonic solution for the combination of known and unknown labels is an operation that, in the worst case, scales as $\mathcal{O}(n^3)$, though typically executes in quadratic time. Furthermore optimizing hyperparameters for the kernels used to define edge weights adds an expensive optimization search on top of the main problem, requiring additional matrix inverses and the computation of determinants. Second, the harmonic function finds a function that is smooth with respect to an undirected graph's structure. In order to address a broader class of graph-based problems, we would need to consider cases when the adjacency matrix is not symmetric. Third, when the unlabeled data consists of a massive amount of high dimensional data, the limit of the squared energy loss function converges to a weighted gradient penalty term. Depending on how the observations are distributed, this limit function may be minimized with graph labelings that are arbitrarily close to zero at unlabeled indices. While this behavior is not generally observed in practice, it suggests limitations to applying graph-based SSL in the "big data" context.[107] Additionally, it was shown that, while the SSL estimator on unknown labels (assuming the labeling is clamped at known labels) is a consistent estimator in the presence of unlimited data, this is only the case when the rate of increase of unlabeled data is slower relative to labeled data.[45] One workaround to the ill-posedness described here is to maintain the relative proportion of labeled and unlabeled data $\frac{l}{u+l}$ even while $l+u$ grows.[139] This approach, though, does not seem to describe typical data analysis situations, where it becomes more difficult to obtain labeled data as the size of a dataset grows. An alternative approach simplifies the regularization of the estimated labeling by considering smoothness constraints using only a small subset of the eigenvectors (the smoothest ones).[79] While this avoids some of the degeneracy issues in large datasets that have been brought up,

it likely oversmooths datasets with primarily local structure. Still, some of these theoretical limitations of graph-based SSL have led to research in alternative models for semi-supervised learning, such as the use of neural networks[144] or graph filtering.[27, 96]

Modeling Limitations

Zhu *et al.* point out that the smoothness assumption on the graph's labeling can be limiting. By using the graph Laplacian for regularization, the lowest frequency eigenvectors are weighted most strongly, leading to the estimation of graph labelings that are primarily smooth based on the first few eigenvectors. Detecting local structure will be difficult in these circumstances. One suggestion, they propose, is the use of techniques from compressed sensing to represent function labelings, letting $\hat{f} = Pv$ where v is a sparse vector having support on low and high frequency eigenvectors.[154]

Alternative Graph Types

The difficulty of doing SSL on directed graphs relative to undirected graphs lies primarily in how to define the proper smoothness metric. Whereas in undirected graphs, two nodes' edge-connectness increases the likelihood of their being placed in the same class, the (possibly) hierarchical structure of a directed graphs, means it is important to consider the cascade of relationships a single node's class effects. For this reason, it is necessary to impose a smoothness that respects the directional relationships in a directed graph. Two common choices are to measure *co-linkage* in terms of shared parents or in terms of shared children. The smoothness function in equation 4.3 defined by Zhou *et al.* takes into account the similarity of two nodes' common parents to decide how much to weigh differences in their class labels.[152][4]

$$S(f) = \sum_{u,v \in V} \sum_{p \in \Pi(\{u,v\})} \frac{w_{pu} w_{pv}}{q_p} \left(\frac{f_u}{\sqrt{p_u}} - \frac{f_v}{\sqrt{p_v}} \right)^2 \tag{4.3}$$

4. A node v's in-degree is denoted p_v and its out-degree is denoted q_v. The set of nodes, all of which have a directed edge connected to the nodes in W, the parent nodes of W, are denoted

$$\Pi(V) = \cap_{w \in W} \{u \mid \overrightarrow{(u,v)}\}$$

Scalability The approach of Zheng *et al.* was to recognize that "not all data is created equal," and that comparable performance may be obtained in graph-based SSL with only a fraction of the unlabeled data.[151] Ebert *et al.* went further, in showing that if they combine graph-based SSL with additional active learning techniques, this modified SSL procedure is boosted beyond the performance of the traditional graph-based SSL alone.[48] More recently, the emerging field of graph embeddings has provided a helpful outlet for inductive learning. In the *Planetoid* framework, every node is embedding in a metric space to minimize the loss both in terms of its local graph distances and to node's class label.[148] Two approaches similar to our own are the use of spectral graph wavelets or spline-like wavelets[49] to enable the scalability of the traditionally spectral-based Laplacian regularization.

4.2 Community Detection

4.2.1 Problem Context

Network structure provides a convenient model for interactions in a very general sense. Practically any setting in which the relatedness between elements qua metric A is assumed to determine their relatedness qua metric B may be modeled by a graph. All the more so in today's "big data" setting, data is rarely observed in neatly organized $m \times n$ datasets, and must be modeled with a more general structure.[59] As noted already in section 4.1, it is often convenient to use a graph model even when a dataset is structured in order to use a graph as a way of summarizing local and global vertex interactions. The relevance of network models to such a wide variety of domains (i.e. computer science, genomics, sociology, neuroscience, finance) has undoubtedly motivated a wealth of research in *community detection* (CD). In analogy to data analysis in Euclidean space, CD is a method for clustering graph-structured data. More precisely, CD is the pursuit of subsets of nodes for which the level of interconnectedness among nodes in the subset is greater than the level of connectedness between the elements of the subset and the elements of its complement. In order to proceed, a few terms

must be defined: *community*, *connectedness*, and *greater than*.

In the words of Fortunato, typical communities are "dense subgraphs which are well separated from each other."[59] This seemingly noncontroversial definition leaves undefined what "denseness" and "well separatedness" mean, but it also addresses cleaner settings where communities are equivalent to a partition, or *hard clustering*. This does not pose an issue for thinking of communities as a proxy for a binary or multi-class classifier, but in more real-world network settings, community structure is unlikely to be this one-dimensional. Fortunato notes that, in reality, communities may be overlapping (requiring *soft clustering*), hierarchical (requiring a dendrogram-construction method), have differing levels of inward connectedness (requiring scoring to be adaptive based on context), have unbalanced sizes (which can cause issues of resolution limits), and combinations thereof. At the end of the day, some structural assumptions must be introduced to make CD tractable.

In the previous section on graph-based SSL a small portion of the nodes have a label or class associated with them, while the majority of nodes' edge connections are known, but labels are not. CD is a wholly unsupervised task, with no node labels being known or considered at the onset. Especially when real-world large-scale social networks are no longer small enough (as was the case for Zachary's Karate dataset) to visually evaluate community quality, methods that are completely unsupervised are endowed with renewed importance. Generally, CD is an unsupervised task whose output is the community labeling. It may also function as a pre-processing step in another learning algorithm, essentially serving as a graph feature extraction tool.[5]

5. The relevance of CD to a plethora of fields has led to a number of distinct forms of CD, sometimes specializing in a particular graph type, or simply holding different assumptions. This explosion of interest and research has actually motivated a focus on the CD of CD research, identifying methods that are more closely related, groups of researchers who tend to cite each other, and the most influential ideas and researchers.[39, 81] Surely the effectiveness of a technique for investigation is proven when practitioners use it to investigate themselves!

4.2.2 Two General Approaches to Defining Communities

Scoring Function-based

One popular approach is to define a scoring function s that will capture a certain variety of community structure. The breadth of scoring functions one might take is overwhelming and we consider two of the most dominant examples: *conductance* (C) and *modularity*(Q). The definition of conductance is presented in equation 4.4 and the definition of modularity is presented in equation 4.5. In the equations that follow, the adjacency matrix is A, c is the community labeling (with the label of node v being c_v), c_{unique} is the vector of distinct elements of c, and for a vertex set $S \subset V$, $s(V, c) = \frac{1}{2} \sum_{v \in V} \sum_{w \neq v} A(c_v = c_w)$.

$$C(c) = \frac{1}{|c_{\text{unique}}|} \sum_{\kappa \in c_{\text{unique}}} \frac{\sum_{u,v \in V:\ c_u = \kappa,\ c_v \neq \kappa} A_{uv}}{\min\left(s\left(\{u \mid c_u = \kappa\}\right),\ s\left(\{u \mid c_u \neq \kappa\}\right)\right)} \tag{4.4}$$

$$Q(c) = \frac{1}{2s(V)} \sum_{\kappa \in c_{\text{unique}}} \left(\sum_{u,v:\ c_u = c_v = \kappa} A_{uv} - \frac{k_u k_v}{s(V)} \right) \tag{4.5}$$

The conductance C is a straightforward metric to understand, it is the number of edges that need to be cut in order to separate this community of nodes from the graph relative to (assuming that the former term in the min is smaller) the overall connectedness of the community. A slightly deeper interpretation of the conductance is a graph version of the surface area to volume ratio. The surface area of an object is the total amount of connection the object has to the outside world - this is the cut size. The volume is the amount of interconnectedness that exists in the object - this is the normalization term, calculating the total weight of the edges internal to the community. In Euclidean space, the surface area to volume ratio is minimized by having as few corners as possible - for shapes with equal volume, the triangle has the largest surface area to volume ratio, while the circle has smallest. In the graph context, a "corner" will be a node that is misplaced - adding more outward connections to the community than its inward connections. So the community that

minimizes conductance may be thought of as a the graph version of a circle. In the electric network setting, conductance has a meaning that does not correspond immediately with graph conductance. The electrical network version of conductance, and its inverse resistance, will be used heavily in chapter 6. In the overall summary of a community labeling using conductance, the conductance of each community is typically calculated with the average or maximum conductance of the communities being the quantity to be minimized.

The modularity for a single community is the average difference between the number of edges that are observed from the number of edges that are expected. By considering the sum for each community separately, we obtain a big picture description of how much of the edges associated with the community's nodes are actually present inside of the community. Other definitions of modularity exist, but this is the most well known version, introduced to the ML community in 2006.[108]

Some methods will aim to directly optimize C, Q or other scores similar to or derived from them. The CD problem then amounts to defining an appropriate scoring function and then minimizing an objective function that depends on it. Others, such as our own, use the scores as a *post facto* way of ranking communities. In chapter 6, for instance, we use a community detection method which is focused on capturing multiresolution structure in the graph.[51] The wavelets that are defined on the graph are sparse, and their support generally corresponds to a group of related nodes. By using scoring functions on all the candidate communities, we are able to rank which are most likely to be legitimate communities, and also compare these scores to null distributions of community scores. This is also the approach taken by many algorithms to determine a stopping condition or selection criteria, such as seed set expansion,[145] and agglomerative or divisive hierarchical clustering.[65, 81] Related to scoring function-based CD, hierarchical and agglomerative clusterings may also be considered a category unto themselves.[26] A particularly relevant agglomerative method is the so-called *Louvain method*.[20][6] This hierarchical clustering method merges nodes based on

6. The Université catholique de Louvain is the institution from which the authors hail.

maximizing the local modularity of each ensuing cluster. The merged nodes are then deemed "supernodes," becoming a weighted combination of the components for the algorithm's next iteration of merging.

Probablistic Model-based

One probabilistic approach to CD is the *stochastic block model* (SBM). Instead of estimating a single community labeling function using the graph structure which is observed as data, the SBM asks which probabilistic graph model is most likely to have generated the edge set we observe. Some assumptions are introduced in order to make the problem tractable:

1. Every edge occurring may be modeled as a Bernoulli random variable,
2. The probability of any two edges existing is independent,
3. The graph observed is an Erdös-Rényi process for a particular matrix $P(\Theta)$, where Θ is an equal sized matrix of Bernoulli random variable parameters, and
4. The parameter of the random variable is a particular fixed value for every community, and for each community-community edge variety.

By basing its assumptions so strongly in a particular probabilistic model, SBM is an attractive area for developing theory to guarantee how a SBM will work in a variety of situations. There have also been developments to specify SBMs that weigh edges connected to higher degree nodes as heavier than those connected to lower degree nodes (the degree-corrected SBM),[60] or to assume that the matrix $P(\Theta)$ has hierarchical structure, with communities inside of communities.[113] Recently, some limitations of SBMs have been pointed out, such as the varieties of sparse graphs where weak or strong recovery of communities is not possible with SBMs.[3] The massive and sparse graph setting is our primary area of interest.

An alternative approach with a probabilistic foundation is the category of random walk-based CD algorithms, the most well-known versions of which are the PageRank algorithm[111]

and the Infomap algorithm.[118] Making structural assumptions similar to that of the scoring functions, a random walk is assumed to be more likely to stay inside a community than to leave it, due to the community's cut size relative to the number of internal edges. A related idea is to consider the diffusion from a node, analogous to a notion of information propagation in the network, leading CD to be determined by a diffusion process.[39] The connection between random walks and CD is the basis of the work presented in chapter 6.

4.2.3 Model Evaluation

As with all unsupervised methods, model evaluations must be taken with a grain of salt. After all - which "ground truth" community labeling are we aiming to achieve? There are presumably many legitimate community labelings for any given graph based on different node features. Nonetheless, ground truths, especially in the case of synthetic graphs, have a meaningful interpretation as the labeling of nodes in the model which generated the data. We present two methods for the evaluation of an estimated community labeling relative to a ground truth. Since the number of communities is unknown at test time, a metric that is flexible enough to compare two labelings with a different number of unique labels must be considered. One simple choice is the Rand index. By focusing only on pairwise node labelings between the community labelings, the issue of matching two labelings that might be quite different is avoided. The exact formula for a ground truth community labeling c and an estimated labeling $\hat{c}$ is

$$\mathrm{RSI}_c(\hat{c}) = \frac{\sum_{u,v \in V} \mathbb{I}\left((c_u = c_v) \cap (\hat{c}_u = \hat{c}_v)\right)}{\binom{n}{2}}$$

The RSI is always between 0 and 1, and a value closer to 1 indicates that two clusterings are more similar. The Rand index may also be thought of as a generalization of the confusion matrix for CD. If c and $\hat{c}$ have the same number of unique labels, then for every possible pair of nodes, the entries along the diagonal of the confusion matrix are what is included in the

numerator. For cases when the number of unique labels differs between c and $\hat{c}$ the Rand index is calculating something a bit different. In the case when two labelings are generated randomly, the Rand index's expected value is not constant. For that reason, the adjusted Rand index (ARI) is typically used, where the null distribution of Rand index is assumed to be a general hypergeometric random variable. The ARI has an expected value of 0 when two labelings are generated randomly.

Another CD metric is the normalized mutual information. If we model the two clusterings as two random variables $\mathcal{X}^c$ and $\mathcal{X}^{\hat{c}}$, then the amount of dependency between $\mathcal{X}^c$ and $\mathcal{X}^{\hat{c}}$, or the amount we know about one random variable by knowing the other,[7] depends on the mutual information.

For any two random variables $\mathcal{X}, \mathcal{Y}$, the mutual information is

$$I(\mathcal{X};\mathcal{Y}) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \text{Prob}(x,y)\log\left(\frac{\text{Prob}(x,y)}{\text{Prob}(x),\text{Prob}(y)}\right)$$

In our case, the mutual information between two clusterings will then be

$$I(\mathcal{X}^c;\mathcal{X}^{\hat{c}}) = \sum_{x\in\text{uni}(c)}\sum_{y\in\text{uni}(\hat{c})} \frac{n_x^c n_y^{\hat{c}}}{n_{xy}^{c,\hat{c}}} \tag{4.6}$$

In the formulation of mutual information in equation 4.6, for three subsets c, d, e if d is identical to c except that some of the communities in c are split into subcommunities in d, it will be the case that $I(\mathcal{X}^c;\mathcal{X}^d) = 1$ and that $I(\mathcal{X}^c;\mathcal{X}^e) = I(\mathcal{X}^d;\mathcal{X}^e) = 1$. This is clearly insufficient for characterizing the similarity of two community labelings where the number of communities chosen is critical. The typically used version of this statistic is the normalized

7. For any community labeling c and any specific community label k,

$$n_k^c = |\{u \in V \mid c_u = k\}|$$

For two labelings, c, d,

$$n_{k,l}^{c,d} = |\{v \in V \mid c_v = k\} \cap \{v \in V \mid d_v = l\}|$$

Lastly, for any set X let $\text{uni}(X)$ denote the unique elements of X.

mutual information (NMI), which is

$$NMI(\mathcal{X};\mathcal{Y}) = \frac{2I(\mathcal{X};\mathcal{Y})}{H(\mathcal{X}) + H(\mathcal{Y})}$$

The function $H(\cdot)$ calculates the entropy of a random variable with

$$H(\mathcal{X}) = -\sum_{x\in\mathcal{X}} \text{Prob}(x)\log(\text{Prob}(x))$$

For two random variables $\mathcal{X}^c$ and $\mathcal{X}^d$, it will be the case that $H(\mathcal{X}^c) < H(\mathcal{X}^d)$, leading to $NMI(\mathcal{X}^c;\mathcal{X}^d) \neq 1$ and addressing the issue raised with using mutual information out-of-the-box.

One last facet of evaluating CD techniques is the quantification of underfit and overfit. For a more typical supervised learning context, issues like overfit may be quantified by preserving a test set for evaluation against a ground truth. In CD, the concept of a hold out set is not easy to conceive of - there is not necessarily a ground truth benchmark and there is not a way of dividing a graph into training and test sets. Similarly for quantifying underfit, without a ground truth for comparison, there is no baseline against which parameters may be tuned. Approaches to address these two issues are *link prediction* and *link description*.

Link prediction is a learning task where a subgraph of all vertices and a fraction of the edges is used in training to form community labelings. When the original graph is $G = (V, E)$, the subsampled graph is $G_o = (V, E_o)$ where $|E_o| = \alpha|E|$ for some $\alpha \in (0, 1)$. The community labeling on $V(G_o)$ is used to rank the potential edges $P = \{(v, w) \in V \times V \mid (v, w) \notin E_o\}$ which did not appear in the subgraph based on whether or not they connect nodes in different communities. Depending on the CD method used, the ranking may be a continuous or discrete ranking. For any threshold of the values that make up that ranking, the classification of edges into positive and negative classes produces a confusion matrix relative to the ground truth positive labeling $P \cap E$ and negative labeling of $P \cap E^C$. For a set of different thresholds, an ROC curve may be made from which the AUC may be calculated. The ranking of "out-of-

sample" edges may be considered a measure of overfitting. The task is more difficult when a smaller subgraph is observed. By plotting a method's average AUC for different proportions of edges observed in the subgraph, the method's tendency to overfit is quantified. If a method has lower AUC, it is more likely to fit communities that do not generalize well when new information is observed.[98]

Link description is a similar learning task, where a proportion of the graph's edges are sampled, constructing a subgraph with all the graph's vertices but only a subset of its edges, $G_o = (V, E_o),\ E_o \subset E$. After the community labeling is defined on the subgraph, a score is defined on the edges in P and, given a threshold for the ranked values, a confusion matrix is calculated, relative to a ground truth positive class of E_o and a negative class of E_o^C. Again, using this procedure across different size subgraphs, an array of AUCs are computed, giving a sense of how closely the method fits the observed data.[64] All the metrics listed in this section are used extensively in chapter 6 and chapter 7.

4.2.4 General Review of Methods

Having introduced the basic concepts underlying an array of CD techniques, we provide some comparative analysis of these approaches in practice. These results have greatly informed the direction of our own work. We have gone into some detail in methods that are most relevant to our own, but acknowledge that some of the most successful methods, such as the Potts model of statistical mechanics CD,[117] have not been properly introduced to avoid unnecessary discussion.

One early review of CD methods by Lancichinetti *et al.* was particularly noteworthy for introducing a new benchmark for community detection, effectively imitating real-world network structure in a synthetic graph, the so-called (LFR) graph. Using this graph, the authors were able to evaluate a collection of existing methods on fair, realistic tests. They found that the agglomerative modularity-minimizing Louvain method and the Potts model were able to provide the most reliable recovery of community structure, as measured in NMI.

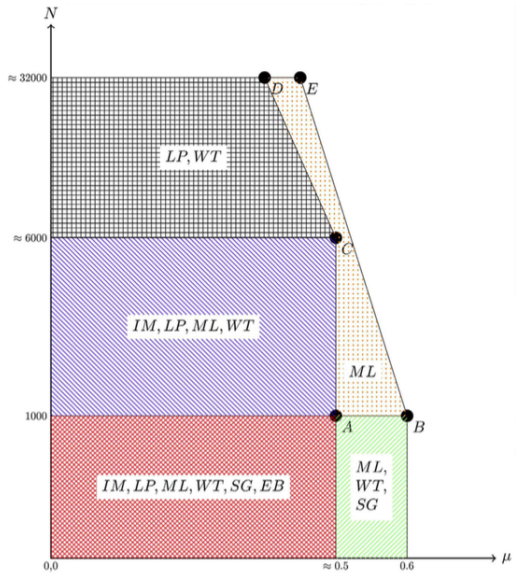


Figure 4.1: Regimes where CD methods are estimated to perform best. LP = Label propagation; WT = Walktrap; ML = Multilevel/Louvain; IM = Infomap; SG = spin glass; EB = edge-betweenness

Subsequent to its introduction, LFR graph models have become a standard synthetic graph benchmark for large-scale CD. In subsequent reviews, such as that of Yang *et al.*, the performance of the Louvain method in most graph settings continues to produce better communities in terms of accuracy and in terms of computation time. Still, weighing factors such as the difficulty of the CD task - where the precision of a method will be more important - or the size of the graph - where the computational expenditure of a method will be more important -, they produce the diagram in figure 4.1 to describe the settings in which different methods are appropriate.[147] We conclude our review of methods here, and will focus on methods that focus specifically on incorporating hierarchical, multilevel, or multiresolution structure in chapter 7.

CHAPTER 5

NEW PERSPECTIVES ON THE MULTIRESOLUTION MATRIX FACTORIZATION

In this chapter we focus on developments in large-scale MRA from several directions. In sections 5.1 and 5.2 we describe and consider modifications to the multiresolution matrix factorization (MMF) designed specifically for the analysis of large graphs. These modifications take note of the typically hyperbolic structure of real-world graphs, and we direct the factorization algorithm to take advantage of this latent geometry. Section 5.1 makes a case for how MMF may be seen as a coarse graph approximation in a low dimensional hyperbolic space, and section 5.2 proposes ways to enrich the underlying hyperbolic space, to produce more expressive and generalizable wavelet bases. In section 5.3 we introduce several ways to use trees, and collections of trees, to generalize wavelet decompositions of graphs and signals. These ideas are motivated by existing methods that make use of collections of trees, as well as untapped ideas from multiresolution analysis in continuous spaces, such as cycle spinning. Lastly, in section 5.4, we consider multiresolution representations of data from the perspective of regularization theory. We show the way that function regularization using wavelets corresponds to Laplacian regularization and describe the way that our central "tree-likeness of data" assumption is equivalent to a particular form of regularization.

5.1 Hyperbolic Properties of Multiresolution Matrix Factorization

5.1.1 MMF: The Algorithm and its Application

The complex hierarchies found in many large, contemporary machine learning datasets, have structure that is not easily summarized by traditional methods such as eigendecompositions. Whereas tools like PCA are useful for reducing the dimensionality of data, the dimensionality

reduction depends on a small number of eigenvectors. Using a small number of eigenvectors, which are generally nonsparse, is computationally expensive and also poorly-matched to the characteristics of many real-world problems. In contrast, the MMF uses a sequence of sparse orthogonal matrices to sequentially find an approximate diagonalization of a matrix, or, more precisely, a factorization of the matrix as a product of an orthogonal matrix, a nearly diagonal matrix, and the orthogonal matrix's transpose. In its most basic interpretation, MMF is a wavelet transform that seeks to find a multiresolution factorization that is as close to the original matrix as possible. From this perspective, for a symmetric matrix $M \in \mathbb{R}^{n \times n}$, the MMF factorization emerges from minimizing the following objective function:

$$\text{argmin}_{Q_1 \in G^k(S_1), Q_2 \in G^k(S_2), \ldots, Q_L \in G^k(S_L), H \in \mathcal{H}^n_{S_L}} \left\| M - Q_1^T \cdots Q_L^T H Q_L \cdots Q_1 \right\|$$

The constraints on H and $\{Q_i \mid 1 \leq i \leq L\}$ may also be expressed as:

1. $H \in \mathbb{R}^{n \times n}$ is a diagonal matrix, except for nonzero entries in the submatrix defined by the indices S_L, and

2. For $1 \leq i \leq L$, $Q_i \in \mathbb{R}^{n \times n}$ is a matrix which is the identity, except for a $k \times k$ rotation matrix for k of the indices out of S_L.

To make the implementation and interpretation as clear as possible, the algorithm for producing said factorization is presented in algorithms 1 and 2, without any substantive changes from its presentation in earlier publications.[85, 86]

In algorithm 1, the function **GivensRotation** returns a $n \times n$ matrix with a rotation to diagonalize the $k \times k$ submatrix corresponding to v_{chosen}. The use of $\mathcal{H}^n_{S_L}$ as a function in the return statement of algorithm 1 is not intended literally, as $\mathcal{H}^n_{S_L}$ is simply a vector space of matrices with nonzero offdiagonals in the submatrix defined by the indices S_L. Rather, the matrix that is returned is the projection of A_L into the space $\mathcal{H}^n_{S_L}$. This is necessary to ensure that the middle term of the factorization is core diagonal. In order to actually perform the projection of A_L into the space of core diagonal matrices with nonzero

Algorithm 1 Computing the MMF for A using the "Greedy Jacobi" algorithm.

Require: k, L, a symmetric matrix $A_0 = A \in \mathbb{R}^{n \times n}$
  $S_0 = \{i \mid 0 \leq i \leq n\}$
  for $l = \{i \mid 0 \leq i \leq L\}$ **do**
    $v_{\text{chosen}}, v_{\text{drop}} = \text{findRotation}(A_{l-1}, S_{l-1}, k, n_{\text{drop}})$
    $Q_l = \text{GivensRotation}(n, v_{\text{chosen}})$
    $A_l = Q_l A_{l-1} Q_l^T$
    $S_l = S_{l-1} - v_{\text{drop}}$
  end for
  return $Q_1, Q_2, \ldots, Q_L$ and $H = \mathcal{H}^n_{S_L}(A_L)$

Algorithm 2 findRotation

Require: A, S, k, n_{drop}
  $v_{\text{base}} = \text{randomChoose}(\text{set} = S, n_{\text{choose}} = 1)$
  $v_{\text{others}} = \text{chooseClosest}(\text{set} = S - v_{\text{base}}, \text{base} = v_{\text{base}}, d = A[v_{\text{base}}], n_{\text{others}} = k - 1)$
  $v_{\text{drop}} = \text{selectDrop}(d = A[v_{\text{base}}], \text{set} = v_{\text{base}} + v_{\text{others}}, n_{\text{drop}})$
  return $v_{\text{base}} + v_{\text{others}}, v_{\text{drop}}$

offdiagonals in positions S_L, we reset $A_L[v_{\text{drop}}] = 0$ (except for the diagonal entries of $A_L[v_{\text{drop}}, v_{\text{drop}}]$ which remain unchanged). Some of the characteristics of this core diagonal matrix are discussed in footnote 7. In algorithm 2, **randomChoose** selects elements from a set without replacement. The two remaining undefined functions - **chooseClosest** and **selectDrop** - are in fact hyperparameters to be set before the algorithm is started. Different approaches to choosing which nodes to select in any given rotation is the topic discussed in section 5.2. Choosing how nodes are chosen to be remove from the set S_L for consideration in future rotations is also a parameter of the model. Typically, though, the entries in d of higher magnitude are selected for rotations and entries with lower magnitude are selected to be dropped. The parametrization of these functions is not defined in algorithm 2 for the sake of brevity. It is helpful to see the factorization that MMF produces, and a visualization of the terms that make up the wavelet transform are in figure 5.1.[1]

Since its introduction,[85, 86] the MMF and its (more scalable) sibling the parallel MMF

1. A parallelized version of the MMF algorithm also exists, where nodes are partitioned into blocks and rotations are executed independently on each block, but for the discussion in this section, the specifics of the parallelized version are not relevant.

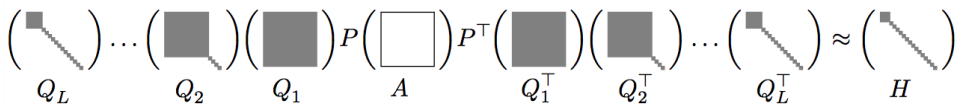


Figure 5.1: A visualization of the way that MMF wavelets are defined through a sequence of k-point Givens rotations of nodes from among a successively smaller active set of nodes.

(pMMF) have found application in the compression of large kernel matrices,[130] as a preconditioner for large matrix operations,[105] and in the scalability of Gaussian Process regression (GPR).[42] Using the same backbone algorithm, these applications use an orthogonal basis of wavelets, where wavelets have varying levels of sparsity. These wavelets are defined through a sequence of local rotations, aiming to preserve localized interactions between datapoints or nodes in a graph. The MMF algorithm may be seen as a greedy Jacobi algorithm for the diagonalization of symmetric matrices, where instead of selecting nodes for rotation to eliminate the largest off-diagonal entry they are selected in order to minimize the overall matrix approximation error as measured by Frobenius norm error.

The closest relatives of MMF in the computational MRA community are diffusion wavelets, [36] spectral graph wavelets,[71] the treelets algorithm,[93] and fast multi-scale matrix factorizations using structured matrix models.[10, 100, 116] MMF is distinguished among these widely used approaches by having the following qualities: (1) being an algorithm whose output is not only wavelets, but a complete matrix factorization, (2) producing a tree of Givens rotations based on agglomerative clustering, (3) employing higher-order local rotations than 2-point rotations alone, and (4) having a highly parallelizeable implementation. This chapter focuses primarily on unique features (2) and (3) in order to orient the role of MMF within the world of both multiresolution analysis and hyperbolic embeddings.

5.1.2 MMF: Factorization or Forest?

To understand our subsequent discussion of MMF as a metric embedding, we detail precisely how we define a single MMF's graph of rotations. In each stage of MMF's factorization

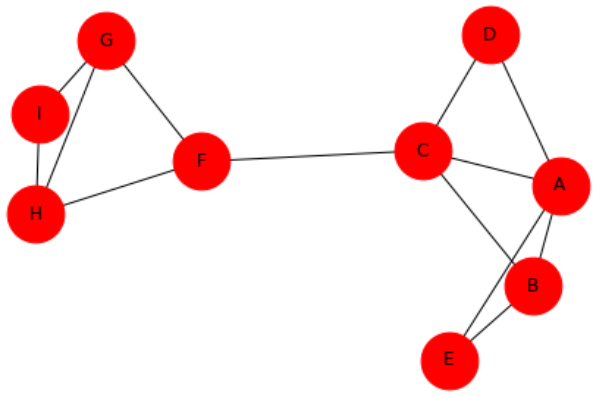


Figure 5.2: The original graph with respect to which subsequent rotation DAGs are defined

algorithm, a small set of nodes[2] is selected for a Givens rotation from among all nodes still considered "active."[3] Among these rotated nodes, some nodes are removed from the active set and some nodes are preserved in the active set.[4] At each iteration, though, the active set invariably shrinks. In the case when $k = 2$ and $n_{\text{drop}} = 1$, the sequence of rotations may be recorded as a tree. Consider the simple graph in Figure 5.2 and the corresponding MMF-Givens rotation trees in Figures 5.4, 5.5, and 5.6. The MMFs in these figures have rotations defined according to the schedule in figure 5.3. These three MMFs demonstrate how the interplay of k and n_{drop} works. If n_{drop} is large, then a larger number of indices are dropped at each iteration of the factorization, the algorithm converges faster, and the corresponding tree has smaller diameter than when n_{drop} is smaller. This is evident in the manner that the graph in Figure 5.5 has a smaller relative diameter than the tree in figure 5.4.

When k grows but n_{drop} does not, there are more total edges in the rotations tree (i.e. more iterations), leading to the graph of rotations no longer being a tree, but rather a directed acyclic graph (DAG). The size of the loops will depend on the strength of the

2. The size of this set is determined by the MMF's parameter k.

3. This is the set of nodes which have either not been involved in a rotation yet, or have been involved but still kept for use in future rotations.

4. The number of nodes that are removed out of all nodes rotated is determined by the MMF parameter n_{drop}.

Keep	Discard
A	D
I	G
H	I
C	A
B	E
C	B
F	H
C	F

(a) Sequence of rotations for $k = 2$, $n_{\text{drop}} = 1$.

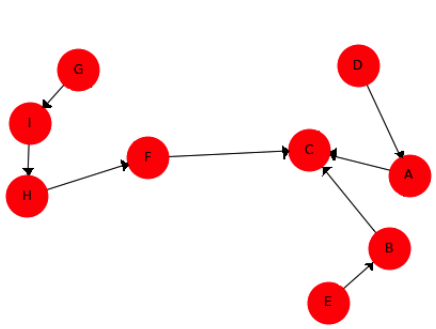
Keep	Discard1	Discard 2
G	H	I
A	B	E
C	A	D
C	F	G

(b) Sequence of rotations for $k = 3$, $n_{\text{drop}} = 2$.

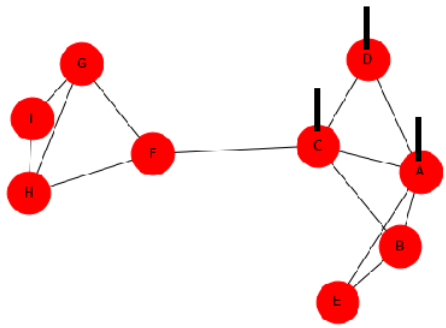
Keep1	Keep2	Discard 1
A	B	E
C	A	B
C	A	D
G	H	I
F	G	H
F	C	G

(c) Sequence of rotations for $k = 3$, $n_{\text{drop}} = 1$.

Figure 5.3: Sequence of rotations for different pairs of (k, n_{drop})

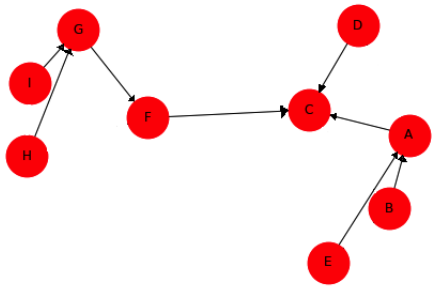


(a) Givens rotations graph

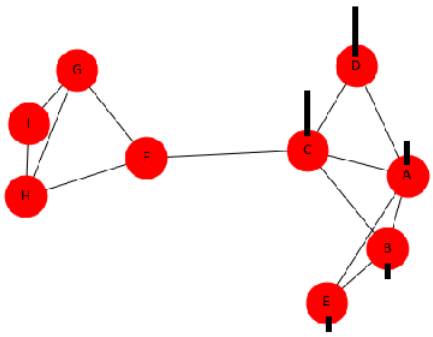


(b) Wavelet with largest support including clique ACD

Figure 5.4: Givens 2-point rotation, $n_{\text{drop}} = 1$

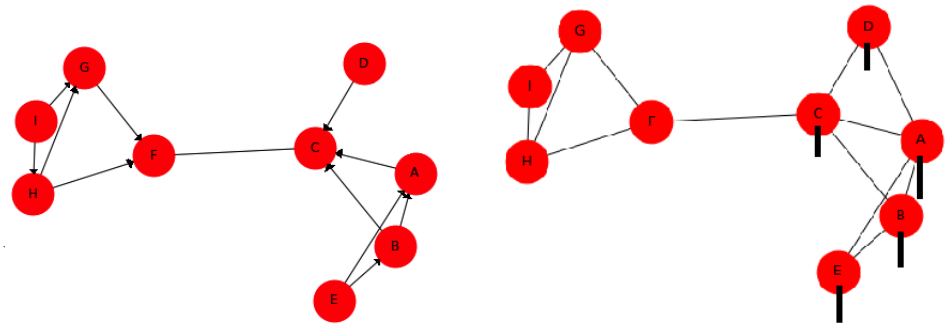


(a) Givens rotations graph



(b) Wavelet with largest support including clique ACD

Figure 5.5: Givens 3-point rotation, $n_{\text{drop}} = 2$



(a) Givens rotations graph

(b) Wavelet with largest support including clique ACD

Figure 5.6: Givens 3-point rotation, $n_{\text{drop}} = 1$

graph's clustering structure.

A loop of length 3 exists in a 3-point Givens rotations DAG if some of the non-dropped nodes involved in rotation i are subsequently rotated together again in a subsequent rotation. In situations where large loops exist in a graph, or where a large number of nodes are in a single cluster, if $n_{\text{drop}} < k - 1$ then there will be loops in the Givens rotations DAG, and the resulting rotations graph will be less tree-like.[136] However, if the input graph is a tree, or approximately tree-like, the DAG will be as well.[70] In this sense, the rotation DAG produced by MMF is a "hyperbolicization" of the graph, edges are removed to make it more tree-like.

5.1.3 Interpretation of MMF

As already noted in section 2.3.3 and the beginning of this chapter, MMF's use of higher order rotations distinguishes its wavelets from other methods and extends its explanatory power. It also endows MMF with a unique interpretation as a means of producing an MRA basis. In the traditional case of wavelets being used to decompose signals in $\mathbb{R}$, predefined wavelet construction methods exist whose wavelets are guaranteed to be both sparse and locally smooth. In the graph setting, since every graph defines its own metric space, it is impossible to have a similarly algorithmic construction method independent of the observed

data. MMF, though, provides a lens to see every graph as a real-world realization of a hyperbolic metric space subject to some random distortion. By considering the Givens rotations DAG produced by any MMF, we have a window into a hyperbolic metric space representation of the graph. This hyperbolic space is a discrete space, whose edges are local rotations. The wavelets produced by these rotations, might not be smooth with respect to the original input graph, but they are smooth with respect to the hyperbolic representation.[5] In figures 5.4(b), 5.5(b), and 5.6(b), we present some example wavelets with respect to the hyperbolic space characterized by the DAG of their Givens rotations, and the original graph. By seeing MMF's backbone, the rotations graph, as a hyperbolic embedding of the input graph, MMF serves as a vehicle for realizing two distinct goals. On the one hand, as a matrix factorization, factorizing the graph Laplacian in particular, MMF is situated among factorization-based graph embedding techniques.[66] On the other hand, by considering the DAG of rotations itself, MMF gives a simplified, tree-like picture of a graph, without actually needing to estimate the wavelet basis itself. Is MMF a multiresoluton matrix factorization or a multiresolution matrix tree?

5.1.4 Limitations

While MMF finds a tree, or a tree-like representation of a graph, the objective function for MMF (minimizing Frobenius norm error between the original graph and factorization) is not equivalent to finding an optimal representation. In fact, there is a tension at each iteration of the MMF algorithm between local rotations that enforce a strong hyperbolic structure and how closely the MMF approximates the graph. We turn to some ways of extending MMF in the graph context that take advantage of the fact that hyperbolic structure underlies many of the real-world graphs we anticipate observing. Subsequently, we change direction slightly by considering, not how MMF individually may be improved, but how collections trees in

5. In section 5.4.3 we return to the idea of hyperbolic representations of graphs through the lens of graph regularization.

general improve graph approximations, and by association, how collections of MMF may be used and interpreted.

5.2 New Methods for Choosing Rotations in the Multiresolution Matrix Factorization

Rotations are MMF's wavelet building blocks. The resulting error in MMF's wavelet approximation will be small only if nodes have been selected for a rotation which are as similar as possible to each other. For a human or a computer, selecting a set of k elements out of n based on their k-wise similarity is no small task. The most pivotal challenge in attempting to construct a greedy factorization such as MMF is to ensure that each greedy step is taken as carefully and deliberately as possible.

5.2.1 Two Ways of Measuring Similarity in a Tree-like Graph

We suggest two ways of orienting MMF to perform in the graph context by (1) noting that we will be considering massive graphs with high levels of edge sparsity (the number of edges in a graph grows sub-quadratically as a function of the number of nodes), and (2) that we anticipate finding hierarchical structure in graphs.[6] These two points of orientation serve as our guides for suggesting new ways to choose which groups of nodes should be rotated together in each rotation stage. In line with the methodology proposed by Mishne *et al.*, we consider that a matrix is not always the best guide for its own organization, and subsequent factorization, in the presence of more suitable metrics.[104] For this reason, while MMF is adaptive to general matrices, we consider two graph metrics with interpretations uniquely suited to the real-world data we anticipate encountering. The two metrics we consider - one

6. Even in the absence of there being a logical rational for hierarchical structure, we assume some hierarchy exists. For instance, in any graph, unless there is a single cluster of nodes with a uniform degree distribution, *some* hierarchy exists, even if it is only determined by ranking nodes based on their degree. Among hand-written digits, for example, a graph of their similarity has a latent hierarchy which will structure ones based on their one-ness. A one situated at the hub of a hub-and-spoke cluster is the "parent" one for many distinct varieties of ones, whereas ones situated in the spokes will be more unique and have fewer close relatives.

based on effective resistance and one based on random partitions - are focused specifically on graphs' tree-likeness and use tools derived from hyperbolic embeddings and from collections of spanning trees.

As has already been related in chapter 3, hyperbolic embeddings provide a natural representation for many real-world graphs. Whereas a graph Laplacian holds valuable information about graphs in terms of their adjacency matrix and as a regularization operator, the distance between nodes in their hyperbolic embedding may provide an easier metric of node similarity than the mere existence or lack of existence of an edge between them.

Algorithm 3 Random partitions algorithm for graphs using spanning trees

Require: nodes a, b, adjacency matrix A, and number of trees n_{trees}
  count $\leftarrow 0$
  $F \leftarrow \text{SpanningTreeAlg}(A, n)$
  for S in F **do**
    root $\leftarrow$ Sample $(\{i \mid 1 \leq i \leq \dim(A)\} - \{a, b\})$ {Ideally this sampling step would upweigh nodes closer to the graph's center.}
    c $\leftarrow$ LeastCommonAncestor(S, root, a, b)
    if $d_S(a, c) < d_S(a, \text{root})$ and $d_S(b, c) < d_S(b, \text{root})$ **then**
      count $+= 1$
    end if
  end for
  return $\frac{\text{count}}{n_{\text{trees}}}$

Next, our motivation for employing collections of spanning trees to assess node similarity comes from research on "big data kernels." The random forest kernel, random partition kernel, and Mondrian kernel all provide ways of measuring similarity between observations in large-scale kernel-based learning.[40, 15] These kernels have been presented in the structured dataset setting and we propose to extend their application to the unstructured world of graphs. Just as we assume two nodes' similarity is higher if they cooccur on the same side of lines or planes in $\mathbb{R}^n$, we also assume two nodes are more similar if they are in the same branch of a tree. We present our specific algorithm for the graph implementation of the random partitions algorithm in algorithm 3. The use of effective resistance is similarly based on spanning trees, and may in fact be calculated as the probability of an edge occurring

in a spanning tree. The spanning tree separation (SPS) has also been used as a tool for measuring similarity in hierarchical clustering tasks.[82]

5.2.2 Implementation

Using these two similarity metrics, effective resistance and random partitions, in addition to the original graph Laplacian, demonstrates the MMF as a more general tool for matrix organization and factorization. Graph Laplacians may be necessary for applications such as semi-supervised learning or Gaussian Process regression, but the graph Laplacian itself may not be the best tool for data organization. The original form of MMF aims to find an approximate factorization of A using the following objective:[7]

$$\min_{H \in \mathcal{H}_L^n, Q_1, \ldots, Q_L \in \mathcal{O}^n} \left\| A - Q_1^T \cdots Q_L^T H^{(L)} Q_L \cdots Q_1 \right\|_{\text{fro}}^2 \tag{5.1}$$

The greedy Jacobi algorithm for solving this optimization problem aims to find the best rotation to minimize the error at each iteration. After L steps the output of the algorithm will be the matrices $Q_1, \ldots, Q_L, H^{(L)}$, which yield an approximate factorization of A as

$$Q_1^T \cdots Q_L^T H^{(L)} Q_L \cdots Q_1 \tag{5.2}$$

The adage "garbage in, garbage out" applies in this situation. The MMF will only be as good as the rotations that make it up - the less similar the nodes in a rotation, the less similar their local environments will be, leading to more error accumulated in the stage. We aim

7. The $\mathcal{H}_L^n$ in line 5.1 refers to the set of $n \times n$ core-diagonal matrices of order L. A matrix M is core-diagonal when it may be expressed as the sum of a diagonal matrix D and, given a set of indices K, another matrix E^K, where $\forall i, j \notin K,\ E^K(i,j) = 0$. Trivially, every matrix is core-diagonal if we let $K = \{1, \ldots, n\}$. The order L of the set determines the maximum cardinality of K. For a simple, non-parallelized MMF where n_{drop} is a fixed integer, $|K| = \max(1, n_{\text{drop}} L)$. Regarding the rotation matrices Q_i, besides belonging to the set of $n \times n$ rotation matrices, an additional constraint (not mentioned in line 5.1, but very important) is the nodes which the rotations may include. Whereas at the initial stage of determining Q_1, all nodes are considered, in the subsequent Q_2, only $n - n_{\text{drop}}$ are considered, having excluded the indices dropped in the first iteration. The active set included in rotations strictly decreases at each iteration as a function of i.

to improve MMF's ability to select nodes for a rotation by selecting rotation matrices using similarity metrics other than the entries of the matrix itself. So, instead of minimizing the objective in line 5.1 for the factorization of A, we define a distance metric D and minimize the objective in line 5.3.

$$\min_{H \in \mathcal{H}_L^n,\ Q_1,\ldots,Q_L \in \mathcal{O}} \left\| D - Q_1^T \cdots Q_L^T H_D^{(L)} Q_L \cdots Q_1 \right\|_{\text{fro}}^2 \tag{5.3}$$

If the problem at hand only requires a collection of wavelets, then the columns of the matrix $Q_1^T Q_2^T \cdots Q_L^T$ are all that is needed. If a full factorization is required for approximate matrix operations on A, then the factorization $Q_1^T \cdots Q_L^T H_A^{(L)} Q_L \cdots Q_1$ where $H_A^{(L)}$ is defined as the matrix which minimizes the objective in line 5.4.[8]

$$\min_{H \in \mathcal{H}_L^n} \left\| H - Q_L \cdots Q_1 A Q_1^T \cdots Q_L^T \right\|_{\text{fro}}^2 \tag{5.4}$$

5.2.3 Evaluations and Observations

We empirically validate our conjecture that measures of graph distance other than the graph Laplacian can guide the selection of nodes for rotation better than the original matrix entries. We demonstrate using synthetic graphs with varying character that alternative methods outperform the standard approach in terms of matrix reconstruction error and in terms of accuracy in applied tasks. The matrix reconstruction error is measured using the squared Frobenius norm error relative to the original matrix's squared Frobenius norm. Accuracy is assessed in a straightforward inference problem using label propagation on a graph for multi-class classification. The difficulty of the task grows as a function of the number of communities that exist in the graph. The graph size is held constant at 1024 nodes across all simulations so that as the number of communities grows, the community size gets smaller.

8. Note that, while the objective function in line 5.3 is minimized over core diagonal matrices and orthogonal matrices, the objective function in line 5.4 is only minimized over core diagonal matrices - the orthogonal matrices are the same ones that were discovered in line 5.3.

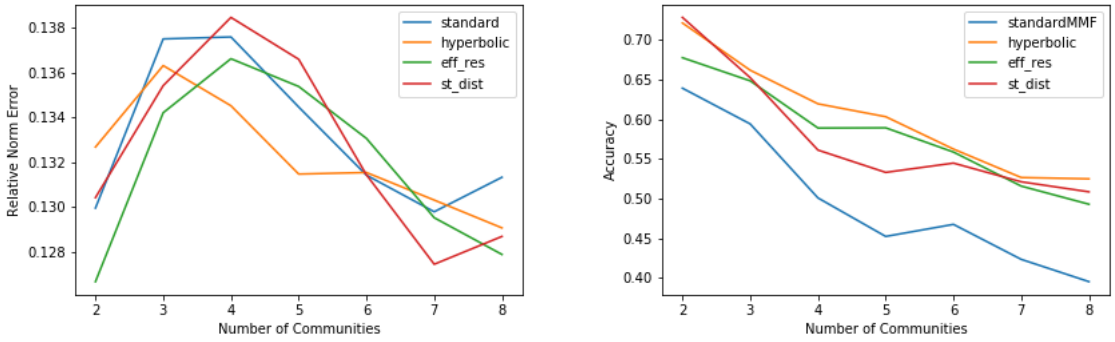
In SSL, where only a small portion of nodes' labels are known, the task is harder when the total set of nodes belonging to each class is smaller. We do not consider real-world graphs in the experiments that follow, as the ability of methods to perform in more or less challenging scenarios is of interest. For that reason, it is critical that we have the control to determine how difficult a given reconstruction or prediction task will be.

Our first application of these alternative distance measures is in a label propagation task on the Sales-Pardo (SP) graph. SP graphs have strong clustering structure, possibly at multiple scales of resolution. For the adjacency matrix of an SP graph refer to figure 5.10. All nodes in a cluster are modeled to have almost identical degree, resulting in a degree distribution that is uniform over a very small number of similar values. We consider it first due to its simplicity, intuitiveness and ease of implementation.

As noted already, we consider the effectiveness of different distance measures in terms of matrix error and task accuracy and record results in figure 5.2.3. As a function of community size, the matrix reconstruction error using all distances is effectively identical in figure 5.8(a). The relative error is between 0.128 and 0.138 for all community sizes. They also possess the same feature of exhibiting increased rates of relative error for intermediate numbers of communities. If one had to identify a method with the worst reconstruction error it would probably be the hyperbolic distance. When we consider the accuracy of labeling using label propagation in figure 5.8(b), though, the picture is quite different. The accuracy of MMF using the standard distance measure of the graph Laplacian is nearly ten percent lower than all other distances, with the hyperbolic distance performing uniformly better across all community sizes. One obvious takeaway is that matrix reconstruction error is not always the best proxy for the goodness of a matrix approximation procedure. We surmise that first embedding a graph in a multidimensional hyperbolic space effects a regularization step, magnifying existing graph structure.[9]

We next consider a more realistic synthetic graph type called the LFR graph. LFR differs

9. We return to this in section 5.4.3.



(a) Matrix reconstruction error - $\frac{\|M - M_{\text{approx}}\|_{\text{fro}}^2}{\|M\|_{\text{fro}}^2}$

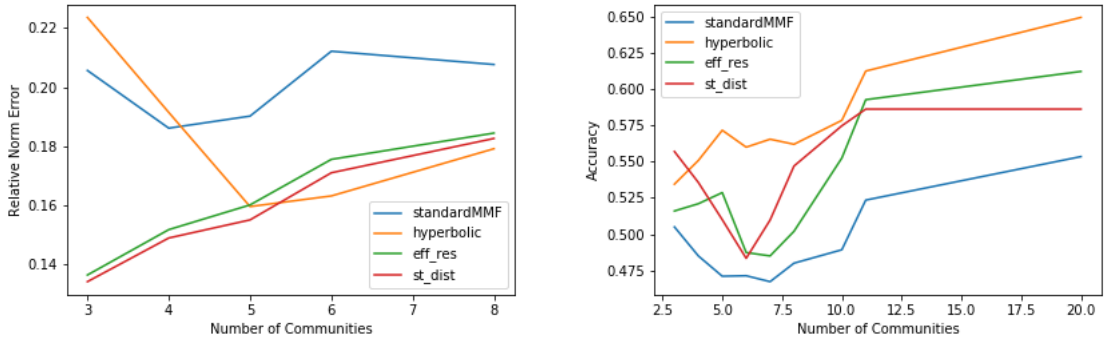
(b) Prediction accuracy where training is 10% of nodes

Figure 5.7: Matrix reconstruction and predictive accuracy for MMF on SP graphs using four different methods for selecting the nodes to be in a rotation: **standard** (matrix entries), distance in a **hyperbolic** embedding, **eff**ective **res**istance distance (estimated using a finite sample of spanning trees), and **s**panning **t**ree distance (average distance across a finite set of spanning trees). Results are averaged over five repetitions

from SP primarily by the degree distribution of its nodes. Whereas SP graphs model nodes within each community as having approximately equal degrees, LFR graphs model a power law distribution of node degrees, modeling the more realistic hub-and-spokes graph model observed in real-world situations. In this setting, we observe similar results as was the case in the SP graph. Regarding their matrix reconstruction, all distance metrics are nearly identical and show no clear trends as a function of community size. When the accuracy of the methods at a prediction task is considered, however, the methods are suddenly differentiated. The standard MMF using matrix entries to determine rotations is outperformed by the three new graph metrics considered. The three methods produce similar levels of accuracy across all community sizes, indicating their predictive capabilities, and their predictive edge relative to standard MMF, are related.[10]

In summary, we have featured a property in our multiresolution factorization that aligns

10. While we did not intend this to be the case *a priori*, we observe that it seems the label propagation task for SSL becomes easier as a function of community size. We had actually intended community size to make a classification task harder. For LFR graphs, this is not quite the case because the hub-and-spokes-ness of the model means that small groups of nodes have very small diameter and very few out-of-community connections. Regardless, we find accuracy as a function of community size an interesting relationship by observing that the three alternative graph metrics all exhibit similar trends in performance.



(a) Matrix reconstruction error - $\frac{\|M-M_{\text{approx}}\|_{\text{fro}}^2}{\|M\|_{\text{fro}}^2}$

(b) Prediction accuracy where training is 10% of nodes

Figure 5.8: Matrix reconstruction and predictive accuracy for MMF on LFR graphs using four different methods for selecting the nodes to be in a rotation: **standard** (matrix entries), distance in a **hyperbolic** embedding, **eff**ective **res**istance distance (estimated using a finite sample of spanning trees), and **s**panning **t**ree distance (average distance across a finite set of spanning trees). Results are averaged over five repetitions

with previous findings in the estimation of distances on large graphs. The estimated distances that emerge from Euclidean embeddings of graphs tend to be less accurate than the same calculations from hyperbolic embeddings of graphs.[125, 150] Similarly, in the case of calculating distances for the sake of identifying small clusters of nodes bearing high levels of similarity, we find that by preprocessing data using a hyperbolic embedding[11] we are able to boost the accuracy of the resulting approximate factorization in a simple label propagation task. Surprisingly, this predictive edge is not apparent in the approximation error evidenced by Frobenius norm error, indicating that the objective of this preprocessing step is not merely to achieve lower Frobenius norm error in matrix reconstruction, but to enforce a structure on the graph that makes the subsequent learning process more accurate. Notably, it is not only the hyperbolic embedding that achieves a noticeable increase in predictive accuracy. Similar performance is achieved by spanning tree-based graph metrics: the random partition metric and the effective resistance metric. The connection between hyperbolic embeddings and the spanning tree distances may be explained by the hyperbolicity of products of tree

11. The preprocessing step we are referring to is, specifically, substituting the regular graph metric with pairwise distances in the hyperbolic embedding space.

graphs discussed in section 3.3.3. We are not satisfied by the current characterization of these products' hyperbolicity, and proving tight bounds on these graphs' hyperbolic embeddings' distortion or their δ-hyperbolicity is our primary topic of interest for future work.

5.2.4 Complexity Considerations

MMF has thus far provided a valuable contribution in terms of matrix compression and approximation, and in terms of speeding up expensive tools for statistical inference. Among the plethora of matrix approximation techniques, we aim to ensure that the methods presented in section 5.2.3 do not pose a computational burden. The original MMF method's running time for finding k-point rotations $\mathcal{O}(n^{k+1})$[85] was improved in a parallelized method to $\mathcal{O}(c^{k+1})$ where n is the matrix size and c is the typical size of components in a matrix partition. Empirically, the running time of pMMF was found to scale approximately linearly with the dimension of the matrix.[86]

By including preprocessing steps that enforce tree-likeness in the data, we inevitably add some additional computation time. The complexity of hyperbolic embeddings varies based on the technique. The Poincare embedding suggested by Nickel *et al.* uses stochastic gradient descent to find node embedding coordinates. The computational and memory complexity depend linearly on the number of nodes and is easily parallelizable.

For years, the primary algorithm for sampling spanning trees from the uniform spanning tree distribution were derived by Aldous[8] and (separately) by Broder[8] for which the running time was on the order of the cover time for the graph. That is, the amount of time for a random walk to cover all the nodes in the graph. The complextiy of an unweighted graph's cover time is $O(nm)$ where n is the number of nodes and m is the number of edges. Later work has improved this algorithm to enable the sampling of trees uniformly with high probability in $\tilde{O}(m^{\frac{4}{3}})$, $\tilde{O}(mn^{\frac{1}{2}})$, or $\tilde{O}(n^{\omega})$[12] depending on the properties of the graph

12. ω denotes the exponent for the fastest square matrix multiplication algorithm. Generally we assume $\omega \approx 2.373$, but for sparse matrices it may be smaller.

at hand. The later stage of computations of distance or clustering for the tree graph are computed in linear time.

Evidently, including the preprocessing steps will not add runtime for a hyperbolic embedding, and for the sampling of spanning trees, the complexity is only slightly increased. Regardless, the overall factorization computation remains sub-quadratic and the performance gains are noteworthy. More investigation of the computational burden of the algorithms in practice is a topic of further investigation in future projects.

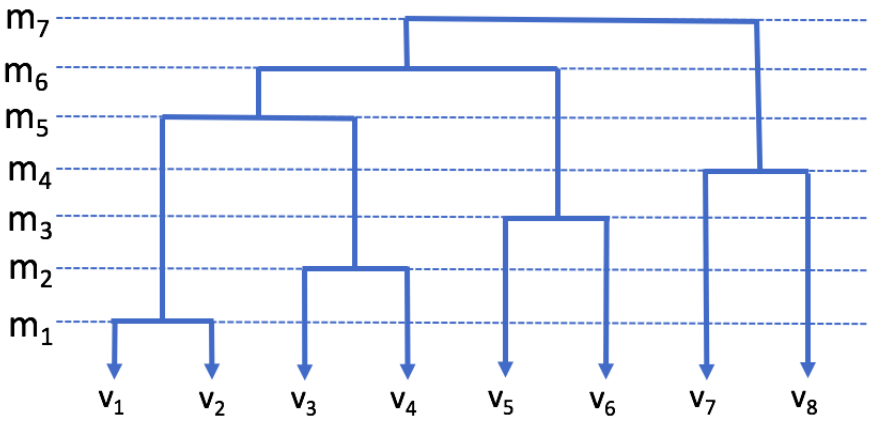
One more promising direction for quickly selecting rotations comes from the world of numerical optimization. Shalit & Chechik propose a gradient descent algorithm for orthogonal matrices using Givens rotations.[122] Given an objective function f which depends on an orthogonal matrix, they define a specific algorithm for choosing Givens rotations updates and the direction for descent to minimize f. They describe this algorithm taking $O(n)$ time and with guaranteed convergence to a local optimum. This formula fits our situation spectacularly, where we aim to lose as little information as possible in each rotation stage. A clear next step would be to combine the Givens rotations coordinate descent with MMF, and in doing so construct a less greedy approximate factorization algorithm. One limitation, however, is that the convergence and runtime depend on needing to define 2-point Givens rotations whereas MMF employs rotations of higher order.

5.3 Multiresolution Analysis Using Collections of Trees

Having discussed the collections of trees model from a hyperbolic embedding standpoint in chapter 3, we also consider the model from an MRA standpoint. It is plainly clear that there are deep roots connecting tree metrics with multiresolution bases.[13] Several widely used methods make use of the dendrogram or spanning tree as their key tool for identifying graph structure at different scales. Inevitably, though, the sheer number of edges removed from a graph in order to define a tree subgraph deletes valuable information regarding graph

13. Pun certainly intended.

(a) P_8: Path graph of length eight



(b) The dendrogram for one run of algorithm 4

Figure 5.9: Example of a path graph and its dendrogram.

features. We consider a particularly simple tree-based approach to wavelet construction to make this point and go into a bit more detail than was done is section 2.3.3.

5.3.1 Constructing a Simple Spanning Tree Wavelet Basis

The first approach to using a tree metric (an ultrametric, specifically) for wavelet construction comes from the work of Murtagh who defined Haar wavelet transforms on dendrograms.[106] A dendrogram of n nodes has no more than $n-1$ merges of branches. We restrict the dendrogram to be a binary tree, consisting of merges that are comprised of no more than two nodes.[14] At each merge m_i, we define a graph function that is positive a_i on all nodes in the left branch and negative b_i on all the nodes in the left branch. The values of $a_1, \ldots, a_7, b_1, \ldots, b_7 \in \mathbb{R}$ are defined such that the norm of the function is one and their sum is zero. Consider the example in figure 5.3.1.

14. In general, we may actually choose to say there are exactly $n-1$ merges for the sake of defining an orthogonal basis. If, for instance, a dendrogram merges nodes x, y, z at merge m_i, this merge may be split up as x, y at m_i and $(x, y), z$ at m_{i+1}.

The matrix whose columns are graph functions corresponding to merges in the dendrogram is:

$$\begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ -\frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{1}{3}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{1}{3}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{3}{8}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{3}{8}} & \sqrt{\frac{1}{8}} \end{pmatrix} \tag{5.5}$$

From merge m_1 (and correspondingly m_2, m_3, m_4 in the first four columns) we obtain the simplest basis functions. The next wavelet m_5 has larger support, combining two smaller merges. The first unbalanced function is m_6 and the last wavelet is m_7. We append the constant vector in the last column, which is the only remaining vector orthogonal to all the others. Any two vectors either have no overlapping support, or they do. In the former case they are obviously orthogonal. In the latter case, one vector is constant on the entire support of the other. Since the sum of all vectors is zero, these vectors are also orthogonal. This scheme for wavelet construction is based on using algorithm 4 to define a dendrogram for a path graph of length 8.[15]

5.3.2 Limitations of Tree-based Wavelet Transforms

This simple example can demonstrate the challenge of using trees to summarize (possibly hierarchical) structure. Whereas based on the graph, we would want any inference to assume v_6 is equidistant from v_5 and v_7, the random dendrogram that generated the clustering in figure 5.9(b) implies that v_1 is closer to v_6 than v_7. The simple example of the path graph

15. In the algorithm two sets $A, B \subset V(G)$ have the relation $A \sim B$ if $\exists v \in A,\ w \in B$ such that $(v, w) \in E(G)$.

Algorithm 4 Simple randomized hierarchical clustering algorithm

Require: A graph $G = (V, E)$ and a function $\text{get} - \text{nbrs}$
  S = $\{\{v\} \mid v \in V\}$
  mergeDict = [] {This is a dictionary that maps keys to values.}
  i = 1
  while $|S| \geq 1$ **do**
    Randomly select $s \in S$
    Select any $t \in S$ such that $s \sim t$
    Remove s, t from S and insert $s + t$
    mergeDict$[i] = s + t$
  end while
  return mergeDict

might be too simple because a path graph does not have any explicit structure at multiple levels, but we are nonetheless able to see how trees can delete valuable graph information. This is not a graph-specific problem, but arises the processing of signals on $\mathbb{R}$ as well.

For a very simple signal on the real line, a Haar wavelet basis may be defined similarly. Instead of thinking of the path graph as a graph, we can instead consider it eight equally-spaced observations along the real line. The Haar wavelet transform for this finite set of observations is defined below:

$$\begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & \sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ -\frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & \sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\frac{1}{2} & 0 & \sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\frac{1}{2} & 0 & \sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & \frac{1}{2} & -\sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & \frac{1}{2} & -\sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & -\frac{1}{2} & -\sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & 0 & -\frac{\sqrt{2}}{2} & 0 & -\frac{1}{2} & -\sqrt{\frac{1}{8}} & \sqrt{\frac{1}{8}} \end{pmatrix}$$

This wavelet basis corresponds to a slightly different dendrogram than that in figure 5.9(b). The inability of this basis to capture a signal that is local to the observations at v_4, v_5 (or in the case of the graph vertices v_6, v_7) is a shortcoming arising from a strict hierarchy required

for a proper, orthonormal multiresolution basis. Coifman & Donoho tie this issue to the fact that wavelet bases lack translational invariance (in contrast to the Fourier bases).[35] When a piecewise signal on $[0,1] \subset \mathbb{R}$ has a discontinuity at $\frac{1}{2}$, the HWT as defined above will perfectly capture the signal. If a discontinuity is at $\frac{1}{3}$, though, for example, it will lead to strong pseudo-Gibbs oscillations. The lack of translation invariance for the HWT means that the transform does not learn to shift its mother wavelet by $\frac{1}{3}$ to avoid the discontinuity. The solution suggested by Coifman & Donoho is called "cycle spinning."

5.3.3 Cycle Spinning on $\mathbb{R}$ and Graphs

In attempting to represent signals with wavelet transforms, signal discontinuities may leave simple signal representations like the HWT unable to capture strong local structure. *Cycle spinning* refers to introducing a translational shift to the mother wavelet so that the wavelet hierarchy aligns with local signal features. For a signal s indexed by $[0,1] \subset \mathbb{R}$, if the HWT is denoted by T^{Haar}, then $T^{\text{Haar}}s$ is a sequence of wavelet coefficients. The shifted HWT is T^{Haar}_{γ} and results in $T^{\text{Haar}}_{\gamma}s = S_{-\gamma}T^{\text{Haar}}S_{\gamma}s$ where $S_{\gamma}f$ is a shifting operator for functions, where $S_{\gamma}f(x) = f(x+\gamma(\mod 1))$. Coifman & Donoho suggest that wavelet transforms that are not translation-invariant may be circumvented by finding an optimal shift γ for a given signal or by considering all the γ-shifted HWTs for a set $\Gamma = \{\gamma_1, \ldots, \gamma_k\}$ and averaging the reconstructions for all of them. For a signal with a finite number of observations, the set Γ may consist of the difference between the x indices of all observations and the minimum x index. For the simple path graph example, this would consist of $\Gamma = \{0,1,2,3,4,5,6,7\}$. In the case of a continuous signal, they suggest either integrating the shifted HWT with respect to the uniform measure on $[0,1]$ (i.e. $\Gamma = \{x \mid 0 \leq x < 1\}$) or a finite sample of shifts from the interval [0,1] and approximate an integral using a Monte Carlo method.

We are interested in defining multiresolution bases for graphs that are robust to capturing many types of graph structure and signal varieties. It is therefore necessary to develop a tool for graph wavelet construction that accomplishes what cycle spinning accomplishes for tradi-

tional wavelet construction in $\mathbb{R}^n$. One of the most difficult obstacles in wavelet construction for graphs is determining the meaning of translations and dilations. Lacking these analogs, multiresolution analysis on graphs do not involve proper multiresolution bases according to the conditions of Mallat.[101] The path charted by previous work in multiresolution analysis has been to forego axioms of dilation and translation, instead defining graph wavelets in terms of successively coarse vector spaces. In the case of Murtagh's dendrogram-based Haar wavelets, wavelet definition is much simpler, having relegated all the hard work of structure discovery to a hierarchical clustering algorithm.

We note that translations are not important for multiresolution bases for the sake of being a translation, but for their ability to ensure that nearby elements are grouped in the same wavelet. Translations along the real line accomplish this by moving wavelet borders within a small region, ensuring that nearby elements are, at least sometimes, found on the support of the same local wavelets. In the approach of Coifman & Donoho, which considers HWTs with shifts sampled from a uniform distribution, the probability that two nearby elements are covered by the same wavelet, or that a discontinuity remains close to the boundary of a wavelet's support, is increased. We aim, then, to find a means of defining a set of graph HWTs that consider groupings of graph nodes subject to different directions of closeness.

Given that the original dendrogram defined in figure 5.9(b) is the output of a random hierarchical clustering algorithm, by running this algorithm K times, we collect K dendrograms and K ways of describing nodes in the graph that are close together. In the example of the path graph in figure 5.3.1, this approach would yield K dendrograms which summarize the path graph from different perspectives. In none of these K graphs is there a merge of nodes v_i, v_j unless $|i - j| = 1$. It is furthermore equally likely for v_{i-1}, v_i and v_i, v_{i+1} to be merged in the dendrogram. This means that, using a collection of randomly generated dendrograms, the structure of the path graph may be recovered. Clearly this is overkill for the case of the path graph, but for the ease of describing how the dendrograms are likely to look, it is a useful example. In analogy with the types of computational demonstrations

in Coifman & Donoho's cycle spinning work, the average reconstruction of a smooth signal on a path graph will improve in quality as a function of how many dendrograms are used. To take one more step, cycle spinning in the graph context as described above accomplishes something that cycle spinning for functions defined on $\mathbb{R}$ does not. Whereas shifting signals on [0,1] modulus 1 concatenates the signal at the neighborhood of 1 to the signal at the neighborhood of 0, the graph cycle spinning does nothing of the sort. Instead, every single random dendrogram represents an equally likely clustering.

We know of no other work that has adapted cycle spinning to graph wavelets other than that of Gavish *et al.* in defining wavelets on "trees, graphs, and high-dimensional data." The applied task in that paper is signal propagation for semi-supervised learning. In that task, they use the most popular label prediction outputted from their collection of random wavelet decompositions to make predictions.[63] Their work makes use of the concept of cycle spinning but does not explain the mechanics of how cycle spinning applies in a graph context. Furthermore, the only utility of cycle spinning in the work of Gavish *et al.* is a *post hoc* averaging (or voting) of a collection of models. We consider that the usefulness of cycle spinning in graphs in terms of the value of the wavelets generated (as in section 7). Lastly, we connect the use of cycle spinning to hyperbolicity, random forests, and matrix factorizations. We cover some of these connections in the next section.

5.3.4 Confluence of Signal Denoising, Random Forests, and Low-variance Estimation

We have seen the contribution of a collection of trees as a signal denoising tool as an outgrowth of the existing idea of cycle spinning. We now draw a connection between this graph-based application and another important foundational tool in machine learning - the random forest. Since its introduction in 2001, the random forest technique has been groundbreaking as an easy-to-implement, nearly parameter free model that tends to outperform nearly any other nonparametric method.[21] In its traditional setting, the random forest is

a collection of decision trees, each one having been generated from a random subset of features. When we consider the dendrogram a decision tree, with classes consisting of clusters of nodes, a collection of dendrograms may be viewed through the same lens.[16]

Having thus far discussed dendrograms, we aim to go even further by cutting the computationally expensive hierarchical clustering stage out of the process. We have already discussed in section 3.3 the surjective map of spanning trees to dendrograms. By considering the hierarchical clustering structure that may be drawn from a single spanning tree, we now have a three step process to obtain nondeterministic wavelet bases: (1) sample a spanning tree from the uniform spanning tree distribution, (2) define a dendrogram based on spanning tree, (3) define a wavelet basis based on this dendrogram. Zooming out, this pipeline is then viewed as a single thread, among many, making up the fabric of a random forest of tree classifiers.

Random forests are not the only conceptual foundation for using collections of trees for inference and prediction. A particularly insightful approach comes from Jebara & Long's work relating the tree-structuredness assumption for a dataset to a notion where the independence of observations in a dataset is termed tree-dependent identically distributed (TDID).[77] Influenced by early work on inference over distributions of trees,[76] the TDID assumption for a dataset $X = (X_i)_{i=1}^N$ makes an important change to the traditional IID assumption. Whereas the IID assumption for X conditioned on a parameter Θ would lead to the likelihood function

$$\text{Prob}(X_1, X_2, \ldots, X_N \mid \Theta) = \prod_{i=1}^{N} \text{Prob}(X_i \mid \Theta)$$

16. The analogy between decision trees and dendrograms is quite clear. Still, whereas the use of a decision tree in data analysis is meant to make predictions about new points, the graph setting is a bit different in that all nodes are available from the training stage. In what sense is a dendrogram a decision tree? If a partial labeling is known for the graph nodes, the unlabeled nodes' labels may be inferred by the labeling of the nodes they are closest to in the dendrogram. Alternatively, if a new node is added to the graph, its edge connections to existing nodes will indicate where it should be placed in the dendrogram's hierarchy. From multiple perspectives, the dendrogram yields a decision tree-like summary of the graph's information.

modeling the data using the TDID assumption leads to the likelihood function

$$\text{Prob}(X_1, X_2, \ldots, X_N \mid \Theta) = \prod_{i=1}^{N} \text{Prob}(X_i \mid X_{\text{parent}(i)}, \Theta)$$

After defining an edge-weighted, data-driven graph of X using any variety of similarity metric, a rooted tree of the graph will define $X_{\text{parent}(i)}$ for each X_i along with a parameter vector Θ. By making use of Kirchoff's Matrix Tree Theorem, evaluating the expectation or the maximum *a posteriori* estimate is no more difficult than computing the determinant of the graph Laplacian for the weighted graph Q whose entries are $Q_{ij} = \text{Prob}(X_i \mid X_j, \Theta)$. Computing the expectation of functions on a graph with respect to the measure of its spanning trees is an original approach to considering how trees are able to collectively summarize the tree-like structure of a graph.

Another interesting connection drawn between tree-likeness and a distribution of spanning trees is the Mixture-of-trees model.[103] After defining a likelihood function on trees as in a TDID model, a small sample of trees is selected, they are initialized with weights, and then tree weight parameters are used to train the mixture of trees to represent a dataset. The mixture of trees model is then used for density estimation and classification. In their discussion they note that, while individual tree models have fast, yet limited predictive power, the mixture of trees maintains the same computational complexity of tree-based inference while increasing (as a function of the number of trees) their flexibility. They also demonstrate in experiments that tree-based classifiers are reliably robust to irrelevant attributes, a feature that decision trees also exhibit in random forests. This work proves foundational to much subsequent research in the Bayesian community on assuming a dataset may be modeled with tree-like graph structure, or may be modeled by a collection of tree graphs.[19, 5, 78]

5.4 Multiresolution Approximations as Tools for Regularizing Tree-like Structured Data

The last perspective we consider on the role of trees and wavelets in the analysis of network data is that of regularization. Having previously discussed the role of the Laplacian operator in measuring function smoothness in section 2.3.2, we delve deeper into regularization on graphs and the role of wavelets therein.

5.4.1 Regularization Overview

In functional analysis, a very straightforward question is how to define the norm of a square integrable function $f : \mathbb{R} \to \mathbb{R}$. The integral defines the inner product in $L^2(\mathbb{R})$ so the norm is naturally

$$\|f\|^2 = \int_{x \in \mathbb{R}} f(x)^2 \, dx$$

For the purpose of this chapter, we choose to define the norm in different terms. Recall that when the Fourier transform $\tilde{f}$ of a function $f \in L^2(\mathbb{R})$ is known, the norm may also be defined as the norm of the transform:

$$\|f\|^2 = \|\mathcal{F}f\|^2 = \|\tilde{f}\|^2$$

which implies that

$$\|f\|^2 = \int_{\omega \in \mathbb{R}} |\tilde{f}(\omega)|^2 \, d\omega$$

When Fourier analysis is used for the purpose of smoothing signals, one approach is to shrink the high frequency portion of the Fourier transform to zero before taking the inverse transform. Suppose that the low frequency portion is denoted $P_L \subset \mathbb{R}$ and the high frequency portion is P_H with $P_L \cap P_H = \emptyset$ and $P_L \cup P_H = \mathbb{R}$. Define $\tilde{f}_{\text{smooth}} = \tilde{f}$ except that

$\forall \omega \in P_H,\ \tilde{f}_{\text{smooth}}(\omega) = 0$, i.e.

$$\tilde{f}_{\text{smooth}}(\omega) = \tilde{f}(\omega)\mathbb{I}(\omega \in P_L)$$

This smoothing operation has the effect that

$$\|\tilde{f}_{\text{smooth}}\|^2 = \int_{\omega \in P_L} |\tilde{f}(\omega)|^2 \, d\omega + \int_{\omega \in P_H} |\tilde{f}(\omega)|^2 \, d\omega = \int_{\omega \in P_L} |\tilde{f}(\omega)|^2 \, d\omega < \|\tilde{f}\|^2$$

The inverse transform of $\tilde{f}_{\text{smooth}}$ will differ from f by having none of the high frequency Fourier components, losing (to some extent) the ability to describe local variations. This is the reason we refer to $\mathcal{F}^{-1}\tilde{f}_{\text{smooth}}$ as a smoothed version of f.

It may be the case that, instead of passing f through this low-pass filter, we only intend to quantify the smoothness of f. Define the function $r(\omega) = \mathbb{I}(\omega \in P_H)$ and let

$$\|f\|_r = \langle \mathcal{F}f, r \rangle = \langle \tilde{f}, r \rangle = \int_{\omega \in \mathbb{R}} \tilde{f}(\omega) r(\omega) \, d\omega = \int \mathbb{I}(\omega \in P_H) \tilde{f}(\omega) \, d\omega$$

The seminorm $\|\cdot\|_r$ is thus measuring the degree to which there are high frequency components making up f's Fourier decomposition. Note that for this simple function r there is no difference between large and small ω - the integral over the entire region P_H is evaluated with equal measure dedicated to all regions. Functions f for which $\|f\|_r$ is small, or equal to zero, are primarily smooth, whereas when $\|f\|_r$ is large, the function is *rough*. A functional measuring the relative smoothness of a function or signal is called a *regularizer*.

Most regularizing functionals P_s in fact, like $\|\cdot\|_r$, are of the form

$$P_s f = \|f\|_s = \int \mid \tilde{f}(\omega) \mid^2 s(\|\omega\|^2) \, d\omega \tag{5.6}$$

If this functional is to penalize functions that are not smooth, then s must be a nonzero

function that increases as a function of $\|\omega\|^2$.[17]

5.4.2 General Regularization on Graphs

Using the graph Laplacian, we may define an analogous regularization procedure. For a graph G and a function on the graph $f : V \to \mathbb{R}^{|V|}$, $\|f\|$ is easy to define, since it is just a vector of finite length:

$$\|f\|_2^2 = \sum_{i=1}^{|V|} f_i^2$$

Using the language of linear algebra, we may define f in terms of a (trivial) basis $E = \{e_1, \ldots, e_n\}$[18] as

$$f = \sum_{i=1}^{n} f_i e_i$$

This allows us to define $\|f\|^2 = \langle f, f \rangle = \sum_i \sum_j f_i f_j e_i^T e_j = \sum_i f_i^2$. Especially in a graph setting, E does very little to explain any features of f beyond the trivially local neighborhood of each vertex. A well-known property of the graph Laplacian demonstrates its use as a difference operator:

$$f^T L f = \sum_{(i,j) \in E} (f_i - f_j)^2$$

One might ask - how does harmonic analysis explain what the Laplacian matrix is doing here? If L has the eigenvalue decomposition $L = P \Lambda P^T$, then

$$f^T L f = f^T P \Lambda P^T f = f^T P \Lambda^{\frac{1}{2}} \Lambda^{\frac{T}{2}} P^T f = \langle \Lambda^{\frac{T}{2}} P^T f, \Lambda^{\frac{T}{2}} P^T f \rangle = \left\| \left(P \Lambda^{\frac{1}{2}} \right)^T f \right\|^2$$

17. If it were not to increase as a function of ω then functions with medium levels of roughness would be penalized more than functions with large levels of roughness, so to speak.

18. e_i is the vector which is all zeros except for a one at index i.

Since P is matrix whose columns are orthonormal, we may express the vector f in terms of a weighted sum (by ρ_i) of the columns of P (p_i):

$$f = \sum_{i=1}^{n} \rho_i p_i$$

and then the square norm above may be simplified

$$\|\Lambda^{\frac{T}{2}} P^T f\|^2 = \|\Lambda^{\frac{T}{2}} P^T \sum_{i=1}^{n} \rho_i p_i\|^2 = \|\sum_{i=1}^{n} \lambda^{\frac{1}{2}} \rho_i P^T p_i\|^2 = \|\sum_{i=1}^{n} \lambda^{\frac{1}{2}} \rho_i e_i\|^2$$

$$= \langle \sum_{i=1}^{n} \lambda^{\frac{1}{2}} \rho_i e_i, \sum_{i=1}^{n} \lambda^{\frac{1}{2}} \rho_i e_i \rangle = \sum_{i} \sum_{j} \lambda_i^{\frac{1}{2}} \lambda_j^{\frac{1}{2}} \rho_i \rho_j e_i^T e_j = \sum_{i} \lambda_i \rho_i^2$$

This norm corresponds to the norm of a vector $P^T f$ which is f after the following change in basis:

$$\begin{pmatrix} e_1 & \cdots & e_n \end{pmatrix} \Longrightarrow \begin{pmatrix} p_1 & \cdots & p_n \end{pmatrix}$$

Depending on the degree to which f aligns with different vectors in the transformed space, those directions will be weighted by λ_i.

A Simple, Illustrative Example

In chapter 2, we considered the graph Laplacian operator through its analogy with the continuous Laplacian operator on a regular lattice. Now we approach regularization from a different direction, by observing the action of the operator in practice and inferring the meaning of its columns. We first assume that the diagonal entries $(\lambda_i)_{1 \leq i \leq n}$ of Λ are arranged in decreasing order. Consider the function f and another function g where $\forall i \geq 2$, $(Pg)_i = (Pf)_i$ and $(Pg)_1 = (Pf)_1 + 1$. Then we know that $f^T L f < g^T L g$ because

$$g^T L g = \lambda_1(\rho_1^2 + 1) + \sum_{2 \leq i \leq n} \lambda_i \rho_i^2 = \lambda_1 + \sum_{1 \leq i \leq n} \lambda_i \rho_i^2 = \lambda_1 + f^T L f$$

Since this increase of the magnitude of one direction in the transformed space increases the value of $f^T Lf$, we conclude that the sum of the squared differences between function values at adjacent nodes has increased. Not only has it increased, but it has increased through this modulation of 1 more than it would have increased had one been added to any other index of the vector Pf. The Laplacian operator is measuring the magnitude of the graph function in each of the directions of the transformed space, and weighing the contribution of these magnitudes differently. The components that increase the quadratic form $f^T Lf$ the most are those that are responsible for increasing the square differences among adjacent graph function values. The analogy between the continuous Laplacian operator and the graph Laplacian operator is now clear. Just as the regularizing function s in equation 5.6 was a nonnegative-valued monotone increasing function, so too the seminorm we obtain from $\|f\|_L = f^T Lf$ may be modified using a similar nonnegative, monotone increasing function to measure (or rather penalize) non-smoothness in different ways.[19]

$$\sum_i \lambda_i \rho_i^2 \Rightarrow \sum_i s(\lambda_i)\rho_i^2$$

Starting with defining kernels on graphs, Smola & Kondor define a principled family of regularization operators, and their corresponding kernels, by defining a class of regularizers that modulate the eigenvalues of the graph Laplacian in different ways. For instance, they demonstrate how, through different choices of s, one may produce the regularized Laplacian, diffusion process, p-step random walk, and inverse cosine regularization operators.[128] Other work has focused on developing alternative (though quite similar) graph regularization techniques.[18]

19. From a different perspective, one might be interested in measuring the similarity between two nodes on a graph, leading to defining graph kernels.

5.4.3 Regularization Using Tree-based and Hyperbolic Approximations of Graphs

Besides using the graph Laplacian, or functions of it, for graph regularization, the use of wavelet decompositions to approximate the action of Laplacian has strong precedents. For instance, SGWs have been used for SSL as an approximation to the traditional SSL with Laplacian regularization for classification[49] and regression.[126] In this section we delve into the interpretation of wavelet decompositions as tools for regularizing graph functions.

Suppose that the graph Laplacian L has eigendecomposition $L = P\Lambda P^T$. After approximating the graph using a dendrogram, such as using the technique described in section 5.3.1, a wavelet may be defined for each merge node in the dendrogram, as described earlier. The approximation to the graph Laplacian using this wavelet basis will be $L \approx WFW^T$. Since there is some error in WFW^T reconstructing the graph Laplacian, the seminorm $\|f\|_{W,F} = f^T WFW^T f$ will only approximate $\|f\|_L = f^T Lf$. To the extent that approximation is a good one, though, the wavelet-based seminorm will accomplish essentially the same thing as the Laplacian seminorm, computing squared differences between function values that are edge connected, i.e. measuring roughness. In the case of the wavelet-based seminorm generated from a dendrogram approximation, the extent to which it differs from L will depend on the reasonableness of the tree-like assumption. We return to the path graph example considered earlier in figure 5.3.1. For ease of presentation, we re-present the HWT

matrix W and frequency matrix F in lines 5.7 and 5.8, respectively.

$$W = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ -\frac{\sqrt{2}}{2} & 0 & 0 & 0 & \frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\frac{1}{2} & \sqrt{\frac{1}{12}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{1}{3}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{1}{3}} & \sqrt{\frac{1}{24}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & 0 & \frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{3}{8}} & \sqrt{\frac{1}{8}} \\ 0 & 0 & 0 & -\frac{\sqrt{2}}{2} & 0 & 0 & -\sqrt{\frac{3}{8}} & \sqrt{\frac{1}{8}} \end{pmatrix} \tag{5.7}$$

$$F = \begin{pmatrix} \frac{5}{2} = 2.5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{5}{2} = 2.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{5}{4} = 1.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{13}{12} = 1.08\bar{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{3} = 0.\bar{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{5.8}$$

In this very simple graph, it is evident that wavelets are also providing a notion of function smoothness. A function that changes value in the manner that p_1 does is certainly "rougher" than the function in p_7 or p_8. The dendrogram, though unable to capture the graph perfectly, provides a sense of distances and structure. Furthermore, just as we noted when discussing collections of trees, by considering the level of smoothness dictated by a large collection of dendrograms, we would obtain a better estimate than any component on its own.

One more perspective on regularization via wavelet approximation of L is in order. While

from one perspective we might say that the dendrogram is a coarse graph approximation (in case of the path graph it certainly is), we could also embrace the inexactedness of wavelet approximation as a form of regularization in and of itself. Instead of regularizing functions on the data, the tree model of the graph regularizes the data itself to assume a known structure. We make this point along the line of the same argument made in section 5.2.3 that a hyperbolic embedding or a tree-based similarity measure can dictate more reliable notions of node similarity than the original graph data itself. Relatedly, one work on SSL found that, when prior information existed about node clustering, by regularizing functions with respect to this clustering structure they were able to obtain better results than when the clustering information was ignored.[146] That work is acknowledging that the picture conveyed by a single graph of edge connections (even before any approximations) can misconstrue some relationships between nodes. Another related idea is present in anchor graph regularization (AGR). This form of regularization estimates functions by measuring their roughness with respect to a smaller set of "anchor nodes" instead of the complete graph.[97, 137] AGR is primarily used for scaling graph-based learning to larger datasets, but depending on how the anchors are chosen, it may be seen as leading to a similar result as the structural graph assumptions we have presented.

Through graphs' muddling the waters separating features and response, discretized nodes in graphs provide an interesting counterpoint to continuous metric spaces. For a standard dataset living in Euclidean space the data have an exact position indicated by the features and the response is then a labeling of these positions. Then, through feature selection or other forms of dimensionality reduction, irrelevant features may be identified and discarded. In a graph setting, however, there is only one variety of features - edge connections. In the setting of label propagation, if any extraneous features (i.e. edges) are present in the graph, the practitioner depends on the strengths of the signal being strong enough to overcome the noise introduced by these edges. Our approach takes a different route. We see the structural assumption of tree-likeness akin to a dimensionality reduction tool for graphs, connecting

the worlds of regularization and hyperbolic embeddings.

Dimensionality Reduction Example

To test MMF's role as a dimensionality reduction tool, we conduct simple experiments testing its ability to recover a tree-like structure subject to a noise corruption. The most conservative claim that can be made about MMF is that it is a numerical approximation tool. When there is a covariance matrix too large to be inverted by traditional methods, MMF approximates a matrix with an WHW^T factorization, where the columns of W are sparse basis functions and H is a core-diagonal matrix. After WHW^T is computed, this factorization makes it possible to compute the matrix's approximate inverse in linear time. Approximate methods are important for scalability, but we make a stronger claim than numerical approximation.

Consider a graph $G = (V, E)$ with strong clustering structure, such as that it may be modeled by a SBM.[20] To model a less stable tree-like structure, we first consider k spanning trees of G.[21] We then define a new graph $G^{\text{tree}} = (V, E^{\text{tree}})$ where $E^{\text{tree}} = \cup_{1 \leq i \leq k} E(\text{ST}_i(G))$.[22] In the case when $k = 1$, G^{tree} is simply a spanning tree; when $k > 1$, G^{tree}'s edge set is simply the union of the edge sets of the respective spanning trees. We consider the subgraph G^{tree} where there is not-as-strong clustering structure (since many elements of $E(G)$ have been deleted) as also tree-like, as it is the same graph with weaker clustering structure. Inference on G^{tree}, such as through label propagation, is more difficult than inference on G, and we choose to use G^{tree} as our test case. This very tree-like graph is contrasted with the Erdös-Renyii graph G^{ER} whose parameter is fixed so that $\mathbf{E}(|E(G^{\text{ER}})|) = |E(G^{\text{tree}})|$.

Graph Laplacian reconstruction and label inference is done on a graph that is a mixture

20. For instance, assume that G has two equally sized clusters C_1, C_2, and that $\forall v, w \in C_i, \text{Prob}\left((v, w) \in E(G)\right) = p_i$ and $\forall v \in C_1, w \in C_2$, $\text{Prob}\left((v, w) \in E(G)\right) = q_{12}$.

21. By less "stable," we mean less robust to noise. For a graph with distinct clusters that obviously have strong community structure, the addition of new edges independent of the clustering will have a smaller impact than if the clusters are only weakly connected. For the merger of a small number of spanning trees, on the other hand, the addition of a few random edges may wildly change the estimated distances between any two nodes.

22. $\forall v, w \in V(G^{\text{tree}})$, $d_{G^{\text{tree}}}(v, w) \leq \min_i d_{\text{ST}_i(G)}$.

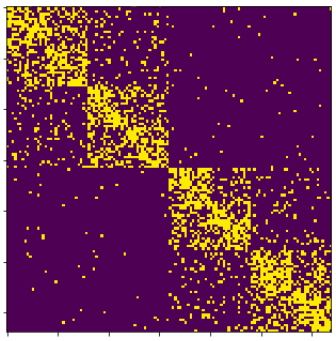
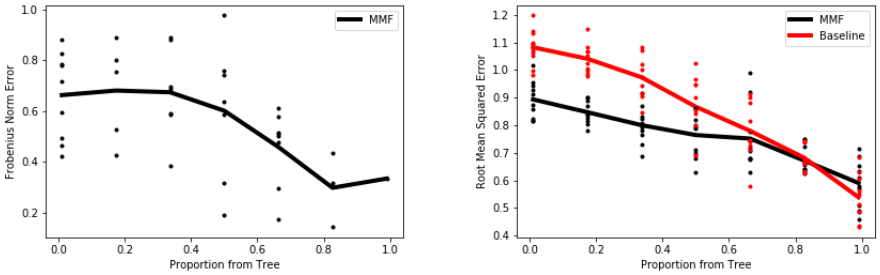


Figure 5.10: Sales-Pardo adjacency matrix

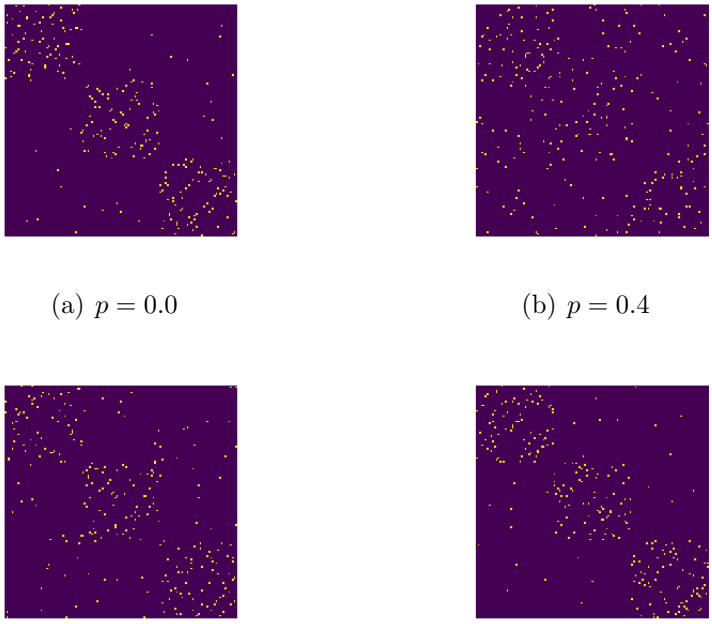


(a) (b)

Figure 5.11: Performance of MMF as a tree-structure regularizer

of G^{tree} and G^{ER}. Specifically, a new graph H_p is defined, where p is the probability that any given pairwise connection among the nodes of V is decided using G^{tree} and with probability $1-p$ it is decided using G^{ER}. p, then, serves as a tuning parameter to control how much tree structure is present in the graph under consideration. This parameter p is the independent variable in figures 5.11 where higher values of p correspond to graphs that are more tree-like. A selection of several graphs for different values of p is shows in figure 5.12. In these examples, $k = 3$.

The takeaway from figure 5.11(a) is that, depending on the graph, MMF will have different degrees of matrix reconstruction error. For graphs where $p \approx 1$, H_p is essentially a copy of G^{tree} which is a sparsification of G; graphs of this variety naturally match the structure that MMF will impose. For $p \approx 0$, on the other hand, H_p is essentially G^{ER} and the MMF algorithm will ultimately enforce a tree-like structure for a graph that is random. From a numerical approximation perspective, there is not be a difference between approximating



(a) $p = 0.0$ (b) $p = 0.4$

(c) $p = 0.8$ (d) $p = 0.9$

Figure 5.12: Different mixtures of G^{tree} and G^{ER}

random or structured matrices - the goal is only scalability. For a dimensionality reduction tool, though, the tool's loss of information will be different whether the data at hand lives in a low dimensional manifold or not.

The features of this plot acquire additional relevance in light of the results in figure 5.11(b). We consider a label propagation task where the labels of 30% of the nodes of H_p are used to infer the labeling of the remaining nodes. The "ground truth" function is defined to be smooth with respect to the community structure of the SBM G, from which H_p is derived. The unobserved labels are inferred by assuming that the labeled function may be modeled with a Gaussian Process whose covariance matrix is $\mathcal{L}_{H_p}^{\dagger}$. The baseline method uses $\mathcal{L}_{H_p}^{\dagger}$ for inference while the MMF method uses the MMF's wavelet transformed reconstruction of $\mathcal{L}_{H_p}^{\dagger}$ for inference. For large values of p, where little information about G has been obstructed by noise, the two approaches are essentially the same, with the baseline approach slightly outperforming MMF. For lower values of p, the difference between the

approaches is stark, with the MSE from MMF's inferred labeling being remarkably lower than the baseline method! The usual motivation for using approximation methods is for scalability, but in this experiment we find that using an approximate method is actually improving inference relative to an exact method. This reorients our interpretation of MMF, indicating that there are scenarios when it is advisable to use this wavelet transform instead of an exact method. All the more so in a regime of large datasets, the use of MMF promises the benefit of scalability along with enforcing a hierarchical structure on the data for inference tasks.

5.5 Summary

In this chapter we have provided a context for understanding the multiresolution analysis on graphs using the MMF and collections of trees. The foundation for the MMF-generating algorithm depends on sequences of rotations of matrix indices which correspond to node indices. By modeling the sequence of rotations using a directed graph, the sequence of rotations may be matched to a hyperbolic embedding. Connecting the MMF algorithm to a hyperbolic embedding provides a means for assessing when MMF is likely to work and when it will not. We empirically find, furthermore, that by selecting which nodes should be rotated together using tree-based metrics, the accuracy of MMF reconstructions is improved. The key advantage of said reconstructions is that they translate into better inference even when existing tree structure is corrupted by noise. We discovered a surprising property of MMF-based inference, where, for certain graphs, labelings inferred using MMF were actually more accurate than those obtained with traditional, exact methods. All together, these findings support the hypothesis that many datasets may be modeled as having hierarchical structure, which in turn leads to beneficial results when using MMF. Relatedly, many applications of hierarchically-structured models can face limitations similar to those encountered in applications of multiresolution analysis for signal processing. By adapting the idea of cycle spinning from the traditional signal processing context to graph signal processing, these limitations

are dampened significantly.

CHAPTER 6

COMMUNITY DETECTION USING RESISTANCE NETWORK MODELS

6.1 Background on Seed Sets

Nascent abilities to collect progressively more massive amounts of data continually motivate new approaches to large-scale data analysis. Recognizing that many data collection pursuits involve unstructured data, and that the data collected is of a scale so large that it is infeasible for a human being to label training data with their associated classes, the fields of unsupervised learning and semi-supervised learning gain new importance.[153, 17] We focus on a contemporary challenge in the class of community detection (CD) problems. The aim of CD is to leverage the known features of a complex system - namely its nodes and edges - to estimate unobserved functions, such as class labelings, on the network. The foundation of network analysis in general and CD in particular is that the nodes and edges of a graph convey valuable information, information that can be deployed for practical applications. In the last decade, the CD problem has motivated solutions in a constantly growing set of fields ripe for application.[62, 121, 61]

A proliferation of work in CD has given rise to the formulation of several distinct problems involving the detection of communities that have different starting assumptions and objectives. Namely, given a graph $G = (V, E)$, one type of community detection aims to develop a full partition of the nodes V into non-overlapping groups.[12, 58] Another type acknowledges that networks will not always admit partitions into disjoint subgraphs and so its objective is to detect possibly overlapping communities, maximizing graph coverage.[112, 143] A less ambitious type is the detection of a single community. This problem begins with a predefined set of seeds which are fixed to be members of the sought-after community.[102, 145, 73, 135]

In addition to the many forms of CD, the number of scoring functions by which these methods are evaluated has also grown rapidly. Central to any approach to CD is the speci-

fication of a scoring function to assess the degree to which community structure is present. The most popular and effective scoring functions are based either on maximizing triadic closures, i.e. the local clustering coefficient, or on minimizing conductance.[145]

In this chapter, our work focuses on improving the state of the art in CD for seed set expansion around a single node, or a small set of nodes. A dominant force in this problem setting is a CD-oriented version of PageRank deemed "personalized PageRank" (PPR). Whereas the original PageRank algorithm is a ranking of nodes' importance (using PageRank contributions) in a general graph, PPR focuses on ranking nodes with scores based on their proximity to a pre-defined seed set.[14, 140, 84] Our contribution builds upon existing applications of PageRank by introducing an effective resistance-based *germination stage* before the propagation of PPR weights begins. Specifically, we propose a two-step approach using PPR on a preprocessed vector that had weight distributed over a new, revised seed set. This set includes the original seed nodes, but it also adds a selection of nodes that are chosen to minimize the effective resistance (ER) diameter of the revised seed set. The stopping rule for identifying the respective points at which the germination stage and PPR end is based on community scoring functions.

The remainder of the chapter is organized as follows: in section 6.2 we review the role that random walks play in the theoretical underpinnings of PageRank and ER. The manner in which these two measures of node similarity differ is illustrated and the rationale for valuing ER in CD is presented. In section 6.3, we present the germination stage algorithm along with its time complexity. We also make explicit the connection between our algorithm and minimizing the ER diameter. In section 6.4, we provide synthetic and empirical data experiments to show the efficacy of our two-step approach in practice. We conclude in section 6.5 with a summary of our contribution and by outlining the new ideas that this work motivates, anticipating developments in the near future for CD.

6.2 Random Walks: Tying Random Walks, PageRank, and Effective Resistance

6.2.1 Problem Statement

Given a seed set $S \subset V$ and assuming some ground truth community $C \subset V$ exists such that $S \subset C$, we are interested in estimating a community $\hat{C} \subset V$, such that $S \subset \hat{C}$ and $\hat{C}$ minimizes a loss function measuring community structure. S may be a set of a single vertex in V or a collection of vertices. Whereas some CD scenarios, such as the "overlapping communities" problem, aim to achieve a high proportion of coverage on the entire network, the aim of this seed set expansion problem is to find a single high quality community, subject only to the constraint that it cover the seeds. In this sense we are addressing a one-class classification problem.

There are two components to addressing this problem. The first is to define the objective function to be minimized which quantifies the quality of community structure. The second component is to achieve this minimum, or to approximately achieve it. Take for instance the scoring function $f_c : S \subset V \to \mathbb{R}$ which measures the conductance of a subgraph. The global objective function based on conductance is:

$$C^* = \text{argmin}_{C \subset V} f_c(C)$$

This problem is NP hard, and so we must find some other recourse towards identifying a viable solution. Thus begins the second component of CD, devising a procedure for actually finding a subgraph with strong community structure.

6.2.2 Random Walks and Community Detection

Random walks on graphs are a fundamental key to understanding the relatedness between vertices. In a network analysis setting, it is frequently assumed that besides the edge set

there are no known vertex-associated features. So, we are left to assume that any nodes' interrelatedness is determined by whether they are edge-connected, if their shortest path distance is small, or if it is "easy" to travel from one node to another. This underlies the popularity of random walk-based kernels in graph-based learning problems.

It is natural to consider the wealth of information that random walks can provide for any problem in CD. In the absence of a single, agreed-upon definition for community structure, the basic idea of "more relatedness inwards than outwards" aligns well with the fact that a random walk after k steps is more likely to be at any individual node in its community than it is to be at any node in any other community. There are several manners in which the role of random walks may be formalized. One popular example is based on the PageRank vector. It produces an estimate of the stationary distribution of a random walk which starts at a set of chosen nodes. If there is truth to the idea that there is significantly more interrelatedness inside a community than at its boundary, then this PageRank vector should assign nodes inside the community higher scores than those which are outside of it. A different formulation is the commute time $C_{v_i v_j}$ between two nodes. The random walk interpretation of this quantity is the expected amount of time for a random walk beginning at v_i to arrive at v_j and then return to v_i. If at any stage of a random walk it is more likely for the walk to stay inside a community than move to a different one, then commute times within a community will typically be shorter than those which must cross the community boundary.

6.2.3 Personalized PageRank

The PageRank algorithm was originally introduced as a means of providing rankings for the pages of the World Wide Web.[111] In the two decades since its introduction, it has evolved as a technique for a wide set of problems in a number of applied fields. The specific development we focus on here is personalized PageRank (PPR), a variant of the PageRank algorithm where the initial mass is fixed at a single node or subset of the nodes $V_o \subset V$ (with

zero mass elsewhere).[55, 84] This translates to an initial zero vector $x \in \mathbb{R}^n$, except that $\forall k \in V_o,\ x_k = \frac{1}{|V_o|}$. The PPR vector is then calculated based on this initial state; the degree to which two nodes $v_i, v_j \notin V_o$ are more or less similar to the seed set V_o is governed by means of their respective indices in the PageRank vector x_{v_i}, x_{v_j}. This method is equivalent to finding the steady state of a random walk with restarts whose mass is initially distributed evenly at the set of nodes which comprise the seed set. Node v's score thus quantifies how likely it is that a random walk which starts at the seed set will end at node v.

This procedure translates into a CD scheme when the indices of the PageRank contributions are listed in a vector x_{PageRank} (after having listed the indices of the vertices in V_o) in the order of their score ranking.[13] The next step in relating this vector to a community is minimizing a scoring function $f : W \subset V \to \mathbb{R}$ which measures the strength of community structure. The chosen community is then $x_{\text{PageRank}}[1 : k_o]$ where $k_o = \text{argmin}_{1 \leq k \leq n} f\left(x_{\text{PageRank}}[1 : k_o]\right)$.

6.2.4 Effective Resistance

By interpreting a general graph as an electric network, one can study graph properties by considering how electricity flows throughout the network. All edges in the graph are considered resistors (in the case of an unweighted graph, they have equal resistance) and the relationship between the resistance of an edge r_e, the electric current over the edge f_e, and the electric potential difference between the nodes connected by said edge $v_{e^+} - v_{e^-}$ are governed by Ohm's Law:

$$f_e = \frac{v_{e^+} - v_{e^-}}{r_e}$$

That is, for a unit of current to flow across an edge, the potential difference at the edge's endpoints must be equal to the resistance. Summarized in matrix form, where $p \in \mathbb{R}^n$ is a vector of potential differences across edges and $B \in \{-1, 0, 1\}^{m \times n}$ is an incidence matrix of G, the flow over every edge is $f = Bp$. A second property of an electric network is the flow

conservation property. If a unit of electric current is sent from node a to b, then at any vertex $v \neq a, b$ in the graph, the sum of the flow of all edges connected to v is 0. Summarized in matrix form, where δ_i is the Dirac delta function, $B^T f = \delta_a - \delta_b$. Noting that $BB^T = L_G$, the graph Laplacian of G, and using Ohm's Law with the flow conservation property, we find that

$$B^T Bp = L_G p = \delta_a - \delta_b \iff p = L_G^{\dagger}(\delta_a - \delta_b)$$

This last formula provides a means of deducing the potential differences across all nodes for a given electric flow across the network. For a single unit of current to flow across an edge, the resistance of the edge must be matched by the potential difference of its endpoints. When the flow of $\delta_a - \delta_b$ is realized on the edge set of G, both the potential difference between a and b and the associated effective resistance (ER) between a and b will be $p[a] - p[b]$; the ER of edge e is

$$r_{ab}^{\text{eff}} = (\delta_a - \delta_b)^T L_G^{\dagger}(\delta_a - \delta_b)$$

Said in a different way, seeing a graph's edge set as a collection of resistors, if the resistors were replaced by a single resistor that acted indistinguishably from the collection subject to any flow vector - in terms of the amount of current and potential difference between edge endpoints - the amount of resistance of said single resistor is r_{ab}^{eff}. Although the ER does not only summarize the relationship between nodes that are connected by an edge, for the purposes of this investigation, it is only the ERs between nodes connected by an edge that will be used directly.

Another useful property of ER in an unweighted graph is that the ER between any two nodes that are edge connected is exactly equal to the probability of those two nodes being edge connected in a random sample from the uniform spanning tree distribution of G. Unfortunately, to reproduce the elegant demonstration of why this is the case would require prohibitively many lines of formulae. This fact in particular provides some useful intuition as to why the ER would be a helpful tool in CD. Suppose a graph has two distinct subgraphs

which are complete, but do not connect to each other except through a single edge. The ER of the edge which connects the two clusters will be very high (it will in fact be equal to one) because it is not possible for a spanning tree of the graph to exist without including this edge. The ER of an edge between two nodes in the same cluster will be comparatively small, because there are a number of spanning trees of the graph that do not make use of this edge. The ER of edges is thus a reasonable means of detecting a graph cluster's boundary because the edges which traverse that boundary have higher ERs. One last popular perspective of ER is that the expected number of steps of a random walk beginning at v_i, reaching v_j, and then ending at v_i (commute distance $C_{v_i v_j}$) is proportional to the ER (with constant $2m$).

6.2.5 Limitations of Effective Resistance

While these facets of ER provide a solid grounding for using it as a tool to understand nodes' interrelatedness, the behavior of ER described above does not necessary scale to large graphs. Specifically, it has been shown for classes of random geometric graphs - specifically kNN-graphs, ϵ-graphs, and Gaussian similarity graphs - that as the size of the graph $n \to \infty$, $C_{v_i v_j} \approx \frac{1}{d_{v_i}} + \frac{1}{d_{v_j}}$, the sum of the inverse of their degrees.[99] Luxburg *et al.* (2010) describe in their paper "Getting Lost in Space" that as the graph size grows, the number of paths between any two nodes increases and a random walk will tend to "forget" where it began, ultimately depending only on the degrees of the points at which it starts and ends. This limitation has in fact motivated a number of modified resistance-based graph distances that circumvent the scalability issue.[99, 109] ER's inability to scale to large datasets carries two significant caveats involving graph types and which scenarios the convergence to $\frac{1}{d_{v_i}} + \frac{1}{d_{v_j}}$ is weakest. Regarding the first point, the convergence results are proven only for graphs with a minimal degree that grows with the graph size. This is not an unreasonable assumption in kNN- or ϵ-graphs generated from data, but it is not an assumption that matches the properties of empirical networks. Secondly, the limitations of ER as a graph distance metric are only shown to be problematic for summarizing global structure, such as for nodes that

are not close in the shortest path distance. CD is primarily concerned with local properties of real-world graphs with strong power law distributions. Furthermore, the ERs that will be used in this study will concern only the ER between nodes that are edge connected, i.e. very local.

6.3 Germination: a Two-step Community Detection Algorithm for Seed Set Expansion

While many algorithms will address an unsupervised CD problem by first establishing seeds and then propagating from those seeds, in the semi-supervised context, the seeds are provided *a priori*, and propagation from those seeds is what the community detector must accomplish. The algorithm we present here takes an alternative route. We introduce what we term a *germination stage*. Just as the germination of a seed is its first step departing from seed form towards becoming a full-fledged plant, the germination of seed sets in CD converts the seed set into a more larger, more reliable set of nodes, from which a community may be cultivated. In this stage, the information contained in the initial seed set is culled and expanded - not to the expected community estimate, but to a richer seed set for the second stage of propagation. Our approach can be compared in spirit to optimization methods which provide a minimization algorithm with a "warm start."

After this first stage we take the traditional route of propagation using personalized PageRank. Our motivation for splitting the CD procedure into these stages is based on the observation that the better the quality of the seeds used for propagation, the better the output. Basing the first stage of seed germination on ER is a deliberate choice based on the theoretical properties of ER. Additionally, empirical evidence suggests that a greedy method using a resistance-based metric will initially choose community candidates at a lower false positive rate than those that would be chosen based on the PPR score vector.

6.3.1 Finding the Set with Small Effective Resistance Diameter

While the basic procedure of germinating a seed set and then propagating its information is quite general, we outline one specific implementation of this approach using the tools of ER and PageRank. The first step is to calculate pairwise ERs for every element of E. This can be done in two ways. The first is the inversion of the graph Laplacian. ER for any pair of nodes $u, v \in V$ can be computed from the inverse Laplacian as

$$r_{uv}^{\text{eff}} = L_{uu}^{\dagger} + L_{vv}^{\dagger} - 2L_{uv}^{\dagger}$$

The second approach is to sample uniform spanning trees from the graph at hand. The ER of an edge can be estimated as the proportion of the finite sample of spanning trees wherein said edge occurs. Depending on the situation, one of these two approaches might be preferable. For sparse, diagonally dominant matrices, fast solvers exist which can find r_{uv}^{eff} in $O\left(n\log(n)\log(\epsilon^{-1})\right)$ where n is the number of nonzero entries and ϵ is an accuracy level.[120] Spanning trees may be sampled (without any restriction on the type of graph) in $O(n^{\frac{5}{3}}m^{1}3)$.[46] The practitioner should thus decide which approach to use based upon the degree of sparsity, the required accuracy, and the ease of implementation. Furthermore, we are intrigued by the possibility of sampling spanning trees in a more efficient manner that guarantees an approximate estimate of the ER with high probability. The outcome of these procedures will lead to an estimate of the ER for every edge that appears in the graph (in the case of spanning trees) or for every pairwise relationship between nodes (in the case of graph Laplacian inversion). For the purposes of our algorithm, though, it is sufficient, even in the second approach, to store and use only the ERs between nodes that are connected by an edge in G.

A greedy algorithm is then implemented to progressively grow the seed set with the objective of minimizing the ER diameter at every stage. The specifics are found in Algorithm 5, where R denotes a sparse matrix containing ERs between nodes that are connected by an

edge, seeds are the initial seeds provided, and f is a community structure scoring function, namely conductance.

Algorithm 5 Greedy Minimizer of ER Diameter

Require: R, seeds, f {P}roduce a useful ordering of the elements of V w.r.t. seed nodes
  cur_comm $\leftarrow$ seeds
  unused $\leftarrow V -$ seeds
  $s = \{\}$
  while len(unused) $> 0 \ \wedge$ checkStop(s) **do**
    $i = \text{argmin}_{k \in \text{unused}} (\min(R[k, \text{cur\_comm}]))$
    $v = \text{unused}[i]$
    cur_comm = cur_comm $\cup \{v\}$
    $s = s \cup f(\text{cur\_comm})$
    unused = unused $\setminus \{v\}$
  end while
  return cur_comm

Two points in the algorithm must be clarified: the stopping condition, and the complexity of line 5. Regarding the stopping function, we employ a procedure introduced by Yang *et al.* (2015) who conduct a "sweep" of the PageRank vector and calculate the successive conductances of larger and larger subgraphs around the seed nodes; they consider the points where local minima occur to be community candidates. Since our objective in this stage of the algorithm is to preserve the purity of the germinated seed set as well as possible, we avoid false positives by always taking the first local minimum as a stopping point in the algorithm. Another convention of Yang *et al.* (2015) was to disregard any local minima for which the scoring function does not subsequently increase 20% before arriving at another local minimum. Since our procedure for identifying a local minimum must control the false positive rate in the germination stage, we use a stricter criteria of 5%. In any case, this means that the cur_comm returned by Algorithm 5 must be trimmed back to the point at which the local minimum actually occurred and cannot be used as is. Though it has not been observed to occur in practice, if it were the case that no local minimum meets this criteria, then the entire set of nodes would be returned.

Next, regarding the time complexity for an algorithm involving a search for the argument

of the minimum of a set of values each of which is the minimum of another list, we show in Theorem 3 that the time complexity is $\mathcal{O}(1)$. We assume at the onset that the graph at hand has a maximum node degree of d, which grows at a slower rate than graph size. Similarly, we assume that the maximum community size n_c is not a graph property that grows at the same rate as graph size. These are basic features of the Lancichinetti-Fortunato-Radicchi (LFR) graph model.[90]

Theorem 3. The time complexity of Algorithm 5 is $\mathcal{O}(1)$.

Proof. In general, finding the minimum of a list of length n is of complexity $\mathcal{O}(n)$. In the worst case scenario of line 5 (when the graph is complete) the number of required operations would be:

$$\sum_{a=1}^{n-\text{len(seeds)}} a(n-a) = \sum_{a=1}^{n-\text{len(seeds)}} an - a^2$$

Since len(seeds) is a small fraction of the entire community, this term may be approximated as $\sum_{a=1}^{n} an - a^2 = n\sum_{a=1}^{n} a - \sum_{a=1}^{n} a^2$. In this formulation, both sums are on the order of n^3. In practice, we observe a local minimum to occur very early, after only considering $\approx \frac{n_c}{10}$ (though the actual proportion would depend on the parameter which describes the power law distribution of the vertices' degrees). But even if we are to consider the algorithm to run for n_c steps, the total operations remain: $\sum_{i=1}^{n_c} id = d\frac{n_c(n_c+1)}{2}$. So, under our assumptions that we are in a sparse graph setting, where community sizes do not grow at the same rate as the number of nodes in the graph, the complexity of Algorithm 5 is $\mathcal{O}(n_c^2) = \mathcal{O}(1)$. □

6.3.2 Theoretical Properties of the Germinated Seed Set

The objective of finding a subgraph which contains the seeds and also has the smallest ER diameter is the problem we wish to solve. There is even some recent work in graph clustering which attempts to find these subgraphs efficiently. One recent paper introduced a polynomial time algorithm to partition a graph into sets of nodes which have a bounded ER diameter.[9] This approach suggests an important inroad into graph clustering based on ER,

but is not directly applicable to identifying a single community. It is also worth considering that the solution to the relaxed minimization problem is extremely fast and does not add much computational overhead to the use of PPR. That is, even with the computation of ERs, the germination of a seed set, and subsequent PPR, the overall complexity is still sub-quadratic. Furthermore, by demonstrating the efficacy of even an incredibly greedy algorithm for minimizing ER diameter, we aim to motivate more advanced methods for finding low ER diameter subgraphs which could be coupled with a subsequent stage of PPR.

That being said, we demonstrate that a bound exists at every step of the while loop of our algorithm. Specifically, the increase of ER diameter at each step is bounded above by the minimum of the ERs between all vertices already included and the newest addition.

Theorem 4. At each stage of the while loop in Algorithm 6.3.1, the increase of the ER diameter of cur_comm is bounded above by the minimum of the ERs between the newest node and the nodes in the current state of cur_comm.

Proof. It must be noted first that $r^{\text{effRes}} : V \times V \to \mathbb{R}$ is a distance metric and that (V, r^{effRes}) is a metric space. Suppose that at step i the newest candidate for community membership to cur_comm is $w \in V$. This could be the case only if $\forall u \in V$,

$$\min\Big(\{r^{\text{effRes}}(u,v) \mid v \in \text{cur\_comm}\}\Big) = \min\Big(\{r^{\text{effRes}}(w,v) \mid w \in \text{unused},\ v \in \text{cur\_comm}\}\Big)$$

This in turn means that

$$\text{diam}^{\text{effRes}}(\text{cur\_comm} \cup \{w\}) \leq$$

$$\text{diam}^{\text{effRes}}(\text{cur\_comm}) + \min\Big(\{r^{\text{effRes}}(u,v) \mid v \in \text{cur\_comm}\}\Big)$$

□

6.4 Experiments

6.4.1 Data Description and Approach

Three different forms of network data are considered below. The first type is a network sampled from a hierarchically-structured stochastic block model (HSBM). The matrix of probabilities along with a sample realization are depicted for ease of visualization in Figure 6.1. This is not a network structure commonly found empirically, but we use this model as a demonstration for the way an ER method works and how we expect it to work in a general network.

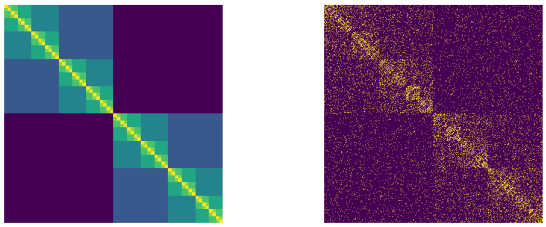


Figure 6.1: A hierarchical SBM and its realization

Next we present synthetic network data generated from the LFR graph model, a more realistic network model than the HSBM.[90] This graph model assumes all the graph's vertices' degrees are distributed according to an exponential distribution. Depending on the parameter of this distribution, graphs may vary from a sparse "hub and spokes" structure to a single cluster with a nearly uniform degree distribution. Using this data we show that the ER component of our method behaves differently depending on what proportion of the community is a high degree node. We also demonstrate that as long as the size of the community grows at a slower rate than the graph as a whole, the performance of our method does not degrade as a function of graph size. This is a specific concern of ours considering the limitations of using ER as a global similarity metric in large networks, as described in

Section 6.2.5.

Lastly, we consider several large data benchmarks for CD. We selected some of the most commonly used networks with labeled ground-truth communities. Though these labelings are not perfect, these benchmarks elicit the closest match to real-world applications available. The specific datasets we consider are the blogs hyperlink network, Amazon co-purchasing network, the DBLP computer science co-authorship network, the Youtube social network, the Orkut social network, and web-linkages between Wikipedia pages.[94]

While many CD methods are validated by juxtaposing an estimated partition with the ground-truth community partition, in the one-community example considered in this paper, the objective is to maximize a combination of precision and recall for the single estimated community. We quantify our method next to traditional PPR using F1 scores. While the comparison of a community estimate to the ground truth is ultimately based on the F1 score or other scores calculated from precision and recall, we also provide a complete precision-recall curve for the entire sequence of the cur_comm vector in order to profile whether a higher precision method is simply more conservative, or if it is uniformly outperforming a competitor. After all, the objective of our study is not specifically to produce an out-of-the-box cutting edge CD scheme, but to demonstrate how much the seed germination stage contributes before PPR begins. For this reason, our benchmark in these experiments is a vanilla PPR algorithm whose only parameter is α, which determines how much weight is redistributed at the initial seed nodes in each iteration.

To conduct each experiment we select a community from the ground truth labeling, and select a fixed number of seeds in that community. Then, we provide the network's edge set and its chosen seeds to the CD algorithm. When results are averaged, this reflects our having run the experiment independently on several randomly sampled synthetic networks, or our having queried either a community estimate around different seeds or a different community label in each iteration.

6.4.2 Hierarchical Stochastic Block Models

The basic intuition that a network node will have lower resistance towards a high degree node than a low degree node is put to the test by this graph model. We show that in the extreme scenario where node degrees are distributed uniformly and there is sufficient within-community interconnectedness, ER will identify all community nodes reliably before making false positives. In contrast to this, we attribute ER's inability to be a standalone CD method in general to the relatively high ER between the hub of a community and its spokes relative to the resistance between the hub of one community and the hub of another. In this general case, only a few paths exist for a low degree node to the core of its community (corresponding to a high resistance path), though there may be more paths from one high degree node to another, even if they are found in distinct communities. In the HSBM setting, all the nodes' degrees are distributed uniformly in a small interval, avoiding this limitation. The ability of ER to perform and scale well in this setting (competitively with PPR), and not even be greatly improved on by a germination stage, is evident in Figure 6.2.

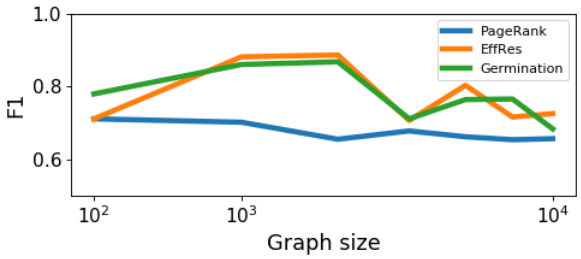


Figure 6.2: F1 scores for networks based on the HSBM model as a function of graph size

6.4.3 Synthetic Network Data

The next network type used is the LFR model of benchmark graphs.[90] In this model, the heterogeneity of the distribution of node degrees and community sizes are controlled

by a power law distribution, along with a parameter governing how high a proportion of a node's edges are inward-facing versus outward-facing. In this more realistic setting, with parameters to tune governing what proportion of a network is "popular" in its community, the contribution of ER to PPR can be assessed.

We show in this more realistic setting that pairing an ER-based germination stage with PPR outperforms PPR alone. In the top portion of Figure 6.3 we show that this result is not dependent on the size of the graph and in the bottom portion we show that this result is not dependent on the parameter of the power law distribution of the model. Though it is apparent that graphs with fewer degrees overall pose more challenging CD tasks, the relative performance of our method is consistently superior.

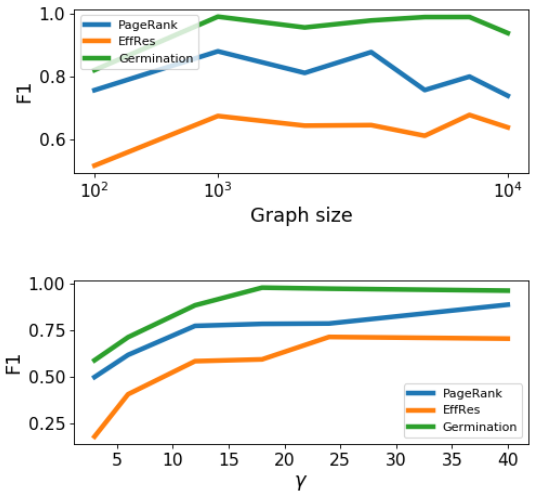


Figure 6.3: F1 scores for networks based on the LFR model as a function of graph size (top) and power law parameter (bottom)

6.4.4 Empirical Network Data

Lastly, we implement our CD approach on publicly available, large-scale, naturally occurring networks for which seed set expansion is particularly relevant. To assess the general

performance of these methods, we consider the mean F1 score over 10 experiments of sampling seeds from a fixed ground truth community. A summary of the (approximate) network sizes, and average F1 scores are presented in Table 6.1. "Germination" in the table denotes using a germination stage pre-PPR. When the difference in F1 score across all experiments is considered between PPR and germination, PPR was improved upon 85% of the time.

Table 6.1: Mean of F1 scores on empirical networks

NAME	n	**PPR** (sd)	**Germination** (sd)
Blogs	10^3	0.95 (0.015)	**0.96 (0.003)**
Amazon	10^5	0.69 (0.07)	**0.74 (0.02)**
DBLP	10^5	0.58 (0.12)	**0.67 (0.13)**
Youtube	10^6	0.42 (0.04)	**0.49 (0.05)**
Orkut	10^6	0.49 (0.21)	**0.60 (0.23)**
Wikipedia	10^6	0.57 (0.25)	**0.68 (0.21)**

For purposes of illustration, we take one instance of the CD task in the Amazon dataset and show the ROC curves for each CD method based on ground truth co-purchasing data. Each ROC curve depicts the precision-recall tradeoff for a progressively larger community estimate. A scoring function would determine which community estimate is selected. In Figure 6.4 we see the result of conducting PPR after a germination stage. Specifically, we observe that the high precision of using ER alone is short lived. After attaining $\approx 25\%$ recall, the ER attains a local minimum, at which point the germination stage stops. This triggers PPR to be initiated on the "germinated" seed set, leading to consistently better precision and recall, without regard for the point at which the stopping function is activated.

6.5 Future Directions

We have shown that a simple modification to PPR can lead to a significant boost in accuracy for CD. While our algorithm's specific contribution to single community seed set expansion is promising, we are even more motivated by degree to which a two-step procedure is beneficial

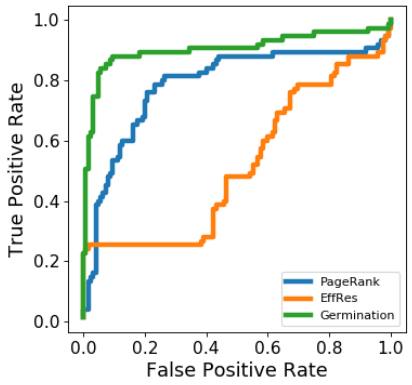


Figure 6.4: ROC curve for a single run of CD

in a single community seed set expansion problem.

In the future, we plan to demonstrate further the power of this two-step approach. Possible directions include considering a larger class of community scoring functions, algorithms for locally minimizing ER diameter, and thoroughly investigating the accuracy-time tradeoffs of these decisions.

CHAPTER 7

COMMUNITY DETECTION USING GLOBAL & LOCAL INFORMATION

Chapter 7 describes the application of multiresolution methods to the problem of community detection (CD). The detection of communities in a graph setting without any *a priori* labeling calls for flexible methods equipped to learn general graph structures.

7.1 A Brief Survey of Approaches to Community Detection

CD aims to find graph partitions that identify subgraphs for which there is a relatively high degree of within-subgraph interconnectedness and low degree of between-subgraph interconnectedness. For a partition of a graph into k subgraphs, $P^{V,k} = \{V_i \subset V \mid 1 \leq i \leq k\}$, a scoring for interconnectedness S, and real-valued graph functions $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$, any method for CD is an approach to optimizing (or optimizing a relaxation of):

$$\min_{P^V \in \mathbb{P}} \sum_i f(S(\{v \in P_i^V\}, \{v \in P_i^V\})) + \sum_{i \neq j} g(S(\{v \in P_i^V\}, \{v \in P_j^V\}))$$

7.1.1 Detecting Communities at Multiple Scales

Traditional CD methods uncover graph clustering consisting of a partition of nodes into disjoint sets. This is sufficient for simple graph structures where there is small variance in the distribution of community sizes and node degrees. In more general scenarios, such as when communities are of different sizes, or when clusters have core-periphery structure, we require methods that consider graph structure at multiple scales. All the more so in real-world scenarios, realized networks are rarely approximated well by coarse global methods such as those that we have already described in chapter 4. It is not necessary to assume that a partition $P = \{P_i\}_{i \in I}$ exists such that $\forall i, j \ P_i \cap P_j = \emptyset$ and in addition $\forall i, j, \ |P_i| \approx |P_j|$. Instead, we consider the cases where $\forall i \in I, \ \exists j \in I - \{i\}$ such that $P_i \cap P_j \neq \emptyset$ or even

possibly that $P_i \cap P_j = P_j$. In the last case specifically, it is obvious that partitions of communities at a single level and at a single scale will be insufficient.

The CD methods to address these more complex situations address the cases when networks are *hierarchical*, *multi-level*, or *multi-scale*. We define these terms for ease of discussion:

Hierarchical The degree to which a graph has *hierarchical* structure is a function of the distortion of the graph's best hyperbolic embedding. We choose to assume that hyperbolic embeddings (especially in low dimensional space) are emblematic of hierarchical structure, and that the difficulty of finding a low-error hyperbolic embedding is a signal for a lack of hierarchical-ness in the graph. Hyperbolic space is a convenient example of the more general class of ultrametric spaces, where nodes' mutual distances are computed with respect to how similar their shortest paths are to the graph's root node. The reason we consider hyperbolic space instead of another type of metric space model is the increased usage in recent years of hyperbolic embeddings for modeling tree-like structures.[87, 110] The rationale for consider hyperbolic space as a model for hierarchical graphs instead of the more straightforward hierarchical tree model, is that it may serve as a model for hierarchicalness even in settings when a hierarchy is generally observed, but is not strict.[1]

Multi-level The *multi-level-ness* of a graph $G = (V, E)$ is defined in terms of a corresponding directed acyclic graph (DAG) tree graph $T = (W, E_T)$, rooted at $w_0 \in W$. As a DAG, $\forall w \in W$ there exists exactly one shortest length path from w to w_0. The structure of G is defined in terms of T using a map S from W to V, mapping elements of W to subsets of V. $S(w_2) \subset S(w_1)$ if and only if there is a path in T beginning

1. An example comes from the animal kingdom. Even a person who is not well-versed in Linean taxonomy would be able to do a relatively good job classifying the animal kingdom into local and global categories. The set of dogs is inside the set of canines which would probably be together in a cluster distinct from primates and birds. This hierarchical structure generally rings true, but there are also many examples that test the crispness of these boundaries. Fish, for instance, generally refers to animals living under water. But whales are mammals, and lungfish are likely the genetic antecedent to earth-dwelling animals. There are undoubtedly some pesky elements in the "hierarchy" whose distance to other elements is not well-modeled by an ultrametric, but the structure remains somewhat hierarchical, or in the language of chapter 3, tree-like.

at w_1 and ending at w_2. $S(w_0) = V$, and so $\forall w \in W$, $S(w)$ is a subset of V, where successively smaller subsets are indexed by the images of vertices along that path from w_0 to w, ending in $S(w)$. If $w \in W$ is a leaf of T, then $\forall v_1, v_2 \in S(w)$, v_1 and v_2 are in the same clusters at every single level. If there exists an edge from w_1 to both w_2 and w_3 (and no other vertices), then $S(w_2) \cap S(w_3) = \emptyset$ and $S(w_2) \cup S(w_3) = S(w_1)$. G has an implicit hierarchy of clusters based on T, but the graph G itself may not bear similarity to a tree structure.[2]

Multi-scale A graph having *multi-scale* (or *multiresolution*) structure is a more general form of *multi-level* structure where, instead of the T defined above, there may be multiple paths from the root node to another node in the structure-summarizing tree graph. This generalized structure has been proposed before as a "generalized tree-based transform."[115]

We are most interested in *multi-scale* structure because it is the most general case and may be extended to the largest number of graph types. We also focus on instances when communities in a graph are of different sizes. Multiresolution analysis on graphs is adaptive to the discontinuous features of a graph, and so there should be no difference between identifying equally sizes clusters, from two clusters of different size.

Detecting a multi-scale structure depends on defining a hierarchy. One challenge for scoring function-based methods is a *resolution limit*. Especially when the scoring function is modularity, this limit has been well documented.[57] One variety of resolution limit is the difficulty of detecting two distinct communities when the ratio of within community to between community interconnectedness is not large enough. Another variety of resolution limit, and the bigger bottleneck, is the difficulty of separating a small community from a larger one when the difference in their sizes is large enough. Some methods have described

2. This is the case, for instance, in a standard hierarchical stochastic block model where the probabilistic model may be summarized using a tree, but the graph may not (without a lot of information loss). In the simplest case, this would be two distinct complete graphs which have a small number of edges connecting them.

ways of circumventing these limits by enforcing a hierarchical clustering on the network. For instance, a method that finds communities using modularity maximization may find hierarchical clusterings by recursively partitioning the graph into clusters.[119, 113, 75] Alternative approaches develop modularity-based functions that assign more weight to smaller communities, addressing both forms of resolution limits,[28] or introduce a parameter to tune for community size.[91] From a different direction, agglomerative hierarchical clustering may be represented using a dendrogram recording the merging of clusters until the entire graph is in a single cluster. This is the approach taken by the Louvain method described in section 4.2.2.[20] Empirically, the Louvain method has proven more adaptive to a variety of graph types, and serves as a motivation for our work that defines wavelets by learning a graph's structure agglomeratively.

7.2 Wavelets as Tools to Multi-scale Community Detection

7.2.1 Background

As described earlier in Chapter 2, wavelets emerged from the field of signal processing to provide a set of orthogonal basis functions localized in the time and frequency domains. In the case of signals, the nonzero support of the basis functions corresponds to a region of the signal where there may be a notable nonconstant behavior. A suitable set of wavelet basis functions for decomposing a large set of signals will contain the functions which are rough in regions where the set of signals are likely to be rough and smooth in regions where the set of signals are likely to be smooth.

In the case of graphs, in the presence of community structure, we expect functions on the graph (i.e. class labeling, responses) to be smooth with respect to the graph's community structure. For instance, if a graph's community structure were known and an analyst were attempting to decipher a graph function in the presence of noise or missing data, a set of wavelets which are smooth within communities and rough between communities would

be capable of de-noising the signal. Some previous research has used the Laplacian as a regularizer for functions on a graph and thereby as an estimator of vertex labelings in the presence of missing or noisy data.[157] Challenges with the application of spectral methods especially at a large scale[107] have led to theory from spectral theory being channeled to similar wavelet-based methods.[93, 63]

Considering wavelets as tools to describe graphs with either a fine or coarse brush provides wavelets with a very general role in machine learning on graphs. Just as wavelets in $\mathbb{R}$ allow the decomposition of a signal into basis functions at multiple levels of resolution, wavelets on a graph allow the decomposition of a graph into components at analogously different levels of resolution. Undoubtedly, frequency-valued eigenvectors also encode graph structure at different levels of resolution, but multiresolution analysis on graphs extends the eigendecomposition by defining wavelets with sparse components, leading to more reliable summaries on small scale structures.[49] Since it is not known ahead of time whether a graph will exhibit primarily global or local structure, it is critical to use methods that will work flexibly in either scenario. The primary aim of this chapter is to showcase the ability of wavelets to find community structures that are undiscovered by spectral clustering. In fact, we aim to demonstrate a finer point: the ability of MMF wavelets to find community structures that are undiscovered by spectral graph wavelets, showing that even wavelets derived from graph eigenvectors retain some of the limitations of using spectral methods in a graph setting.[133, 88]

Spectral graph wavelets are scale-dependent, with scale parametrized by γ. The wavelets themselves are measurements of the coordinate-wise similarity, subject to the scaling of γ, between different nodes' entries in the n eigenvectors. So, while the wavelets produced by this method are sparse and supposed to be approximating local behavior, their derivation is dependent on how well the eigenvalue decomposition itself is estimating local graph behavior. How well does a wavelet method rooted in an eigenvalue decomposition performs CD and which alternative methods exist for the CD problem?

7.2.2 Contributions

We introduce community detection using a wider class of wavelets, and in line with previous work in community detection,[64] we verify that there is "no free lunch." We claim that different types of wavelets address the CD problem differently, with strengths in different graph types. Specifically, we find that community structure in synthetic graphs generated from SBMs or HSBMs are recovered most efficiently with spectral-based wavelet methods, while smaller communities and communities with hub-and-spokes-like structure, modeled using LFR, DC-SBM, or real-world data are recovered more reliably using wavelet methods that make use of agglomerative clustering. There are many strategies in agglomerative clustering for how to choose which nodes should be pooled and how to preserve information from initial mergers during higher order clusterings. The approach of our method - generating wavelets by means of a k-point MMF - selects vertices by pairing a node and other $k-1$ nodes with whom they share maximum inner product in the current state of the graph Laplacian. The Laplacian is then updated using k-point Givens rotations. Further explanation of the procedure for MMF may be found in chapter 5.

Cycle Spinning and Wavelet Smoothing

Agglomerative clustering alone is typically not sufficient to recover graph structure. A single subgraph consisting of a tree, or at least a tree-like, approximation may be subject to noise or be simply insufficient to approximate a graph at all relevant scales. Due to reasons of scalability, we wish to stay in the world of sparse approximations, and so we consider the use of *cycle spinning*[35] in the world of community detection. Cycle spinning refers to the introduction of random translations into existing wavelet bases' supports in order to obtain smoother representations of signals using sparse basis functions. Its initial introduction applied the idea to simple wavelet designs in real-valued Euclidean metric spaces. Implementing the random translations in said Euclidean spaces is straightforward, but translations in the graph setting are difficult to define. In the context of semi-supervised learning, cycle spin-

ning has been used as a rationale for averaging outputs of wavelet-based methods to recover more robust output.[63] In the context of CD, we include the result of several nondeterministic wavelet-producing algorithms as an input into the same hierarchical clustering tool that has been used in past work.[133] We consider the increased explanatory power of a collection of wavelet bases as akin to the use of spectral wavelets at multiple scales. We call the MMF wavelets MMFW (multiresolution matrix factorization wavelets) in contrast to SGW (spectral graph wavelets). In all side-by-side comparisons with spectral graph wavelet-based methods, the same number of total wavelets are being used by each method.

7.2.3 Perspectives on MMF Wavelets

The MMF of a graph Laplacian is a means of producing a wavelet basis for the space defined by the graph Laplacian. Importantly, a set of parameters specify exactly how the MMF algorithm works at each stage of the algorithm. To clarify what these parameters are, and to demonstrate the varieties of graphs that MMFW can represent, we investigate the way that these parameters affect CD performance. The parameters we consider are:

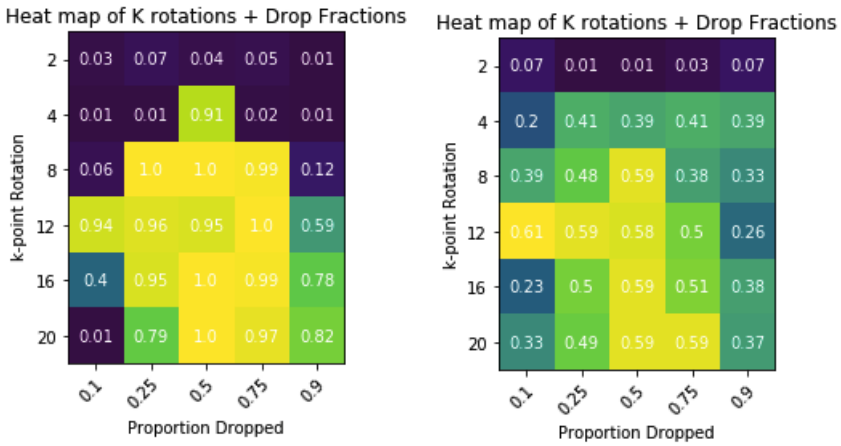
k The order of the rotations at each stage of the algorithm

f_{drop} The fraction of the rotated nodes at each stage to drop to the inactive set of indices

t_{drop} The type of method used for selecting which nodes to keep in the active set and which to drop to the inactive set

n_{reps} Number of sets of wavelets (multiple of vertex set cardinality) used as features for CD

We consider concurrently the effect of the order k of rotations and the proportion of those k nodes which are dropped to the inactive set at each stage of the algorithm. The two plots in figure 7.1 show the accuracy of MMF for CD given specific combinations of k and f_{drop}. This perspective on two parameters shows that in the case of uniform degree distributions, more nodes being dropped at each stage yields stronger performance, because



(a) Relationship between k and the proportion of k dropped at each iteration for a S-P graph

(b) Relationship between k and the proportion of k dropped at each iteration for an LFR graph

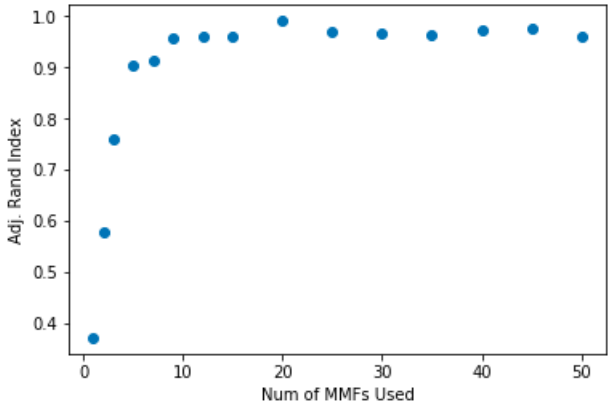
Figure 7.1: MMF accuracy as a function of different parameter values

the relationship between nodes in the same community may be summarized very coarsely. For LFR graphs on the other hand, dropping a lower proportion of rotated nodes yields better performance. In this more complicated graph structure, reducing information too quickly loses needed information.

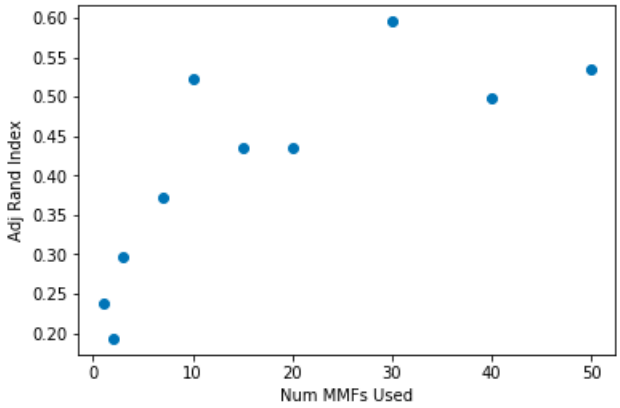
The n_{reps} parameter, governing how many times an MMF produces wavelets for a graph, is extremely influential in terms of performance improvement. Besides parallelization within the MMF algorithm, simply running more versions of the same thing to produce more wavelets is easily parallelizeable, leading to improved performance at effectively zero cost. In figure 7.2, we find that in each graph type, performance relative to a defined benchmark increases quickly as a function of n_{reps}.

7.3 Benchmark Comparison

The existing, primary benchmark towards which wavelet-based CD methods may be compared is the aforementioned SGW decomposition plus hierarchical clustering (SGW+HC).[133] We refer to our own method as MMFW+HC because its features are the wavelets of the MMF. We consider only datasets, synthetic or actual, where ground truth communities are



(a) Sales-Pardo graph



(b) LFR graph

Figure 7.2: Role of number of sets of MMFW on MMF accuracy

known *a priori* and all estimated clusterings are compared to said ground truth using the adjusted Rand index.

A second approach to assessing the quality of a community detection algorithm in a purely unsupervised setting (where no supposedly ground-truth labeling exists), is the performance of link prediction and link description tests using the CD algorithm's output. In a link prediction task, the training stage consists of running an unsupervised community detection algorithm on a subset of the graphs total edge set. The complement to this subset is the test set. Then the estimated communities are used to rank all remaining edges (other than those actually observed at train time) with the objective of ranking the test set's edges higher than nonexistant edges. The classification of removed edges and nonexistant edges at test time may be assessed using typical techniques such as the area under the ROC curve. In a link description task, coined in Ghasemian *et al.*,[64] we aim to measure the degree of overfitting by a method's CD scheme. The scores assigned by a method to all node pairs in the edgeset are considered relative to their membership in the observed and unobserved sets of edges. When a method does not suffer from overfitting, we would expect good classification accuracy between observed and unobserved edges.

7.4 Application of Wavelet-based Community Detection to Synthetic Data

7.4.1 Graph Models

As a first step, and a way of assessing which graph features influence the efficacy of a method, we consider three varieties of synthetic data. The simpler models are stochastic block models and hierarchical stochastic block models. In both of these cases, clusters are considered homogenous in node degree. The most relevant feature of the graph model is the extent that communities are relatively complete graphs and contain no nodes that are notably more or less popular.

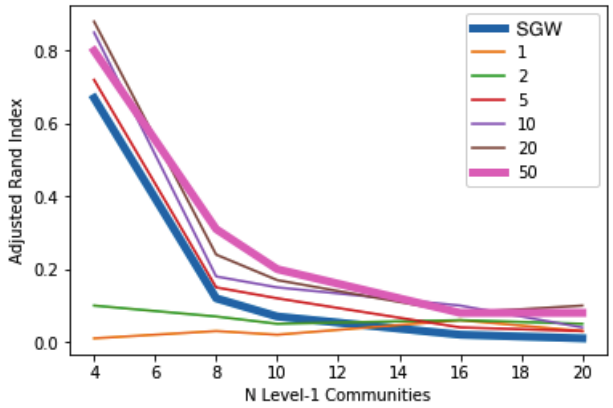
By considering a wide array of real-world networks in disparate fields, it has been shown that network models where nodes are distributed according to the exponential distribution describe reality more accurately. This indicates that in any community there are a very small proportion of popular nodes and most nodes in a network are on the periphery. Extensive empirical data has suggested that a vanilla SBM is insufficient to model real-world networks and led some researchers to develop more complex models such as degree-corrected SBMs. Another model is the LFR model which assumes a power law decay in node degree within each community. LFR also assumes community size is governed by a power law distribution. The contribution of our work is to accurately recover all levels of graph structure, and specifically to focus on the recovery of local graph structure. For this reason, these models will be tested in scenarios where small communities must be recovered in the presence of additional, higher-level organization.

7.4.2 Results

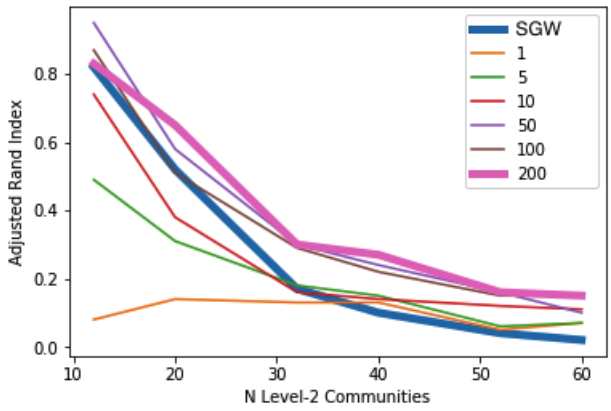
Homogenous Degree Network Models

The HSBM, or Sales-Pardo graph is a graph model where all nodes in a community have approximately the same number of connections. As oppose to a standard SBM, the Sales-Pardo model uses a hierarchical probabilistic structure to produce a corresponding multi-level structure in the realized graph.

In Figure 7.3 we find the success of a method as a function of the difficulty of the task. Each line reflects a different model, with "Tremblay" referring to SGW-HC and the other lines are identified by an integer k which represents an MMF-based model with kn features, n being the number of nodes. In position 7.3(a) we consider the identification of only the higher level structure and in position 7.3(b) is the detection of smaller, lower-level communities. In both cases, the effectiveness of all methods decays as a function of the number of communities. Nevertheless, the use of MMFW exhibits uniformly better



(a) Detection of high-level communities



(b) Detection of low-level communities

Figure 7.3: Detecting graph structure for different hierarchical levels. SGW is compared to MMFW for a selection of different numbers of MMF used.

performance than SGW-HC at every level of difficulty. It is apparent that in this case of a uniform degree distribution a large community may be detectable for a small parameter of connectedness, while in a smaller community more edges are required in order for it to be recovered. In this example, parameters were kept constant across every realized graph and this leads to a decay in performance as a function of the total number of communities. While performance is uniformly better when MMFW are used, the graphs considered here are less realistic than those with power law node degree distributions considered next.

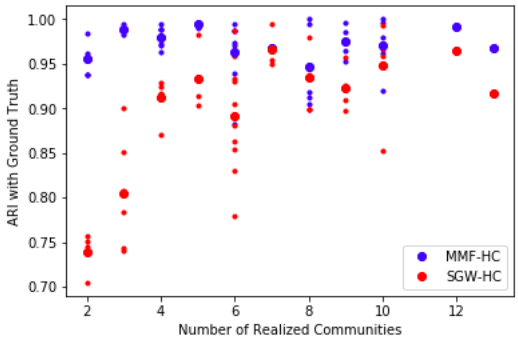


Figure 7.4: Comparison of SGW and MMFW as a function of the number of realized communities for graphs generated using the LFR model

Power Law Degree Network Models

The non-homogeneity of the networks' nodes' degrees makes CD for these models more difficult. A community's subset of popular nodes may be easy to group together, but nodes at the periphery have fewer connections to the community's core, providing more opportunities for algorithms to make mistakes. On the other hand, popular nodes are more likely to be edge-connected to other communities' nodes than lower degree nodes. For these two reasons, this model type poses additional challenges for CD. In figure 7.4.2 we compare the performance of wavelet-based CD using SGW and using MMFW. In this context, we expect larger communities, with correspondingly large cores of popular nodes and large peripheries to be more difficult to accurately label than smaller communities. At the same time, the unsupervised nature of CD makes identifying the number of communities difficult, causing challenges for very large numbers of communities. We observe in the plot that for a graph of fixed size, performance degrades for the smallest number of communities (< 5) and for the most(> 10), but in a middle interval ($5 - 10$) performance is highest.

In the experiments described so far, the correct number of communities was actually input to each algorithm from the start. We address some aspects of the fully unsupervised problem of detecting community structure in chapter 9.

Name	n	%	Example
Biological	192	33.6%	Network of protein-protein interactions in 11 species
Economic	122	21.3%	Medieval Russian river-based trade network
Informational	22	3.8%	Co-purchasing of books on politics from Amazon
Social	124	21.7%	E-mail network at a Spanish university
Technological	74	12.9%	Class dependencies in a software program
Transportation	38	6.7%	Availability of seats on flights between US cities

Figure 7.5: Count of graphs within each network domain

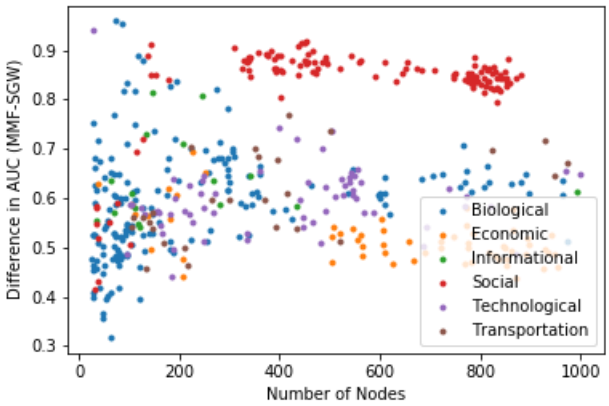
7.5 Application of Wavelet-based Community Detection to Generic Real World Datasets

Having established some patterns of behavior for wavelet-based methods in synthetic network datasets, we consider a class of commonly used network data for CD tasks.

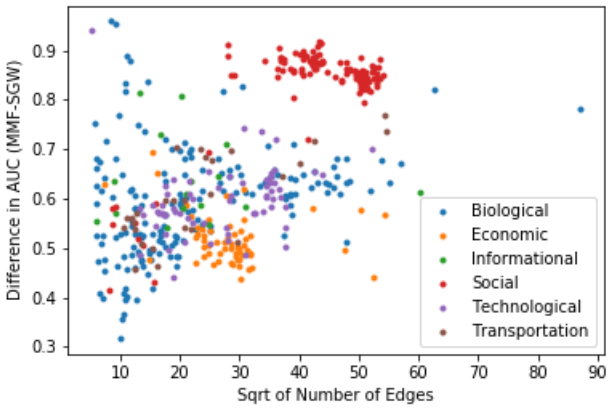
7.5.1 Data Description

The CommunityFitNit collection was first used to provide evidence of different CD procedures' tendency to over- and under-fit community structure. [64] They evaluated different methods' overfitting behavior as a function of graph size, graph type, and community characteristics. We similarly seek to exhaustively compare the behavior and empirical performance of wavelet-based CD methods across a very large dataset of graphs. The graph features we consider are the numbers of nodes and edges in a graph and the network domain to which the networks belong. The domains we consider are biological, economic, informational, social, technological, and transportation. Their respective counts are recorded in Table 7.5. The table also includes one representative example of the categories. It has previously been demonstrated that the network domains tend to lead to different community structures.[64] We do not consider other characteristics of these datasets in depth beyond the high-level features mentioned above which have been used in past studies.

The data contained in this corpus are purely unlabeled tasks and we remain agnostic to essentially any domain-specific knowledge to bias the unsupervised nature of the learning



(a) Comparison of MMF and SGW as a function of number of nodes



(b) Comparison of MMF and SGW as a function of number of edges

Figure 7.6: Difference in MMFW and SGW performance for a wide gamut of network types

algorithm. To establish our accuracy relative to an unlabeled dataset we measure success using a link prediction task.

7.5.2 Results

The method considered to assess our MMFW-based method's is a state of the art wavelet-based CD method (SGW),[133] We present the difference between different methods as a function of the number of nodes and the number of edges in a graph.

The next two section, 7.6, is an exploratory analyses showing the effectiveness of MMF

in an applied setting. The aim is *not* to compare MMF-based CD to other CD methods, but rather to discover its performance relative to other existing, sensible non-ML community labelings. The application is not "traditional," and has not appeared in other CD literature. Our aim is only to demonstrate the success that an MMFW-based method can promise in new frontiers of CD.

7.6 Application of Wavelet-based Community Detection for Financial Data

We next consider the use of wavelets for community detection among stocks in the S&P 500. Clustering of time series poses a unique challenge because several different distinct steps must be defined.[138, 52] First, a metric must be defined for comparing the similarity between different time series, which may cover different time intervals. Several approaches exist for comparing long-term time series. For example, dimensionality reduction,[68] Fourier features, or wavelet-based semblance analysis.[37] Next, a clustering procedure is used to identify relevant groups of time series.[138] By structuring this clustering problem as a graph, and detecting communities therein, a richer category of clustering procedures may be used. Past work has surveyed existing distance measures and clustering techniques for this variety of CD.[54] We compare our estimated multi-scale communities with two existing labelings. We do not seek to use this dataset as a benchmark for comparing different CD methods, but rather as a means of discovering what the MMFW-CD method is learning about the most relevant companies in the US stock market.

7.6.1 Problem Setting

The stock market is typically considered one of the most salient measures of the world economy, representing changing levels of valuation and optimism in national and international markets. The unpredictability of individual stock prices led Eugene Fama to propose the

random walk model for stock prices: a stock price's history is such a weak predictor of future performance, that one is better off assuming its price changes may be modeled by a random walk.[53] That being said, for two stocks which have very similar histories, the performance of one stock may serve as a reliable predictor of the other's. This leads many economists to produce massive correlation tables for companies in the stock market, using information regarding one company to influence their valuation of other highly correlated companies.[56] One such table using stock price time series' wavelet decompositions as the metrics of comparing distinct series, is presented in Figure 7.7. The figure organizes the S&P 500 companies first using the Thomson Reuters Business Classification (TRBC) identification of sector, and then organizes each sector based on the TRBC classes of industries of which it is made up. The sectors considered are: Basic Industries, Capital Goods, Consumer Durables, Consumer Non-Durables, Consumer Services, Energy, Finance, Health Care, Miscellaneous, Public Utilities, Technology, Transportation, and unassigned. The subcategories of industry, for the energy sector for example, are consumer electronics, industrial machinery, integrated oil companies, metal fabrication, oil and gas production, and oilfield services. While in general there are many companies involved in each of these sectors and industries, their representation in the S&P 500 may be small, leading to only one or two companies per industry.

We consider a novel use of community detection in financial data, whereby community membership is used instead of stock correlations alone. We make the initial assumption that stocks belonging to the same communities will exhibit similar future price movements. We further assume that sub-communities exist, whereby the smaller the community, the more similar community members' price movements will be. This basic assumption, if verified, may be leveraged to very practical (and lucrative) ends by identifying stocks that are likely overvalued, undervalued, or simply in need of further investigation to figure out what is going on.

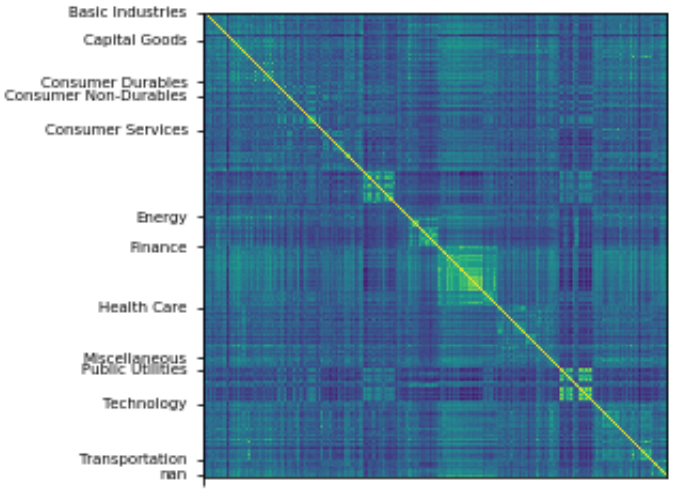


Figure 7.7: Similarity between companies using the similarity of the wavelet decomposition of their time series, ordered according to TRBC labeling

7.6.2 Description of Data

Our dataset consists of six years of stock prices for the companies belonging to the S&P 500, the 500 companies in the American stock market with the highest market capitalizations. Since the companies belonging to this group change over time, we selected the 486 current S&P 500 members who have belonged to the group continuously since January 1, 2013. We choose to begin our analysis in 2013 to ignore, to whatever extent possible, the effects of the recession in the preceding years, which would likely lead to an overestimation of stocks' correlations from the recovery period. The company information used consists of opening and closing stock prices for each relevant day in the aforementioned time period.

7.6.3 Method

For the first stage of time series comparison, we use Haar wavelets on the real line to reduce the dimensionality of each signal in the collection and smooth some of their irregularities to make comparisons easier. Using wavelet semblance analysis, we are able to define pairwise

similarity between companies' time series data. Semblance analysis measures the degree to which the phase of two time series align. By basing the semblance analysis on wavelets, as oppose to Fourier features, the phase of pairs of time series may be compared even when frequency is changing over time.[37] The matrix of pairwise similarities between companies that are computed using semblance analysis define the weights of the edges in the network of stocks within which we will detect communities. We assess our ability to detect communities in two ways. First, we compare our estimates of community membership at different scales relative to existing known labelings, such as company sector (larger clusters) or company industry (smaller clusters). We see how our estimates differ from these known labelings in general and over time. Next, we consider the link prediction task as a measurement of how well our community labeling models the network, apart from what known labelings indicate. Last, we consider the prediction task where the training set is time series data until time n and the test set are subsequent days.

7.6.4 Results

Objective 1: Comparison of MMFW CD to Benchmark

Our first aim is to assess how closely our community estimates match known labelings - in this context, the TRBC sector and industry labelings. Relative to these two labelings, the MMFW-based CD method attains an adjusted Rand index of 0.32 and 0.30, respectively. From this perspective, our estimated labeling is not very effective at recovering these pre-defined company classifications. It is important to note, though, that while these "ground truth" labelings are accepted as providing a reasonable clustering of S&P 500 companies, they are not primarily based on daily stock prices, but more detailed, often qualitative information including company filings and "market perception."[132] So while an approach with purely stock price-based CD is not able to substitute for an approach that takes into account a broader class of information inputs, it is nonetheless providing a different, possibly helpful

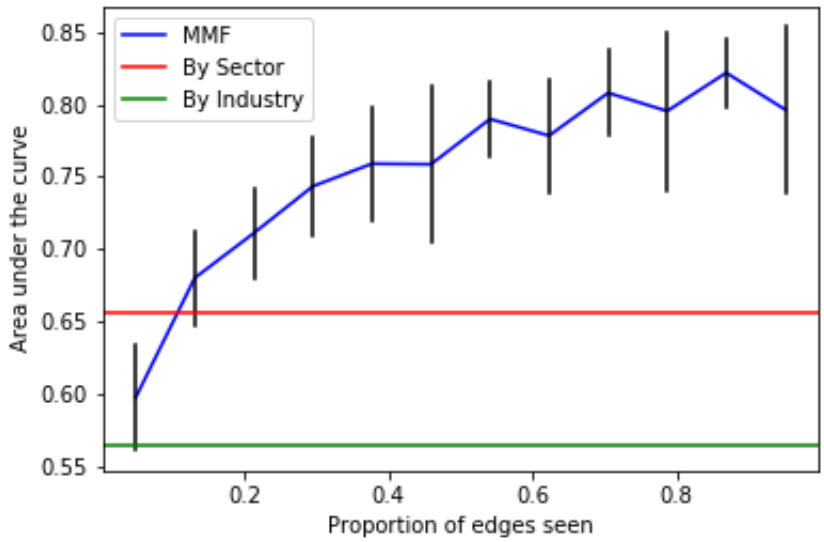


Figure 7.8: Link prediction by MMFW-based CD versus "ground truth" TRBC labeling

perspective.

Objective 2: Which clusterings capture graph structure best?

To demonstrate how CD can contribute to classification of these institutions, we consider our second objective: to measure the extent to which a number of suggested classifiers match the empirical community structure as represented by similarity between time series' wavelet decompositions. We use a link prediction task to measure how well these labelings capture graph structure. We compare the TRBC labelings of sector and industry to the MMFW-based CD. In Figure 7.8 we observe that the communities defined by existing, widely accepted benchmarks does not predict missing links better than MMF unless less than 20% of the total edges are observed.

The types of communities MMF identifies also have different properties than those provided by TRBC. While the sizes of the communities suggested by the sectors and industries have median (and IQR) of 33 (14, 49) and 3 (2, 5), the median (and IQR) of the sizes of communities estimated by MMFW is 13.5 (9, 28.5). Based on the IQR's estimates of how spread

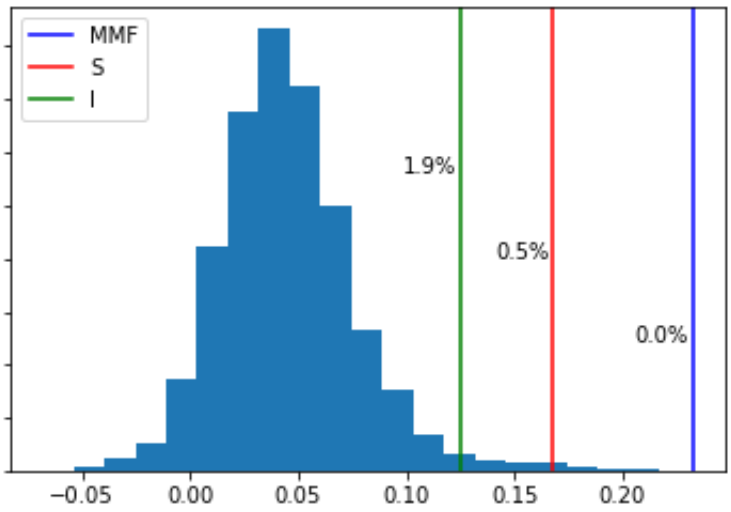


Figure 7.9: Average within-community similarity relative to a bootstrapped sample of within-community similarities

out the community sizes are, the skewness of MMF and industry-based community assignments is larger than that of sector-based assignments. Furthermore, the median community size from MMFW's communities is in between that of the industry-based and sector-based assignments, respectively. This gives us the impression that MMF's performance in the link prediction task results from MMF having a relatively large spread of community sizes, with a higher number of small communities than large ones. By preserving the skewness of community structure sizes in this dataset (unlike sector-based assignments) and still identifying communities of larger size (unlike industry-based assignments) MMF is able to use multiple levels of resolution to describe community structure in the S&P 500.

Objective 3: Prediction of future price movements

The next experiment is based on the assumption that community structure is able to serve as a feature for the purpose of prediction. In the case of time series, this is expressed in the assumption that companies in the same community will have similar stock price movements.

We divide the collection of companies' time series into three parts, the data used begins

with daily observations on February 8, 2013 and ends on February 7, 2018. We divide this dataset into three parts. The unsupervised learning stage (February 8, 2013 to February 4, 2015), the anomaly detection stage (February 5, 2015 to January 30, 2017), and the test stage (January 31, 2017 to February 7, 2018). The data from the unsupervised stage is used for community detection. With community structure estimated, the anomaly detection data is used to identify the ten percent most "unusual" companies within each community. To define company unusual-ness we sum the magnitude of the semblance between the company and all other members of its community. The companies whose sums are beneath the tenth percentile are deemed the anomalies of the community, exhibiting less inward similarity during the anomaly detection stage than other companies in the community. In the test stage, we consider the percent change in stock price on day January 31, 2017 + T relative to day January 30, 2017. This measures the percentage change of the company's stock for each day in the test stage relative to the last day of the anomaly detection stage.

The key CD-based assumption we make regarding the "unusual" companies is the following: if during the anomaly detection stage the unusual company's net stock movement from January 30, 2017 relative to February 5, 2015 was larger than the average of the community, we anticipate that in the test stage, the price will go down. If its net movement was smaller than the community average, we anticipate the price will go up. This assumption may be formalized that each member of the community is modeled as a random variable whose expected percent change on any day is estimated using the average percent change of all community members. Furthermore, the stocks within a community are not independent - they are in the same group precisely because of the high correlation of their movements - therefore, we consider one of the "unusual" companies undervalued if it has had negative price movement relative to the rest of the community, and overvalued if the opposite is true.

We test the effectiveness of CD relative to existing labeling benchmarks, such as sector or industry, in the task of predicting that a given day in the test period will have a positive return relative to the last day of the anomaly detection period. Over time, we anticipate a

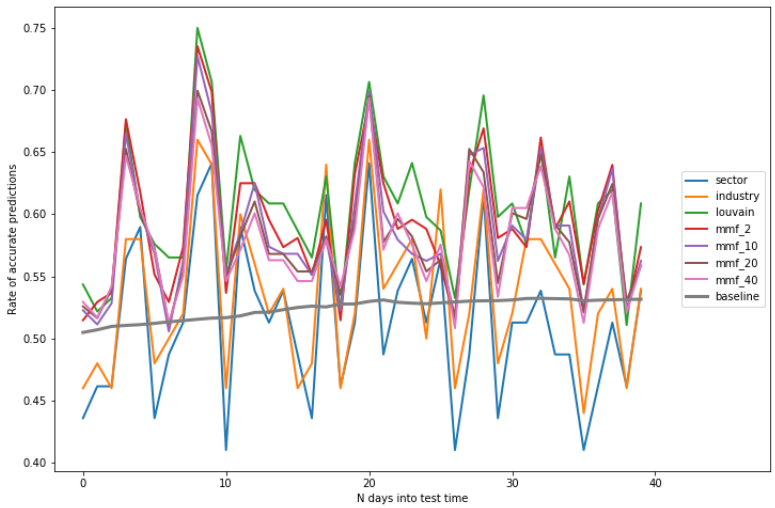
decrease in the ability of a company's community to predict said company's movement. At the beginning of the test stage, however, the difference between the "unusual" company under consideration and the community average is great enough, that we anticipate undervalued stocks having a positive price movement (relative to the community average) and overvalued stocks will have a relatively negative price movement.

To report our results, we consider the rate of correct prediction of price movement for all unusual stocks across all communities for a given labeling method. The labeling methods we consider are sector-based, industry-based, Louvain CD-based, and MMF-based. We consider several iterations of MMF using different sized communities. A predicted price movement is deemed correctly classified as positive or negative, if the percent change of the company's stock is indeed more positive than the communities average or if it is indeed more negative than the communities average.

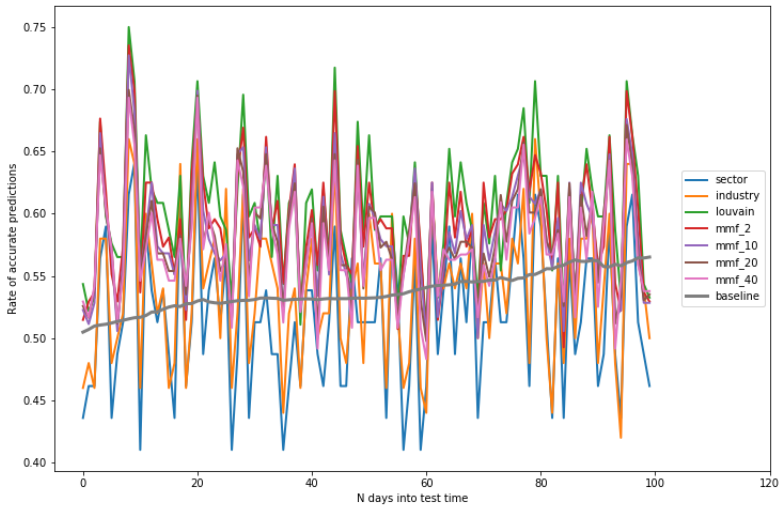
Figure 7.10 is the first perspective on the results for each labeling method, for both a short (panel 7.10(a)) and long (panel 7.10(b)) time frame. Over the test period, the baseline is gradually increasing because the average percent change for S&P 500 stocks in that time period increased. This means that if a community's average change remained constant, but an unusual stock within that community was predicted to simply track the S&P 500, the baseline is the rate of correct predictions that would be made.

We find that over a time frame of ≈ 40 days, the predictions using a CD method are dramatically better than the predictions made using conventional categorizations of sector and industry, and furthermore beat the baseline prediction by several percentage points. The difference between the Louvain method and any of the implementations of MMF is very slight but the Louvain method appears to exhibit better performance. We attribute this to the hierarchical clustering algorithm being used by MMF. Perhaps if we were to incorporate the clustering method used by the Louvain method, we would see additional increases in performance.

We also note that the MMF labeling with the fewest communities - only two - tends



(a)



(b)

Figure 7.10: Predicting the movement of stock prices in a test set using a pre-learned community labeling

to predict price movement better than the labelings with more communities. We suggest that this may be the result of the relatively small sample size considered for analysis. If this experiment were conducted on the entire US stock market instead of just the S&P 500 stocks, we anticipate that there would indeed be an advantage for the MMF method to use more communities.

When we consider prediction of price movement relative to community average over a longer time period, the effects we noticed over a ≈ 40 day period are less apparent. This is not surprising, because the discrepancy between the community average and the "unusual" community member will have lessened over time.

7.6.5 Summary

Having used the MMFW-based CD method in a financial application, we see the benefit of applying CD methods to very general problems. It emerges that the clustering imposed by conventional benchmarks, such as sector or industry, are quite different from the community labeling produced using our approach. Through link prediction tasks, we are able to compare the ability of MMFW to capture network structure relative to these benchmarks, and characterize some of the specific differences between MMFW-based communities and those that are conventionally used. The most interesting feature we discovered is the power law-distribution of community sizes that emerge when MMFW-based CD is used instead of existing benchmarks. Furthermore, in an original and quite challenging prediction task, of detecting anomalous companies within communities and using community structure to make predictions, MMFW-based and Louvain CD strikingly outperform the baseline prediction and existing conventional company labelings. This result makes a strong case for the use of unsupervised learning to discover complex community structure that exists among companies in the US stock market. A next step for this investigation would be to consider all companies traded on the US stock market instead of a not-so-representative sample of the five hundred largest ones.

CHAPTER 8

CONCLUSION

The complicated field of large-scale machine learning requires a veritable war chest of tools to enable scalability, accuracy, and precision. We have humbly presented a multi-faceted approach to graph-based analysis. Using recent developments in hyperbolic representations of data, multiresolution analysis, and community detection, we demonstrate the effectiveness of the MMF in the network analysis habitat. Four novel contributions have been central to this thesis.

First, by using the vocabulary and tools of hyperbolic metric embeddings, we provide a theoretical foundation for our approach to multiresolution analysis on discrete spaces - MMF - as well as other similar methods. Thinking of matrix factorizations in terms of underlying hyperbolic embeddings also motivated our suggestion that the quality of the factorization may be improved by making use of "big data" tree-based kernels. Employing these similarity metrics enforces that the wavelets extracted using MMF are adapted to be smooth in a metric space that is tree-like.

Second, attaching an interpretation of the MMF as a hyperbolic metric embedding, provides a much needed explanatory backing to the application of the MMF algorithm. Wavelet transforms in general are endowed with regularizing assumptions. Each variety of wavelet transform makes its distinct structural assumptions about data, whether it be Haar, Daubechies, or Mexican hat. In the case of graphs, however, where each graph defines its own distinct metric space, the structural assumptions needed to define a multiresolution basis must be more generalizable. We have shown that by defining orthogonal wavelet bases which are smooth with respect to a hyperbolic embedding space, wavelets are constructed that reliably represent the types of data found in real-world datasets. Secondly, wavelets that are smooth in an underlying hyperbolic metric space give the ML practitioner the ability to regularize graph functions with respect to this space. Whereas past work aimed to learn graph labelings by constraining functions to be smooth with respect to the observed

graph topology, by instead constraining functions to be smooth with respect to a hyperbolic embedding, we add a regularization term that serves as a robust de-noiser.

Third, we have made a foray into the world of seed set expansion to show that modeling diffusion from a seed set using a collection of spanning trees leads to a "germinated" seed set that serves as a better starting point for personalized PageRank. By detecting communities in a two-step process, we have added another parameter into the fields of seed set expansion that is able to accommodate a more diverse set of graph types.

Lastly, we have presented a wide array of applications - both synthetic and actual - to show the manner in which MMF may be used for CD, and to demonstrate its competitive edge relative to existing methods based on wavelet transforms. In a novel application of CD for stock market predictions, we have shown how an MMF-based CD scheme is a very general, flexible tool that may be used in unsupervised and semisupervised learning paradigms.

Two areas remain that would have fitted into the narrative of this thesis but were not sufficiently explored to be included at this time. First, a more rigid characterization of how collections of trees are hyperbolic is needed. We have shown how tree product metric spaces exhibit many characteristics that are similar to hyperbolic metric spaces, but this is not always the case and we would like to specify when it is not the case. The δ-hyperbolicity of a tree product will depend entirely on the space the trees are spanning. We attempted to describe this dependence but do not currently have the mathematical formalism to do it precisely.

Second, a "low lying fruit" next step of applying MMF wavelets is the computation of graph statistics. Graph statistics belong to an exciting and quickly growing field of hypothesis testing on graphs where wavelet representations of graphs have much to contribute. One relevant publication uses the wavelets derived from random spanning trees to generate Haar wavelets for use as the backbone of hypothesis testing; the extent to which graph activation functions align or do not align with these wavelet functions (as measured for instance through inner products) indicates if a function is smooth with respect to graph structure or not.[123]

Similarly, the ability to use a hyperbolic representation to well-approximate a graph, such as by using MMF, could prove a promising starting point for testing if a graph has clustering structure or not. Testing for graph structure might be seen as a precursor to large-scale CD because it could assist an algorithm in identify where best to seek out community structure.

It is understandable that some questions may linger regarding the aims and accomplishments of this thesis. It is my hope that the epilogue will help clarify things.

CHAPTER 9

EPILOGUE: A FICTIONAL ILLUSTRATION OF APPROXIMATE ORTHOGONAL BASIS REPRESENTATIONS

A Language for Change: Part 1

There was once a man who worked at a company called "Salient Signal Analysis," and it was the day of his retirement. His name was Samuel Samuel Auerbach. He always thought it was fitting that his initials matched those of the company where he worked; it reassured him that he had chosen the right career. His name had also led to unanswerable questions. Having discovered his birth certificate (and his thentofore unknown middle name) only late in life, years after his parents had passed away, he was unable to ask his parents why his middle name was the same as his first name. Was it a mistake in registration at the hospital? Was it intentional? Had they told him the reason and he had forgotten? It is tragic that questions of such obvious magnitude might be left indelibly unanswered.

Retirement was an unusual moment for Samuel. The act of stopping to do goal-oriented tasks usually implies that one has decided that a goal has been reached, a goal is unachievable, or a goal is simply no longer interesting. In the case of retirement, the decision is less explicit. It is simply through the passage of time that the nature of a person's goals, or others' attitudes towards that person's goals begin to change. Does a person truly decide to retire - or does retirement happen to them? Samuel thought, I think the function of the brain's frontal lobe has something to do with this.

It was Samuel's nature to reflect. He had no shortage of questions, but the communication of those questions with color and precision remained a great challenge for him. It was curious then that his professional career had been in the service of a company whose purpose was the transmission and interpretation of signals. He had worked in a division of the company aimed at theoretical aspects of signal transmission. It was a pain for Samuel that his own research to develop more efficient and effective methods for information transmission was ultimately

so challenging to communicate to other human beings. It was really not so complicated, if his audience would only take the trusting step that any given sentence would be clarified by the one that succeeded it. He was in the habit of rehearsing his professional introduction internally, ready to perform at a moment's notice for a willing conversant. He planned to say something like:

– My area of expertise is compression. Bubble wrap can be carried from one continent to another with only its form, reduced of its "content" - compressed. A pastoral scene can likewise be reduced from a spectrum of unbounded color and texture to an indistinguishably similar discretized version. Is it not the case that one thousand or five hundred shades of red will suffice in the place of a palate with billions - or really infinite - reds? This is the essence of compression - the act of removing essentially invisible material from information, losing an unnoticeable amount of information, while lightening the cost of transmitting said information.

He thought, on the day of my retirement is it still necessary to rehearse this madness? But the momentum of habit pushed him forward...

– This, you see, is a compression in the visual domain. What can be ignored in a signal without setting off any alarms? An alternative approach - indeed the one that I am working on - challenges the utility of the "visual domain" approach because, though barely noticeable, beauty is undeniably lost in moving from an infinite to a finite number of reds. Not only is beauty temporarily lost, but that beauty can never be recovered. There is no reverse algorithm to redraw the full splendor that once existed in the image. What if instead, a vocabulary was developed - not a literal spoken vocabulary, mind you, but a vocabulary of abstract thought and visual concepts - whereby through the overlay of a fixed number of prelabeled indices, a complete scene could be reconstructed. Through a communal consensus on the meaning of these concepts, and the means with which they might be taught to the next generation, a message from one human being to another could be spoken, compressed, transmitted, uncompressed and understood. The richness of the message would be reduced

by the finiteness of human communication, but the listener's reconstruction would appear as deep and rich as an uncompressed thought.

Here he would imagine his interlocutors becoming impatient and he would begin speaking faster.

–My supervisors, you see, say I am merely proposing to reduce humanity's vocabulary. They say that I want people to talk about simpler things so that my job is easier. But the truth is - in a sense - they are not far from the truth. The simplicity or complexity of an idea is not an absolute measure, but a quantification of how many possible conceptions there are for its meaning. Though reading is very hard for a child who imagines all the ways that an "a" can sound, a literate adult condescends to his illiterate neighbor, asking why he cannot do something so *simple*. Through deliberation and discussion, with dedication and discovery, the level of communal discourse could be heightened so that the *simplification* of communication could be seen as an *elevation* of communication.

Samuel hoped that his listener might at this point be sufficiently jolted, excited, or convinced that their subsequent queries and clarifications would make his ideas clear. But, in general, it was a cruel irony that his ambitious theories of simplifying the transmission of information would remain out of reach from basic conversation. And verily, by fellow human beings' disinterest to engage with the - for him - simple ideas he proposed, they remained complicated ideas and he remained alone. Yes, indeed - irony.

There was no concrete parting gift. Rather his colleagues announced a "happy hour" at the quiet pub down the street from their offices. Samuel did not frequent establishments of that sort, but for the occasion at hand and out of respect for the planners, he made what he thought would be a short appearance. After several hours had passed, though, he found himself talking to a pleasant lady, someone he did not work with, in a booth under the sheltering glow of a light pendant from above. Her name was either ... he couldn't remember that at the moment. They were, to his surprise, enjoying themselves - he didn't have to ask her to know - and he was hoping that the conversation would not end too soon.

Without being aware of it, he was enjoying speaking with the woman, building a conversation word by word, heart missing a beat as she appeared ready to open her lips, curious how her words might color or recolor his own, and hoping and trusting that she felt the same way.

References

[1] The Wiki Game - Wikipedia Game - Explore Wikipedia! https://www.thewikigame.com/group.

[2] Wikipedia:Size of Wikipedia. *Wikipedia*, August 2019. Page Version ID: 911753385.

[3] Emmanuel Abbe. Community detection and stochastic block models: Recent developments. March 2017.

[4] Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding Metrics into Ultrametrics and Graphs into Spanning Trees with Constant Average Distortion. *arXiv:cs/0610003*, October 2006.

[5] Umut A Acar, Alexander T Ihler, and Ramgopal R Mettu. Adaptive Bayesian Inference. page 9, 2007.

[6] Aaron B. Adcock, Blair D. Sullivan, and Michael W. Mahoney. Tree-Like Structure in Large Social and Information Networks. pages 1–10. IEEE, December 2013.

[7] Asuman G. Aksoy and Sixian Jin. The Apple Doesn't Fall Far From the (Metric) Tree: The Equivalence of Definitions. *arXiv:1306.6092 [math]*, June 2013.

[8] David J. Aldous. The Random Walk Construction of Uniform Spanning Trees and Uniform Labelled Trees. *SIAM J. Discret. Math.*, 3(4):450–465, November 1990.

[9] Vedat Levi Alev, Nima Anari, Lap Chi Lau, and Shayan Oveis Gharan. Graph Clustering using Effective Resistance. *arXiv:1711.06530 [cs]*, November 2017.

[10] Sivaram Ambikasaran, Daniel Foreman-Mackey, Leslie Greengard, David W. Hogg, and Michael O'Neil. Fast Direct Methods for Gaussian Processes. *arXiv:1403.6015 [astro-ph, stat]*, March 2014.

[11] Sivaram Ambikasaran, Michael O'Neil, and Karan Raj Singh. Fast symmetric factorization of hierarchical matrices with applications. *arXiv:1405.0223 [physics, stat]*, May 2014.

[12] Reid Andersen, Fan Chung, and Kevin Lang. Local Graph Partitioning using PageRank Vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486, Berkeley, CA, USA, 2006. IEEE.

[13] Reid Andersen and Kevin J. Lang. Communities from seed sets. In *Proceedings of the 15th International Conference on World Wide Web*, pages 223–232. ACM, 2006.

[14] Bahman Bahmani, Abdur Chowdhury, and Ashish Goel. Fast Incremental and Personalized PageRank. *Proc. VLDB Endow.*, 4(3):173–184, December 2010.

[15] Matej Balog, Balaji Lakshminarayanan, Zoubin Ghahramani, Daniel M. Roy, and Yee Whye Teh. The Mondrian Kernel. *arXiv:1606.05241 [stat]*, June 2016.

[16] H. Bandelt and V. Chepoi. 1-Hyperbolic Graphs. *SIAM Journal on Discrete Mathematics*, 16(2):323–334, January 2003.

[17] Prithish Banerjee, Mark Vere Culp, Kenneth Jospeh Ryan, and George Michailidis. Graph-Based Semi-Supervised Learning With Big Data. *Handbook of Research on Applied Cybernetics and Systems Science*, pages 154–185, 2017.

[18] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and Semi-supervised Learning on Large Graphs. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory*, Lecture Notes in Computer Science, pages 624–638. Springer Berlin Heidelberg, 2004.

[19] Jeff A. Bilmes. Dynamic Bayesian Multinets. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 38–45, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[20] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, October 2008.

[21] Leo Breiman. Random Forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[22] Gunnar Brinkmann and Jack H. Koolen. On the Hyperbolicity of Chordal Graphs. *Annals of Combinatorics*, 5(1):61–69, June 2001.

[23] A. Broder. Generating random spanning trees. In *30th Annual Symposium on Foundations of Computer Science*, pages 442–447, October 1989.

[24] Walter Carballosa, Rocío M. Casablanca, Amauris de la Cruz, and José M. Rodríguez. Gromov Hyperbolicity in Strong Product Graphs. *The Electronic Journal of Combinatorics*, 20(3):2, July 2013.

[25] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. Neural Embeddings of Graphs in Hyperbolic Space. *arXiv:1705.10359 [cs, stat]*, May 2017.

[26] P. Chejara and W. W. Godfrey. Comparative analysis of community detection algorithms. In *2017 Conference on Information and Communication Technology (CICT)*, pages 1–5, November 2017.

[27] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević. Semi-Supervised Multiresolution Classification Using Adaptive Graph Filtering With Application to Indirect Bridge Structural Health Monitoring. *IEEE Transactions on Signal Processing*, 62(11):2879–2893, June 2014.

[28] Shi Chen, Zhi-Zhong Wang, Liang Tang, Yan-Ni Tang, Yuan-Yuan Gao, Hui-Jia Li, Ju Xiang, and Yan Zhang. Global vs local modularity for network community detection. *PLOS ONE*, 13(10):e0205284, October 2018.

[29] Siheng Chen, Rohan Varma, Aarti Singh, and Jelena Kovacevic. Representations of piecewise smooth signals on graphs. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6370–6374, Shanghai, March 2016. IEEE.

[30] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W. Mahoney. On the Hyperbolicity of Small-World and Tree-Like Random Graphs. *arXiv:1201.1717 [physics]*, January 2012.

[31] Victor Chepoi, Feodor F. Dragan, and Michel Habib. Diameters , centers , and approximating trees of δ-hyperbolic geodesic spaces and graphs $_*$ [Extended Abstract]. 2007.

[32] Gene Cheung, Enrico Magli, Yuichi Tanaka, and Michael Ng. Graph Spectral Image Processing. *arXiv:1801.04749 [eess]*, January 2018.

[33] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, May 2008.

[34] CMU DSP Grop. Fast Fourier Transform (FFT). http://www.cmlab.csie.ntu.edu.tw/cml/dsp/training/coding/transform/fft.html, 2020.

[35] R. R. Coifman and D. L. Donoho. Translation-Invariant De-Noising. In *Wavelets and Statistics*, Lecture Notes in Statistics, pages 125–150. Springer, New York, NY, 1995.

[36] Ronald R. Coifman and Mauro Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, July 2006.

[37] G. R. J. Cooper and D. R. Cowan. Comparing time series using wavelet-based semblance analysis. *Computers & Geosciences*, 34(2):95–102, February 2008.

[38] Michel Coornaert, Thomas Delzant, and Athanase Papadopoulos. *Geometrie et theorie des groupes: Les groupes hyperboliques de Gromov*. Springer, November 2006.

[39] Michele Coscia, Fosca Giannotti, and Dino Pedreschi. A Classification for Community Discovery Methods in Complex Networks. *Stat. Anal. Data Min.*, 4(5):512–546, October 2011.

[40] Alex Davies and Zoubin Ghahramani. The Random Forest Kernel and other kernels for big data from random partitions. *arXiv:1402.4293 [cs, stat]*, February 2014.

[41] Marco Di Summa, David Pritchard, and Laura Sanità. Finding the closest ultrametric. *Discrete Applied Mathematics*, 180:70–80, January 2015.

[42] Yi Ding, Risi Kondor, and Jonathan Eskreis-Winkler. Multiresolution Kernel Approximation for Gaussian Process Regression. *arXiv:1708.02183 [stat]*, August 2017.

[43] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Spectral Graph Wavelets for Structural Role Similarity in Networks. October 2017.

[44] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning Structural Node Embeddings Via Diffusion Wavelets. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining - KDD '18*, pages 1320–1329, 2018.

[45] Chengan Du and Yunpeng Zhao. On Consistency of Graph-based Semi-supervised Learning. *arXiv:1703.06177 [stat]*, March 2017.

[46] David Durfee, Rasmus Kyng, John Peebles, Anup B. Rao, and Sushant Sachdeva. Sampling random spanning trees faster than matrix multiplication. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing - STOC 2017*, pages 730–742, Montreal, Canada, 2017. ACM Press.

[47] Jan E. Holly. Pictures of Ultrametric Spaces, the p-Adic Numbers, and Valued Fields. *American Mathematical Monthly*, 108, October 2001.

[48] Sandra Ebert, Mario Fritz, and Bernt Schiele. Semi-Supervised Learning on a Budget: Scaling Up to Large Datasets. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Computer Vision – ACCV 2012*, Lecture Notes in Computer Science, pages 232–245. Springer Berlin Heidelberg, 2013.

[49] V. N. Ekambaram, G. Fanti, B. Ayazifar, and K. Ramchandran. Wavelet-regularized graph semi-supervised learning. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 423–426, December 2013.

[50] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-Stretch Spanning Trees. *SIAM Journal on Computing*, 38, December 2004.

[51] Jonathan Eskreis-Winkler and Risi Kondor. Effective Resistance-based Germination of Seed Sets for Community Detection. *arXiv:1811.12162 [cs, stat]*, October 2018.

[52] Philippe Esling and Carlos Agon. Time-series Data Mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, December 2012.

[53] Eugene F. Fama. Random Walks in Stock Market Prices. *Financial Analysts Journal*, 51(1):75–80, 1995.

[54] Leonardo N. Ferreira and Liang Zhao. Time Series Clustering via Community Detection in Networks. *Information Sciences*, 326:227–242, January 2016.

[55] Dániel Fogaras, Balázs Rácz, Károly Csalogány, and Tamás Sarlós. Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments. *Internet Mathematics*, 2(3):333–358, January 2005.

[56] Kristin J. Forbes and Roberto Rigobon. No Contagion, Only Interdependence: Measuring Stock Market Comovements. *The Journal of Finance*, 57(5):2223–2261, 2002.

[57] S. Fortunato and M. Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, January 2007.

[58] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, February 2010.

[59] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, November 2016.

[60] Chao Gao, Zongming Ma, Anderson Y. Zhang, and Harrison H. Zhou. Community Detection in Degree-Corrected Block Models. *arXiv:1607.06993 [cs, math, stat]*, July 2016.

[61] J. O. Garcia, A. Ashourvan, S. Muldoon, J. M. Vettel, and D. S. Bassett. Applications of Community Detection Techniques to Brain Graphs: Algorithmic Considerations and Implications for Neural Function. *Proceedings of the IEEE*, 106(5):846–867, May 2018.

[62] Ullas Gargi, Wenjun Lu, Vahab S. Mirrokni, and Sangho Yoon. Large-Scale Community Detection on YouTube for Topic Discovery and Exploration. In *Proceedings of 5th International AAAI Conference on Weblogs and Social Media*, January 2011.

[63] Matan Gavish, Boaz Nadler, and Ronald Coifman. Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi Supervised Learning. pages 367–374, August 2010.

[64] Amir Ghasemian, Homa Hosseinmardi, and Aaron Clauset. Evaluating Overfit and Underfit in Models of Network Community Structure. *arXiv:1802.10582 [physics, q-bio, stat]*, February 2018.

[65] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[66] Palash Goyal and Emilio Ferrara. Graph Embedding Techniques, Applications, and Performance: A Survey. *arXiv:1705.02801 [physics]*, May 2017.

[67] M. Gromov. Hyperbolic Groups. In S. M. Gersten, editor, *Essays in Group Theory*, Mathematical Sciences Research Institute Publications, pages 75–263. Springer New York, New York, NY, 1987.

[68] C. Guo, H. Jia, and N. Zhang. Time Series Clustering Based on ICA for Stock Data Analysis. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4, October 2008.

[69] Matthias Hamann. Spanning trees in hyperbolic graphs. *arXiv:0910.5605 [math]*, October 2009.

[70] Matthias Hamann. On the tree-likeness of hyperbolic spaces. *arXiv:1105.3925 [math]*, May 2011.

[71] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on Graphs via Spectral Graph Theory. *arXiv:0912.3848 [cs, math]*, December 2009.

[72] Thomas Little Heath. *The Thirteen Books of Euclid's Elements*. Courier Corporation, 1956.

[73] Alexandre Hollocou, Thomas Bonald, and Marc Lelarge. Multiple Local Community Detection. *SIGMETRICS Perform. Eval. Rev.*, 45(3):76–83, March 2018.

[74] Jeff Irion and Naoki Saito. The generalized Haar-Walsh transform. In *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pages 472–475, Gold Coast, Australia, June 2014. IEEE.

[75] Jeff Irion and Naoki Saito. Hierarchical graph Laplacian eigen transforms. *JSIAM Letters*, 6(0):21–24, 2014.

[76] Tommi Jaakkola, Marina Meila, and Tony Jebara. Maximum Entropy Discrimination. In S. A. Solla, T. K. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 470–476. MIT Press, 2000.

[77] Tony Jebara and Philip M Long. Tree Dependent Identically Distributed Learning. page 8.

[78] Tony S. Jebara. Bayesian Out-Trees. *arXiv:1206.3269 [cs, stat]*, June 2012.

[79] Ming Ji, Tianbao Yang, Binbin Lin, Rong Jin, and Jiawei Han. A Simple Algorithm for Semi-supervised Learning with Improved Generalization Error Bound. *arXiv:1206.6412 [cs, stat]*, June 2012.

[80] Rie Johnson and Tong Zhang. Graph-Based Semi-Supervised Learning and Spectral Kernel Design. *IEEE Transactions on Information Theory*, 54(1):275–288, January 2008.

[81] Bisma S. Khan and Muaz A. Niazi. Network Community Detection: A Review and Visual Survey. *arXiv:1708.00977 [cs]*, August 2017.

[82] Jongkwang Kim and Thomas Wilhelm. Spanning tree separation reveals community structure in networks. *Physical Review E*, 87(3), March 2013.

[83] R. Kleinberg. Geographic Routing Using Hyperbolic Space. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pages 1902–1909, Anchorage, AK, USA, 2007. IEEE.

[84] Isabel Kloumann, Johan Ugander, and Jon Kleinberg. Block Models and Personalized PageRank. *Proceedings of the National Academy of Sciences*, 114(1):33–38, January 2017.

[85] Risi Kondor, Nedelina Teneva, and Vikas K Garg. Multiresolution Matrix Factorization. page 9, 2014.

[86] Risi Kondor, Nedelina Teneva, and Pramod K. Mudrakarta. Parallel MMF: A Multiresolution Approach to Matrix Computation. *arXiv:1507.04396 [cs, stat]*, July 2015.

[87] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marian Boguna. Hyperbolic Geometry of Complex Networks. *Physical Review E*, 82(3), September 2010.

[88] Zhana Kuncheva and Giovanni Montana. Multi-scale Community Detection in Temporal Networks Using Spectral Graph Wavelets. *arXiv:1708.04060 [physics, stat]*, August 2017.

[89] Daniel Lamprecht, Kristina Lerman, Denis Helic, and Markus Strohmaier. How the structure of Wikipedia articles influences user navigation. *The New Review of Hypermedia and Multimedia*, 23(1):29–50, January 2017.

[90] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, July 2009.

[91] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[92] Marc T. Law, Jake Snell, and Richard S. Zemel. Lorentzian Distance Learning. September 2018.

[93] Ann B. Lee, Boaz Nadler, and Larry Wasserman. Treelets—An adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, 2(2):435–471, June 2008.

[94] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[95] Silvio Levy. *Flavors of Geometry*. Cambridge University Press, September 1997.

[96] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. Label Efficient Semi-Supervised Learning via Graph Filtering. *arXiv:1901.09993 [cs, stat]*, January 2019.

[97] Wei Liu, Junfeng He, and Shih-Fu Chang. Large Graph Construction for Scalable Semi-supervised Learning. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 679–686, USA, 2010. Omnipress.

[98] Linyuan Lü and Tao Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, March 2011.

[99] Ulrike V. Luxburg, Agnes Radl, and Matthias Hein. Getting lost in space: Large sample analysis of the resistance distance. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2622–2630. Curran Associates, Inc., 2010.

[100] F. Malgouyres and J. Landsberg. On the identifiability and stable recovery of deep/multi-layer structured matrix factorization. In *2016 IEEE Information Theory Workshop (ITW)*, pages 315–319, September 2016.

[101] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[102] Andrew Mehler and Steven Skiena. Expanding Network Communities from Representative Examples. *ACM Trans. Knowl. Discov. Data*, 3(2):7:1–7:27, April 2009.

[103] Marina Meil and Michael I Jordan. Learning with Mixtures of Trees. page 48.

[104] Gal Mishne, Ronen Talmon, Israel Cohen, Ronald R. Coifman, and Yuval Kluger. Data-Driven Tree Transforms and Metrics. *IEEE transactions on signal and information processing over networks*, 4(3):451–466, September 2018.

[105] Pramod Kaushik Mudrakarta and Risi Kondor. A generic multiresolution preconditioner for sparse symmetric systems. *arXiv:1707.02054 [math]*, July 2017.

[106] Fionn Murtagh. The Haar Wavelet Transform of a Dendrogram. *J Classif*, 24:3–32, June 2007.

[107] Boaz Nadler, Nathan Srebro, and Xueyuan Zhou. *Semi-Supervised Learning with the Graph Laplacian: The Limit of Infinite Unlabelled Data*. 2009.

[108] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, June 2006.

[109] Canh Hao Nguyen and Hiroshi Mamitsuka. New resistance distances with global information on large graphs. In *Artificial Intelligence and Statistics*, pages 639–647, 2016.

[110] Maximilian Nickel and Douwe Kiela. Poincar\'e Embeddings for Learning Hierarchical Representations. *arXiv:1705.08039 [cs, stat]*, May 2017.

[111] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. http://ilpubs.stanford.edu:8090/422/, November 1999.

[112] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.

[113] Tiago P. Peixoto. Hierarchical Block Structures and High-resolution Model Selection in Large Networks. *Physical Review X*, 4(1), March 2014.

[114] John Playfair and William Wallace. *Elements of Geometry; Containing the First Six Books of Euclid, with Two Books on the Geometry of Solids. To Which Are Added, Elements of Plane and Spherical Trigonometry. By J. Playfair. With a Suppl. on the Quadrature of the Circle and the Geometry of Solids.* 1826.

[115] I. Ram, M. Elad, and I. Cohen. Generalized Tree-Based Wavelet Transform. *IEEE Transactions on Signal Processing*, 59(9):4199–4209, September 2011.

[116] Elizaveta Rebrova, Gustavo Chavez, Yang Liu, Pieter Ghysels, and Xiaoye Sherry Li. A Study of Clustering Techniques and Hierarchical Matrix Formats for Kernel Ridge Regression. *arXiv:1803.10274 [cs, stat]*, March 2018.

[117] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, July 2006.

[118] Martin Rosvall and Carl T. Bergstrom. Maps of Information Flow Reveal Community Structure In Complex Networks. 2007.

[119] Marta Sales-Pardo, Roger Guimerà, André A. Moreira, and Luís A. Nunes Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences of the United States of America*, 104(39):15224–15229, September 2007.

[120] Michael T. Schaub, Maguy Trefois, Paul van Dooren, and Jean-Charles Delvenne. Sparse Matrix Factorizations for Fast Linear Solvers with Application to Laplacian Systems. *SIAM Journal on Matrix Analysis and Applications*, 38(2):505–529, January 2017.

[121] Saray Shai, Natalie Stanley, Clara Granell, Dane Taylor, and Peter J. Mucha. Case studies in network community detection. May 2017.

[122] Uri Shalit and Gal Chechik. Efficient coordinate-descent for orthogonal matrices through Givens rotations. *arXiv:1312.0624 [cs, stat]*, December 2013.

[123] James Sharpnack, Aarti Singh, and Akshay Krishnamurthy. Detecting Activations over Graphs using Spanning Tree Wavelet Bases. In *Artificial Intelligence and Statistics*, pages 536–544, April 2013.

[124] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. *IEEE/ACM Transactions on Networking*, 12(6):993–1006, December 2004.

[125] Y. Shavitt and T. Tankel. Hyperbolic Embedding of Internet Graph for Distance Estimation and Overlay Construction. *IEEE/ACM Transactions on Networking*, 16(1):25–36, February 2008.

[126] David I. Shuman, Mohammadjavad Faraji, and Pierre Vandergheynst. Semi-Supervised Learning with Spectral Graph Wavelets. https://infoscience.epfl.ch/record/164765, 2011.

[127] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre 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(3):83–98, May 2013.

[128] Alexander J. Smola and Risi Kondor. Kernels and Regularization on Graphs. In *Learning Theory and Kernel Machines*, Lecture Notes in Computer Science, pages 144–158. Springer, Berlin, Heidelberg, 2003.

[129] Radomir S. Stanković and Bogdan J. Falkowski. The Haar wavelet transform: Its status and achievements. *Computers & Electrical Engineering*, 29(1):25–44, January 2003.

[130] Nedelina Teneva, Pramod Kaushik Mudrakarta, and Risi Kondor. Multiresolution Matrix Compression. In *PMLR*, pages 1441–1449, May 2016.

[131] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincar\'e GloVe: Hyperbolic Word Embeddings. *arXiv:1810.06546 [cs]*, October 2018.

[132] TRBC Financial. Thomson Reuters Business Classification: Methodology. Technical report, January 2012.

[133] N. Tremblay and P. Borgnat. Graph Wavelets for Multiscale Community Mining. *IEEE Transactions on Signal Processing*, 62(20):5227–5239, October 2014.

[134] Catherine Tuglus and Mark J. van der Laan. DISCUSSION OF: TREELETS—AN ADAPTIVE MULTI-SCALE BASIS FOR SPARSE UNORDERED DATA. *The annals of applied statistics*, 2(2):489–493, June 2008.

[135] Twan van Laarhoven. Generative models for local network community detection. *Physical Review E*, 97(4), April 2018.

[136] Verbeek, Kevin and Suri, Subhash. Metric Embedding, Hyperbolic Space, and Social Networks$_*$. June 2014.

[137] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu. Scalable Semi-Supervised Learning by Efficient Anchor Graph Regularization. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1864–1877, July 2016.

[138] T. Warren Liao. Clustering of time series data—a survey. *Pattern Recognition*, 38(11):1857–1874, November 2005.

[139] Larry Wasserman and John D Lafferty. Statistical Analysis of Semi-Supervised Regression. page 8, 2008.

[140] Joyce Jiyoung Whang, David F. Gleich, and Inderjit S. Dhillon. Overlapping Community Detection Using Seed Set Expansion. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2099–2108, New York, NY, USA, 2013. ACM.

[141] A. S. Willsky. Multiresolution Markov models for signal and image processing. *Proceedings of the IEEE*, 90(8):1396–1458, August 2002.

[142] Yaokun Wu and Chengpeng Zhang. Chordality and hyperbolicity of a graph. *arXiv:0910.3544 [math]*, October 2009.

[143] Jierui Xie, Stephen Kelley, and Boleslaw K. Szymanski. Overlapping Community Detection in Networks: The State-of-the-art and Comparative Study. *ACM Comput. Surv.*, 45(4):43:1–43:35, August 2013.

[144] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph Wavelet Neural Network. September 2018.

[145] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[146] Liang Yang, Xiaochun Cao, Di Jin, Xiao Wang, and Dan Meng. A Unified Semi-Supervised Community Detection Framework Using Latent Space Graph Regularization. *IEEE transactions on cybernetics*, 45(11):2585–2598, November 2015.

[147] Zhao Yang, René Algesheimer, and Claudio J. Tessone. A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Scientific Reports*, 6:30750, August 2016.

[148] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings. *arXiv:1603.08861 [cs]*, March 2016.

[149] Meichen Yu, Arjan Hillebrand, Prejaas Tewarie, Jil Meier, Bob van Dijk, Piet Van Mieghem, and Cornelis Jan Stam. Hierarchical clustering in minimum spanning trees. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(2):023107, February 2015.

[150] X. Zhao, A. Sala, H. Zheng, and B. Y. Zhao. Efficient shortest paths on massive social graphs. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 77–86, October 2011.

[151] Zhenyue Zhang, Hongyuan Zha, and Min Zhang. Spectral methods for semi-supervised manifold learning. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6, June 2008.

[152] Dengyong Zhou, Thomas Hofmann, and Bernhard Schölkopf. Semi-supervised Learning on Directed Graphs. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1633–1640. MIT Press, 2005.

[153] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V. Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, May 2017.

[154] X. Zhu, A. B. Goldberg, and T. Khot. Some new directions in graph-based semi-supervised learning. In *2009 IEEE International Conference on Multimedia and Expo*, pages 1504–1507, June 2009.

[155] Xiaojin Zhu. Semi-supervised learning : From Gaussian fields to Gaussian processes. 2003.

[156] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pages 912–919, Washington, DC, USA, 2003. AAAI Press.

[157] Xiaojin Zhu, John Lafferty, Ronald Rosenfeld, Zoubin Ghahramani, and Tommi Jaakkola. Semi-Supervised Learning with Graphs. page 174.