Learning Neural Graph Representations in Non-Euclidean Geometries

Dissertation
zur Erlangung der Doktorwürde
der Neuphilologischen Fakultät
der Ruprecht-Karls-Universität Heidelberg

vorgelegt von

Federico López

Referent: Prof. Dr. Michael Strube
Korreferent: Prof. Dr. Anette Frank
Korreferent: Prof. Dr. Anna Wienhard
Einreichung: 01 Dezember 2021

Abstract

The success of Deep Learning methods is heavily dependent on the choice of the data representation. For that reason, much of the actual effort goes into Representation Learning, which seeks to design preprocessing pipelines and data transformations that can support effective learning algorithms. The aim of Representation Learning is to facilitate the task of extracting useful information for classifiers and other predictor models. In this regard, graphs arise as a convenient data structure that serves as an intermediary representation in a wide range of problems. The predominant approach to work with graphs has been to embed them in an Euclidean space, due to the power and simplicity of this geometry. Nevertheless, data in many domains exhibit non-Euclidean features, making embeddings into Riemannian manifolds with a richer structure necessary. The choice of a metric space where to embed the data imposes a geometric inductive bias, with a direct impact on the performance of the models.

This thesis is about learning neural graph representations in non-Euclidean geometries and showcasing their applicability in different downstream tasks. We introduce a toolkit formed by different graph metrics with the goal of characterizing the topology of the data. In that way, we can choose a suitable target embedding space aligned to the shape of the dataset. By virtue of the geometric inductive bias provided by the structure of the non-Euclidean manifolds, neural models can achieve higher performances with a reduced parameter footprint.

As a first step, we study graphs with hierarchical structures. We develop different techniques to derive hierarchical graphs from large label inventories. Noticing the capacity of hyperbolic spaces to represent tree-like arrangements, we incorporate this information into an NLP model through hyperbolic graph embeddings and showcase the higher performance that they enable.

Second, we tackle the question of how to learn hierarchical representations suited for different downstream tasks. We introduce a model that jointly learns task-specific graph embeddings from a label inventory and performs classification in hyperbolic space. The model achieves state-of-the-art results on very fine-grained labels, with a remarkable reduction of the parameter size.

Next, we move to matrix manifolds to work on graphs with diverse structures and properties. We propose a general framework to implement the mathematical tools required to learn graph embeddings on symmetric spaces. These spaces are of particular interest given that they have a compound geometry that simultaneously contains Euclidean as well as hyperbolic subspaces, allowing them to automatically adapt to dissimilar features in the graph. We demonstrate a concrete implementation of the framework on Siegel spaces, showcasing their versatility on different tasks.

Finally, we focus on multi-relational graphs. We devise the means to translate Euclidean and hyperbolic multi-relational graph embedding models into the space of symmetric positive definite (SPD) matrices. To do so we develop gyrocalculus in this geometry and integrate it with the aforementioned framework.

Zusammenfassung

Der Erfolg von Deep Learning-Methoden hängt stark von der Wahl der Datendarstellung ab. Aus diesem Grund fließt ein Großteil der aktuellen Bemühungen in das Repräsentationslernen, das darauf abzielt, Vorverarbeitungspipelines und Datentransformationen zu entwickeln, die effektive Lernalgorithmen unterstützen können. Ziel des Repräsentationslernens ist es, die Extraktion nützlicher Informationen für Klassifikatoren und andere Prädiktorenmodelle zu erleichtern. In diesem Zusammenhang erweisen sich Graphen als eine geeignete Datenstruktur, die bei einer Vielzahl von Problemen als Zwischenrepräsentation dient. Der vorherrschende Ansatz bei der Arbeit mit Graphen war, sie in einen euklidischen Raum einzubetten, da diese Geometrie sehr leistungsfähig und einfach ist. Dennoch weisen Daten in vielen Bereichen nicht-euklidische Merkmale auf, so dass Einbettungen in Riemannsche Mannigfaltigkeiten mit einer reicheren Struktur erforderlich sind. Die Wahl eines metrischen Raums, in den die Daten eingebettet werden sollen, führt zu einer geometrischen induktiven Verzerrung, die sich direkt auf die Leistung der Modelle auswirkt.

In dieser Arbeit geht es darum, neuronale Graphenrepräsentationen in nicht-euklidischen Geometrien zu lernen und ihre Anwendbarkeit in verschiedenen nachgelagerten Aufgaben zu demonstrieren. Wir stellen ein Toolkit vor, das aus verschiedenen Graphenmetriken besteht, mit dem Ziel, die Topologie der Daten zu charakterisieren. Auf diese Weise können wir einen geeigneten Ziel-Einbettungsraum wählen, der auf die Form des Datensatzes abgestimmt ist. Aufgrund der geometrischen induktiven Verzerrung, die durch die Struktur der nicht-euklidischen Mannigfaltigkeiten gegeben ist, können neuronale Modelle eine höhere Leistung mit einem geringeren Parameterbedarf erzielen.

In einem ersten Schritt untersuchen wir Graphen mit hierarchischen Strukturen. Wir entwickeln verschiedene Techniken zur Ableitung hierarchischer Graphen aus großen Etikettenbeständen. Wir stellen fest, dass hyperbolische Räume in der Lage sind, baumähnliche Anordnungen zu repräsentieren, und integrieren diese Information in ein NLP-Modell durch hyperbolische Grapheneinbettungen, die eine höhere Leistung ermöglichen.

Zweitens beschäftigen wir uns mit der Frage, wie hierarchische Repräsentationen für verschiedene nachgelagerte Aufgaben gelernt werden können. Wir stellen ein Modell vor, das gemeinsam aufgabenspezifische Grapheneinbettungen aus einem Etiketteninventar erlernt und eine Klassifizierung im hyperbolischen Raum durchführt. Das Modell erzielt Spitzenergebnisse bei sehr feinkörnigen Beschriftungen mit einer bemerkenswerten Reduzierung der Parametergröße.

Als nächstes gehen wir zu Matrix-Mannigfaltigkeiten über, um Graphen mit verschiedenen Strukturen und Eigenschaften zu bearbeiten. Wir schlagen einen allgemeinen Rahmen vor, um die mathematischen Werkzeuge zu implementieren, die für das Lernen von Grapheneinbettungen in symmetrischen Räumen erforderlich sind. Diese Räume sind von besonderem Interesse, da sie eine zusammengesetzte Geometrie haben, die sowohl euklidische als auch hyperbolische Unterräume enthält, so dass sie sich automatisch an unterschiedliche Merkmale im Graphen anpassen können. Wir demonstrieren eine konkrete Implementierung des Rahmens für Siegel-Räume, um ihre Vielseitigkeit bei verschiedenen Aufgaben zu demonstrieren.

Schließlich konzentrieren wir uns auf multirelationale Graphen. Wir entwickeln Mittel, um euklidische und hyperbolische multirelationale Grapheneinbettungsmodelle in den Raum der symmetrischen positiv definiten Matrizen (SPD) zu übersetzen. Zu diesem Zweck entwickeln wir den Kreiselkalkül in dieser Geometrie und integrieren ihn in den oben genannten Rahmen.

Diese Zusammenfassung wurde automatisch mit neuronalen maschinellen Übersetzungswerkzeugen übersetzt. Wenn Sie der Meinung sind, dass es noch Raum für Verbesserungen gibt, sollten Sie einen Ph.D. in Computerlinguistik in Erwägung ziehen.

Acknowledgements

First, I want to thank my supervisor Michael Strube for giving me the opportunity to pursue this PhD, and for the scientific freedom, trust, and guidance through all these years. Michael helped me to establish research collaborations and encouraged me to seek internships, which were decisive to the result of this thesis. I would also like to express my gratitude to Benjamin Heinzerling, Anna Wienhard, Beatrice Pozzetti, and Steve Trettel for bearing with me with enormous patience and kindness. For me, it has been an honor to work with them, and our collaborations allowed me to reach places that I could only dream of.

I would like to thank Mark-Christoph Müller for providing his feedback on drafts of this thesis, and to the NLP group at HITS for creating an excellent working environment where I always felt comfortable and respected.

I am also very thankful to my professors Rosita Wachenchauzer and Luis Argerich, and to Marga, Maxi, Alberto, Melisa, and Pablo for transferring their devotion to maths and computer science, but most importantly, for their constant example of hard work, enormous wisdom and enormous humility.

My time in Heidelberg has been a wonderful journey, with some interesting detours due to the pandemic. I want to thank the people who helped me not to lose it, particularly during those times. Sincere thanks go to the international community of PhD students and colleagues that I met at HITS, and to my most amazing flatmates. *Zum Wohl!*

Endless thanks to *La Generación Dorada*, the group on which the sun never sets, and to my lifelong friends from Argentina, for backing me 24-7 and for their invaluable friendship.

Last but not least, I will forever be grateful to my family, without them and their infinite affection and support across the ocean, none of this would be possible, and to Eva, my love and companion in all these years.

Contents

I Preliminaries **1**

1 Introduction **3**

1.1 Graph Embedding Problem 4
1.1.1 Problem Formulation 5
1.1.2 Graphs are non-Euclidean 7
1.1.3 Transductive and Inductive Settings 7
1.1.4 Supervised and Unsupervised Settings 8
1.1.5 Node Features for Graph Embeddings 8
1.2 Why non-Euclidean Geometries? 9
1.3 Research Questions 10
1.4 Contributions 11
1.5 Published Work 12

2 Non-Euclidean Geometry **15**

2.1 Brief History 16
2.2 Riemannian Geometry 19
2.3 Hyperbolic Geometry 20
2.3.1 Poincaré Model 21
2.3.2 Lorentz Model 22
2.4 Space of Symmetric Positive Definite Matrices 23
2.5 Symmetric Spaces, Distances, and Metrics 25
2.5.1 Symmetric Spaces 25
2.5.2 Vector-valued Distance 27
2.5.3 Finsler Metrics 28
2.6 Gyrovector Spaces 29
2.6.1 Gyrovector Spaces for Hyperbolic Geometry 30

3 Graphs **33**

3.1 Definitions from Graph Theory 33
3.2 Measures for Graph Analysis 34

3.2.1 Curvature Analysis . . . 35
3.2.2 δ-hyperbolicity . . . 36

4 Geometric Deep Learning 39
4.1 Manifold Learning . . . 40
4.2 Why Should We Go non-Euclidean? . . . 41
4.2.1 Universal Approximation Theorem . . . 41
4.2.2 Nash Embedding Theorem . . . 42
4.2.3 Curse of Dimensionality . . . 42
4.2.4 Limitations of Euclidean Representations . . . 43
4.3 Geometric Inductive Bias . . . 43
4.3.1 Advantages . . . 44
4.3.2 Challenges . . . 44

5 Related Work and Applications 47
5.1 Graph Embedding Methods . . . 47
5.1.1 Outer Product-based Methods . . . 47
5.1.2 Euclidean Distance-based Methods . . . 50
5.1.3 Non-Euclidean Distance-based Methods . . . 52
5.2 Tasks and Applications . . . 55
5.3 Limitations . . . 58

II Embeddings Graphs in Hyperbolic Space 59

6 Constructing and Exploiting Hierarchical Graphs 61
6.1 Entity Typing in Hyperbolic Space . . . 62
6.1.1 Related Work . . . 63
6.1.2 Objective . . . 64
6.2 Hierarchical Type Inventories . . . 65
6.2.1 Data . . . 65
6.2.2 Deriving the Hierarchies . . . 66
6.2.3 Graph Analysis . . . 68
6.2.4 Embedding the Hierarchies . . . 70
6.3 Model . . . 70
6.3.1 Mention and Context Representations . . . 72
6.3.2 Projecting into the Ball . . . 73
6.3.3 Optimization of the Model . . . 74
6.4 Experiments . . . 75
6.4.1 Setup . . . 75

6.5 Results and Discussion . . . 76
6.5.1 Comparison of the Hierarchies . . . 76
6.5.2 Comparison of the Spaces . . . 77
6.5.3 Error Analysis . . . 78
6.5.4 Analysis Case: OntoNotes . . . 79
6.6 Conclusions . . . 80

7 Inferring the Hierarchy with a Fully Hyperbolic Model 81
7.1 Label Embeddings as Graph Embeddings . . . 82
7.2 Hyperbolic Neural Networks . . . 84
7.3 Fully Hyperbolic Classification Model . . . 85
7.3.1 Mention Encoder . . . 85
7.3.2 Context Encoder . . . 86
7.3.3 Concatenation . . . 86
7.3.4 Distance-based Attention . . . 87
7.3.5 Classification in the Poincaré Ball . . . 88
7.3.6 Optimization . . . 88
7.4 Experiments . . . 88
7.4.1 Data . . . 89
7.4.2 Setup . . . 89
7.4.3 Baselines . . . 89
7.5 Results and Discussion . . . 90
7.6 Ablations and Analysis . . . 92
7.6.1 Comparison of the Spaces . . . 92
7.6.2 Word Embeddings Ablation . . . 93
7.6.3 Component Ablation . . . 94
7.6.4 OntoNotes Dataset . . . 94
7.7 Conclusions . . . 95

III Embeddings Graphs in Matrix Manifolds 97

8 A Framework for Graph Embeddings on Symmetric Spaces 99
8.1 Symmetric Spaces for Embedding Problems . . . 101
8.1.1 Vector-valued Distance . . . 101
8.1.2 Finsler Distances . . . 102
8.2 The SYMPA Framework . . . 103
8.2.1 Choosing a Symmetric Space . . . 103
8.2.2 Choosing a Model of the Symmetric Space . . . 104
8.2.3 Computing Distances . . . 104

8.2.4 Computing Gradients 105
8.3 SYMPA on Siegel Spaces 105
8.3.1 Siegel Space 106
8.3.2 Models of Siegel Spaces 106
8.3.3 Computing Distances on Siegel Spaces 107
8.3.4 Riemannian Optimization on Siegel Spaces 108
8.4 Graph Reconstruction 109
8.4.1 Experimental Setup 109
8.4.2 Synthetic Graphs 111
8.4.3 Real-world Graphs 113
8.4.4 High-dimensional Spaces 115
8.4.5 New Tools to Analyze the Embedding Space 116
8.5 Recommender Systems 119
8.5.1 Experimental Setup 120
8.5.2 Results 120
8.6 Node Classification 121
8.6.1 Experimental Setup 121
8.6.2 Results 122
8.7 Conclusions 123

9 Representing Multi-Relational Graphs on SPD Manifolds 125
9.1 Related Work on SPD 127
9.2 SYMPA on the Space SPD_n 128
9.2.1 Space and Model for SPD_n 128
9.2.2 Computing Distances in SPD_n 130
9.2.3 Computing Gradients on SPD_n 131
9.3 Gyrocalculus on SPD_n 131
9.3.1 Addition and Subtraction 132
9.3.2 Scalar Multiplication and Matrix Scaling 133
9.4 Implementation 133
9.4.1 Embeddings in SPD_n and S_n 133
9.4.2 Isometries: Rotations and Reflections 134
9.4.3 Optimization 134
9.4.4 Complexity 135
9.5 Knowledge Graph Completion 135
9.5.1 Problem Formulation 135
9.5.2 Data 135
9.5.3 Related Work for Knowledge Graph Completion 137
9.5.4 Proposed Models 138

9.5.5 Experimental Setup 138
9.5.6 Results 140
9.5.7 Visualizations through the VVD 141
9.6 Knowledge Graph-based Recommender Systems 142
9.6.1 Data 143
9.6.2 Experimental Setup 144
9.6.3 Results 144
9.7 Conclusions 145

IV Conclusions 147

10 Conclusions and Future Work 149
10.1 Conclusions 149
10.2 Future Research Directions 150

V Appendices 153

A Gyrocalculus 155

B Symmetric Spaces 161

C Siegel Spaces 167

D Differential Geometry of SPD_n 181

E Code and Data Used in this Thesis 195

Bibliography 196

Part I

Preliminaries

Chapter 1

Introduction

"We can't solve problems by using the same kind of thinking we used when we created them."
– Albert Einstein

The goal of representation learning is to embed real-world data, frequently modeled on a graph, into an ambient space. This embedding space can then be used to analyze and perform tasks on the discrete graph, with Deep Learning and neural networks as the *de facto* methodology.

The predominant approach has been to embed discrete structures in an Euclidean space due to its power and simplicity (Skopek et al., 2020). Nonetheless, it would be a remarkable coincidence if Euclidean geometry were both the only geometry that practitioners had tried and the optimal geometry for all problems (Chamberlain et al., 2019). In fact, data in many domains exhibit non-Euclidean features (Krioukov et al., 2010; Bronstein et al., 2017), making embeddings into Riemannian manifolds with a richer structure necessary. The choice of a metric space where to embed the data can be understood as selecting a geometric inductive bias (Tifrea et al., 2019), which does not always adapt to all parts of the graphs.

However, replacing the target embedding space with non-Euclidean manifolds is a non-trivial endeavour. Geometric objects such as geodesic equations, exponential map, or distance function can easily loose their appealing closed-form expressions when working with generic manifolds. Furthermore, operations like convolutions cannot be directly applied on irregular domains. In particular for graph-structured data, it is challenging to define networks with strong structural priors, as structures can be arbitrary, and vary significantly across different graphs and even across different nodes within the same graph (Chami et al., 2020a).

This thesis is about learning neural graph representations in non-Euclidean geometries,

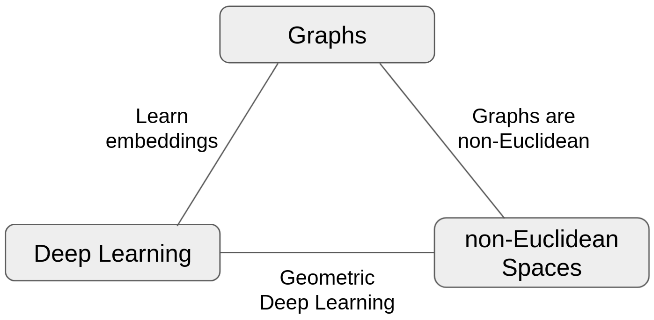


Figure 1.1: Diagram of the main themes covered in this thesis, and the relations that we seek to explore among them.

and showcasing their applicability in different downstream tasks. We introduce a toolkit formed by different graph metrics. The goal of these metrics is to characterize the topology of the data in order to match it with a suitable target embedding space. In this way, neural models can profit from the geometric inductive bias provided by the structure of the non-Euclidean manifolds. Moreover, we propose a general framework for learning graph embeddings in symmetric spaces. This framework allows us to develop tools and neural components to embed data in different Riemannian manifolds, and then operate with it.

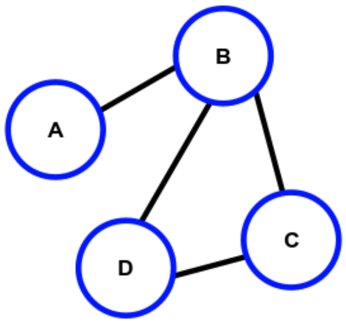
The three pillars on which this thesis stands are graphs, non-Euclidean geometry, and Deep Learning. In Figure 1.1, we represent these three main themes through a graph, together with the relations we aim to explore between them. Overall, we advocate for alternative representation methods, with a well-established mathematical foundation. We hope that this work eases the adoption of non-Euclidean tools and components into neural models from diverse domains, yielding more efficient systems.

In the remainder of this chapter, we first describe the problem we work on and its applications (§1.1). We then discuss the main questions addressed in this thesis (§1.3) and summarize its contributions (§1.4). Finally, we point to our publications related to this dissertation (§1.5).

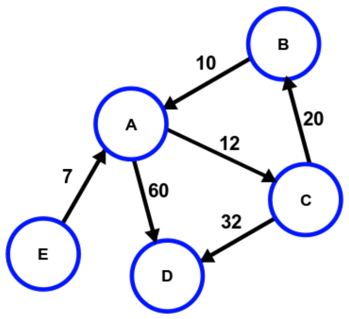
1.1 Graph Embedding Problem

Graphs are a ubiquitous data structure and a universal language for describing complex systems. In the most general view, a graph is simply a collection of objects (i.e., nodes, points or vertices), along with a set of interactions (i.e., edges) between pairs of these objects.

Definition 1.1 (Graph)**.** A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes (or vertices) $\mathcal{V}$ and a set of edges $\mathcal{E}$ between these nodes. We denote an edge going from node $u \in \mathcal{V}$ to node $v \in \mathcal{V}$ as $(u, v) \in \mathcal{E}$. A graph is **undirected** if $(u, v) \in \mathcal{E}$ implies that also $(v, u) \in \mathcal{E}$, i.e. the relationships are symmetric. On the other hand, if $(u, v) \in \mathcal{E}$ does not necessarily



(a) Undirected unweighted graph.



(b) Directed weighted graph.

Figure 1.2: Different types of graphs.

implies that $(v, u) \in \mathcal{E}$, then the graph is **directed** (also called *digraph*). Finally, a graph is **weighted** if there exist a weight function: $w : (u, v) \rightarrow w_{u,v}$ that assigns weight $w_{u,v}$ to the edge connecting the nodes $u, v \in \mathcal{V}$ (see Figure 1.2).

For example, to encode a social network we can use a graph, where nodes represent individuals and edges represent that two individuals are friends. Beyond the distinction between undirected, directed and weighted edges, we will also consider graphs that have different types of edges, such as labels or relations. We refer to those graphs as a multi-relational graph. Figure 1.1 is an example of a multi-relational graph, where edges are labeled with a relation between the nodes.

1.1.1 Problem Formulation

Graph embedding is the task that aims at learning a mapping function from a discrete graph to a continuous domain. Formally, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the goal is to learn low-dimensional representations $\{Z_i\}_{i \in \mathcal{V}}$ (embeddings) for nodes in the graph $\{v_i\}_{i \in \mathcal{V}}$, such that important graph properties (e.g. local or global structure) are preserved in the embedding space (see Figure 1.3). For instance, if two nodes have similar connections in the original graph, their learned vector representations should be close. In the conventional setup, machine learning models learn a node[1] embedding matrix $Z \in \mathbb{R}^{|\mathcal{V}| \times d}$, with $d \ll |\mathcal{V}|$ for scalability purposes, where each row is an Euclidean vector in $\mathbb{R}^d$.

The choice of the Euclidean space $\mathbb{R}^d$ as target embedding space is very common in machine learning. This is the natural generalization of our intuition-friendly, visual three-dimensional space. Other reasons for such a choice are the vectorial structure and simple closed-form expressions of the distance, inner-product and geodesics (straight lines).

[1]We present the problem yielding node embeddings $Z \in \mathbb{R}^{|\mathcal{V}| \times d}$. However, it can also be extended to models that embed an entire graph with $Z \in \mathbb{R}^d$ as a d-dimensional vector for the whole graph, or embed graph edges $Z \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}| \times d}$ as a (potentially sparse) 3D matrix with $Z_{u,v} \in \mathbb{R}^d$ representing the embedding of edge (u, v), or for models for multi-relational graphs that also learn relation embeddings $U \in \mathbb{R}^{|\mathcal{R}| \times d}$.

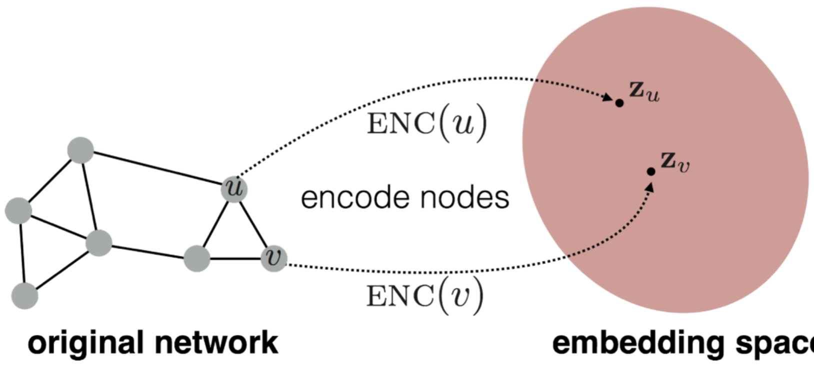


Figure 1.3: Illustration of the graph embedding problem. The goal is to learn an encoder (ENC), which maps nodes to a low-dimensional embedding space. Source: Hamilton (2020).

Moreover, embedding symbolic objects in the Euclidean continuous space allows tackling more complex tasks involving compositionality and non-linear hidden interactions. This is achieved by feeding vectorial data representations as input to (deep) neural networks.

In this thesis, we study methods that allows us to employ non-Euclidean manifolds as target embedding space instead of conventional Euclidean vector spaces ($\mathbb{R}^d$). Furthermore, we generalize graph representation learning to spaces of matrices. We focus our attention in *shallow methods* for graph embeddings, according to the nomenclature of Hamilton (2020) and the taxonomy provided in Chami et al. (2020a). This is, models optimize parameters that are directly used as node embeddings. Hence, we do not cover graph regularization methods, graph auto-encoding methods or neighborhood aggregation methods such as Graph Neural Networks.

Tasks and Applications

We seek to build graph embedding models that can learn from data in order to solve particular tasks. Some of these tasks are node classification, relation prediction, clustering, community detection, and graph classification and regression. Models performing these tasks can be applied to a wide range of applications, such as document (Kipf & Welling, 2017) and protein classification (Hamilton et al., 2017), content recommendation (Ying et al., 2018), knowledge-base completion (Bordes et al., 2013), drug effects prediction (Zitnik et al., 2018), fraud detection (Pandit et al., 2007), healthy food identification (Veselkov et al., 2019) and more. A thorough explanation of these tasks and its applications is provided in Chapter 5.

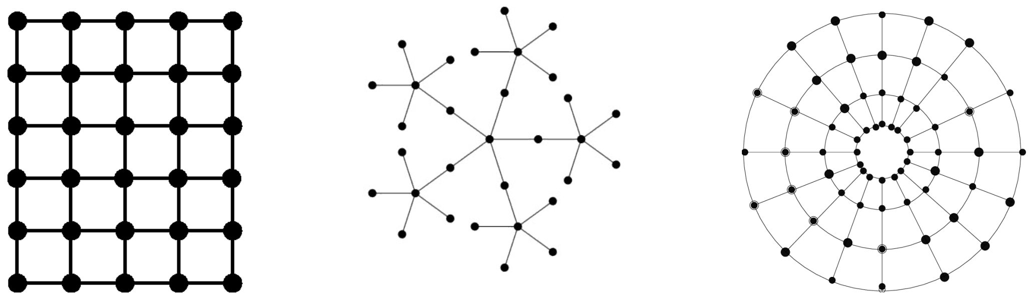

(a) Grid graph. (b) Tree-like graph. (c) Spherical graph.

Figure 1.4: Graphs with different structures. The graphs resemble the Euclidean grid, hyperbolic space and spherical space respectively.

1.1.2 Graphs are non-Euclidean

The predominant approach has been to embed graph-structured data in an Euclidean space. Nonetheless, data in many domains including computer vision (Bronstein et al., 2017), social (Lazer et al., 2009; Verbeek & Suri, 2014), sensor (Gao & Guibas, 2012), gene (Davidson et al., 2002), protein molecular (Gainza et al., 2020) and complex (Krioukov et al., 2010) networks, or the Internet (Boguñá et al., 2010) exhibit non-Euclidean features making embeddings into manifolds with a richer structure necessary.

In this thesis, we explore non-Euclidean geometries as means to introduce a geometric inductive bias. The main goal is to use a continuous embedding space that resembles the underlying discrete structure of the data being embedded (see Figure 1.4). By aligning the target space with the topology of the graph-structured data, we introduce a geometric prior that guides models to achieve higher performance with a reduced parameter footprint.

1.1.3 Transductive and Inductive Settings

Historically, a popular way of categorizing a network embedding method has been by whether the model can generalize to unseen data instances. Methods are referred to as operating in either a *transductive* or *inductive* setting.

In transductive settings, it is assumed that all nodes in the graph are observed in training (typically all the nodes come from one fixed graph). These methods are used to infer information about or between observed nodes in the graph (e.g. predicting labels for all nodes, given a partial labeling). For instance, if a transductive method is used to embed the nodes of a social network, it can be used to suggest new edges (e.g. friendships) between the nodes of the graph. One major limitation of models learned in transductive settings is that they fail to generalize to new nodes (e.g. evolving graphs) or new graph instances.

On the other hand, in inductive settings, models are expected to generalize to new nodes, edges, or graphs that were not observed during training. Formally, given training graphs $(\mathcal{G}_1, ..., \mathcal{G}_k)$, the goal is to learn a mapping to continuous representations that can generalize to unseen test graphs $(\mathcal{G}_{k+1}, ..., \mathcal{G}_{k+l})$. For instance, inductive learning can be used to embed molecular graphs, each representing a molecule structure, generalizing to new graphs and predicting quantum properties (Gilmer et al., 2017). Embedding dynamic or temporally evolving graphs is also another inductive graph embedding problem.

In this thesis, we study graphs in the transductive setting. However, in Chapter 7 we develop a method in which the graph is given only as a set of nodes without edges, and the model learns to place nodes in the space according to their role and co-occurrences in the particular downstream tasks under study. We note that while some models are inherently better suited to different tasks in practice, recent theoretical results by Srinivasan & Ribeiro (2020) show that models previously assumed to be capable of only one setting (e.g. only transductive) can be used in both. Thus, our research can be extended to inductive settings as well.

1.1.4 Supervised and Unsupervised Settings

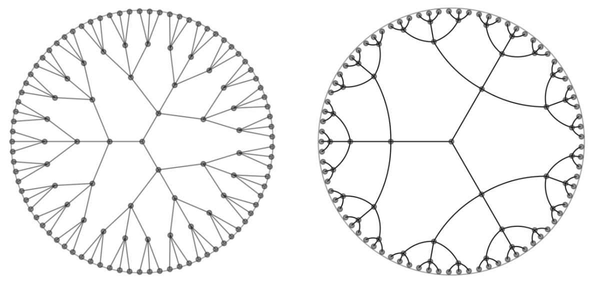
Graph embedding methods can be *unsupervised* in the sense that the only information available is the graph structure (and possibly node features) or *supervised*, if additional information such as node or graph labels is provided. In unsupervised graph embeddings, the goal is to learn embeddings that preserve the graph structure and this is usually achieved by optimizing some reconstruction loss, which measures how well the learned embeddings can approximate the original graph. In supervised graph embeddings, the goal is to learn embeddings for a specific purpose such as predicting node or graph attributes, and models are optimized for a specific task such as graph classification or node classification.

In this work, we study unsupervised methods for graph embeddings, focusing on different reconstruction losses that aim at preserving the graph structure in the embedding space. Nonetheless, in Chapter 7 we work in a supervised setting, learning graph embeddings with a particular purpose defined by a downstream task.

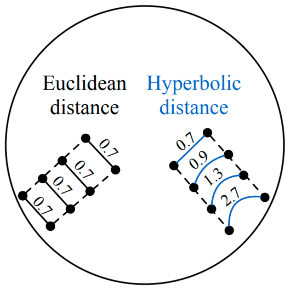
1.1.5 Node Features for Graph Embeddings

Graphs may have node attributes (e.g. gender or age in social networks, article contents for citation networks) which can be represented as multiple functions defined on the graph vertices $f : \mathcal{V} \to \mathbb{R}$ commonly referred to as *node features*. Node features might provide useful information about a graph. Some graph embedding algorithms leverage this information. In other scenarios, node features might be unavailable or not useful for a given task. Hence, graph embedding can be *featureless*.

Note that depending on whether node features are used or not in the embedding algorithm, the learned representation could capture different aspects about the graph. If



(a) Tree embedded in Euclidean and hyperbolic space. Source: Chen et al. (2021a)



(b) Points closer to the center of the space show a smaller hyperbolic distance than points near the boundary. Source: Chen et al. (2021b).

Figure 1.5: Comparison of spaces.

nodes features are being used, embeddings could capture both *structural* and *semantic* graph information. On the other hand, if node features are not being used, embeddings will only preserve *structural* information of the graph. Finally, edge features are less common than node features in practice, but can also be used by embedding algorithms. For instance, edge features can be used as regularization for node embeddings, or to compute messages from neighbors as in message passing networks (Gilmer et al., 2017).

In this thesis we study both cases. We develop *featureless* methods that solely exploit *structural* information. Furthermore, we also propose a way to embed *semantic* information into a non-Euclidean manifold, and by mining the interactions in the space, derive the underlying structure that connects the data points.

1.2 Why non-Euclidean Geometries?

One of the main goals of this thesis is to embed data in non-Euclidean geometries. In this section we provide an intuition of why this approach is advantageous when the structure of the data conforms to the embedding space. To do so we employ trees (a particular type of graph) and the Poincaré ball model of hyperbolic space, but defer the mathematical details of this geometry to Chapter 2. We consider the case of embedding tree structures in an approximately isometric manner. This is, trying to replicate in the space the same distances between nodes than in the graph.

The result of embedding a tree in an Euclidean and hyperbolic space can be observed in Figure 1.5a. Nodes that are high in the tree (such as the root) are placed close to the center of the space, while leaf nodes are placed far from the origin. We observe that the nodes embedded in Euclidean space look more crowded. In particular, leaf nodes from different subtrees are placed very close to each other, generating an undesired overlapping and distorting the metric in the original tree. On the other hand, the hyperbolic space allows sufficient capacity to embed trees. Distances in the space accurately represent distances in

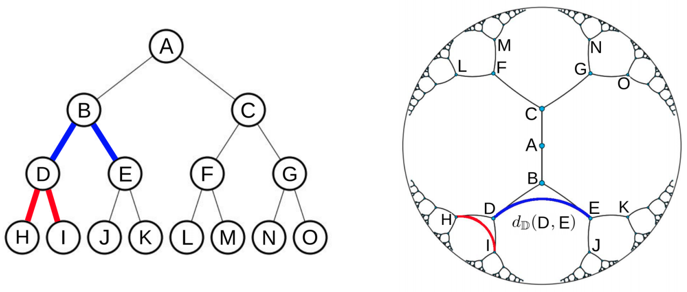


Figure 1.6: Tree embedded in hyperbolic space. Items at the top of the hierarchy are placed near the origin of the space, and lower items near the boundary. Moreover, the hyperbolic distance between sibling nodes resembles the one through the common ancestor, analogous to the distance in the tree. That is $d_{\mathbb{D}}(\mathrm{D}, \mathrm{E}) \approx d_{\mathbb{D}}(\mathrm{D}, \mathrm{B}) + d_{\mathbb{D}}(\mathrm{B}, \mathrm{E})$.

the graph, and this can be better observed in Figure 1.5b and with more detail in Figure 1.6. The reason for this is that while the volume of the Euclidean space grows polynomially with the radius of the ball, in the Poincaré model of hyperbolic space the volume grows exponentially. This is analog to the number of nodes of a full tree growing exponentially with its depth. This phenomenon in which the embedding space grows with a similar rate as the data gives hyperbolic spaces "enough space" to embed trees and hierarchical structures with an arbitrary low distortion (Gromov, 1987; Sala et al., 2018). Intuitively, hyperbolic spaces can be thought of as continuous versions of trees or vice versa, trees can be thought of as "discrete hyperbolic spaces" (Nickel & Kiela, 2017).

Due to the negative curvature and exponentially growing volume, hyperbolic geometry is mathematically suitable to embed hierarchical structures or scale-free networks with heavy tailed degree distributions (Krioukov et al., 2010) that are ubiquitous among real-world graphs (Verbeek & Suri, 2016). For such datasets, this space offers a geometric inductive bias that can lead to improved clustering, interpretability, and generalization properties. Throughout this work, we seek to replicate this behavior with different types of data structures and spaces.

1.3 Research Questions

Representation learning has become an invaluable approach for learning from symbolic data such as text and graphs. It has recently been shown that geometric spaces with constant non-zero curvature improve the quality of the representations for certain data types. One of the most prominent of such cases is the use of hyperbolic space to learn embeddings of datasets that exhibit a hierarchical structure. At the same time, label (or class) inventories for multi-class classification have grown in size and complexity, thus exploiting latent hierarchical information becomes critical to improve performance. While the intrinsic advantages of hyperbolic embeddings to model these structures are well-established, their usefulness in downstream tasks is, so far, less clear. This leads us to our first research question: **How can we derive hierarchical information present in large label inventories, and integrate it in a downstream task through hyperbolic graph embeddings?**

Representing hierarchical data with hyperbolic embeddings also lies at the core of our second research question. In some classification setups, it can be difficult to extract an explicit hierarchical graph. Moreover, the induced hierarchy might not be particularly suited for the final task, or by separately building a graph, and then performing the task, we can ignore crucial information or latent relations present in the data. By learning a task-specific hierarchy, a model could yield different arrangements, tailored for different needs. Therefore, our second research question is: **How can we jointly perform classification and learn task-specific hierarchical representations of the data?**

Previous work has mostly explored Euclidean, hyperbolic, and spherical spaces, or products thereof, to learn graph embeddings. All these spaces are particular instances of a more general class called symmetric spaces. We notice that a consolidated framework in which to include these various examples is still missing. Our third research question aims at tying in previous methods under a consolidated and systematic view. **How can we encompass prior work on graph embeddings in symmetric spaces into a unified framework?**

Finally, in this dissertation we also cover methods for modelling multi-relational graphs. There are efficient models to deal with this type of graphs in Euclidean, complex and hyperbolic spaces. However, they require arithmetic operations specifically designed for those geometries. Furthermore, multi-relational knowledge graphs exhibit an intricate and varying structure as a result of the logical properties of the relationships they encode. Thus, they would profit from learning representations in manifolds with a richer structure. Our third research question is: **How can we translate Euclidean or hyperbolic neural models into more expressive geometries, to represent multi-relational graphs and operate with them?**

1.4 Contributions

With the research presented in this thesis, we make the following contributions:

- To answer the first research question we propose two different techniques for creating hierarchical graphs from large label inventories: from an expert-generated ontology and by automatically mining label co-occurrences. Moreover, we choose a Natural Language Processing task as a test bed, and we pose it as a graph embedding problem, followed by a nearest neighbor classifier in hyperbolic space. This allows us to incorporate hierarchical information through hyperbolic embeddings into a neural model.

- As a solution for the second research question, we introduce a model that jointly learns task-specific graph embeddings from a label inventory and performs multi-class multi-label classification in hyperbolic space. This model achieves a classification performance comparable to state-of-the-art systems on very fine-grained labels with a remarkable reduction of the parameter size. We also show that the graph embeddings automatically infer the latent hierarchy from the class distribution and capture implicit hyponymic relations in the inventory.

- Answering the third research question, we propose SYMPA: a general framework to learn graph embeddings on symmetric spaces. Through our framework, we systematize the use of symmetric spaces in representation learning, a class encompassing many of the previously used embedding targets. These spaces have a compound geometry that simultaneously contains Euclidean as well as hyperbolic or spherical subspaces, allowing them to automatically adapt to dissimilar features in the graph. By employing the SYMPA framework we can choose a Riemannian symmetric space and implement the mathematical tools required to learn graph embeddings. Finally, we demonstrate a concrete implementation of the framework on Siegel spaces, and showcase their versatility for embedding complex graphs without a priori knowledge of their internal structure.

- To address the fourth question we devise the means to translate Euclidean and hyperbolic multi-relational graph embedding models into the space of symmetric positive definite matrices (SPD). To do so, we develop gyrovector calculus, which yields closed-form expressions for several operations in this curved space. In addition, we also apply the SYMPA framework on the SPD manifold to develop algorithms using the vector-valued distance and showcase two main advantages: its versatility to implement more general models, and its use in explaining and visualizing what the model has learned.

1.5 Published Work

Most of the research presented in this thesis is an extension of the published work by the author of this thesis. The hyperbolic nearest neighbor classifier presented in Chapter 6 has been proposed in López et al. (2019). The fully hyperbolic model for hierarchical classification explained in Chapter 7 was introduced in López & Strube (2020). The original idea for the general framework on symmetric spaces for graph embeddings presented in Chapter 8 was laid out in López et al. (2021c), and fully developed in López et al. (2021b).

Finally, the multi-relational graph embedding model developed in Chapter 9 was proposed in López et al. (2021a). The results from many of the baselines models employed in that chapter were collected from López et al. (2021d).

Chapter 2

Non-Euclidean Geometry

"For God's sake, please give it up. Fear it no less than sensual passions, because it, too, may take all your time and deprive you of your health, peace of mind and happiness in life."
– Wolfgang Bolyai urging his son János Bolyai, one of the founders of Hyperbolic geometry, to give up on his work

To define non-Euclidean geometry, we first need to talk about Euclidean geometry. Euclidean geometry is a mathematical system attributed to the Greek mathematician Euclid (*fl.* 300 BC), which he described in his textbook on geometry *"Elements"*, a monumental treatise of mathematics that is likely the most influential book ever written. Euclid builds his geometry as an axiomatic system in which all theorems or "true statements" are derived from a small number of simple postulates or axioms. Euclid gives five postulates for plane geometry:

I. A straight line segment can be drawn joining any two points.

II. Any straight line segment can be extended indefinitely in a straight line.

III. Given any straight line segment, a circle can be drawn having the segment as radius and one endpoint as center.

IV. All Right Angles are congruent.

V. If two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two Right Angles, then the two lines inevitably must intersect each other on that side if extended far enough.

Non-Euclidean geometry arises by either relaxing metric requirements, or replacing the fifth postulate (also known as "the parallel postulate") with an alternative. When the metric requirement is relaxed, then there are affine planes associated with the planar algebras, which give rise to kinematic geometries. In the latter case, by replacing the parallel postulate one obtains hyperbolic geometry and elliptic geometry. The term *non-Euclidean geometry* refers to all the aforementioned cases. However, in this thesis we only focus in hyperbolic and elliptic geometries, and models derived from them.

2.1 Brief History

The debate that eventually led to the discovery of non-Euclidean geometries began almost as soon as Euclid wrote "Elements". As we said, in this book, Euclid begins with a limited number of assumptions (five postulates, five common notions, and 23 definitions) and seeks to prove all the other results (or propositions) in the work. The most notorious of the postulates is often referred to as "Euclid's Fifth Postulate", or simply the parallel postulate, which can equivalently be restated by Playfair's axiom as:

> *"In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point."*[1]

For at least a thousand years, geometers were troubled by the disparate complexity of the parallel postulate, and believed it could be proved as a theorem from the other four. The work of Saccheri on quadrilaterals (see "Saccheri Quadrilaterals" box) led to the first few theorems of the hyperbolic and the elliptic geometries, and they played an important role in the later development of non-Euclidean geometry. All of these early attempts made at trying to formulate non-Euclidean geometry, however, provided flawed proofs of the parallel postulate, containing assumptions that were essentially equivalent to it.

The beginning of the 19th century would finally witness decisive steps in the creation of non-Euclidean geometry. Mathematicians would break Euclid's rule and develop consistent geometries in which the parallel postulate fails. Playfair's version of the postulate offers two ways to negate it: negate the uniqueness of the parallel line, or negate its existence. This can be expressed by the following alternative statements:

a. *"In a plane, given a line and a point not on it,* ***infinitely many*** *lines parallel to the given line can be drawn through the point."*
b. *"In a plane, given a line and a point not on it,* ***no line*** *parallel to the given line can be drawn through the point."*

One can construct new consistent systems of geometry respecting all Euclid's postulates but replacing the fifth for any of these two alternatives (see Figure 2.2).

[1]It is equivalent to Euclid's parallel postulate and was named after the Scottish mathematician John Playfair (1748–1819).

Saccheri Quadrilaterals

A Saccheri quadrilateral is a quadrilateral with two equal sides perpendicular to the base. It is named after Giovanni Gerolamo Saccheri (1667–1733), who used it extensively in an attempt to prove the parallel postulate by Reductio ad absurdum. For a Saccheri quadrilateral ABCD, the sides AD and BC are equal in length, and also perpendicular to the base AB. The top CD is the summit or upper base and the angles at C and D are called the summit angles. The question Saccheri posed was: Are the summit angles right angles, obtuse angles, or acute angles?
As it turns out:

- When the summit angles are right angles, the existence of this quadrilateral is equivalent to Euclid's fifth postulate.
- When the summit angles are obtuse, the quadrilateral leads to elliptical or spherical geometry.
- When the summit angles are acute, this quadrilateral leads to hyperbolic geometry.

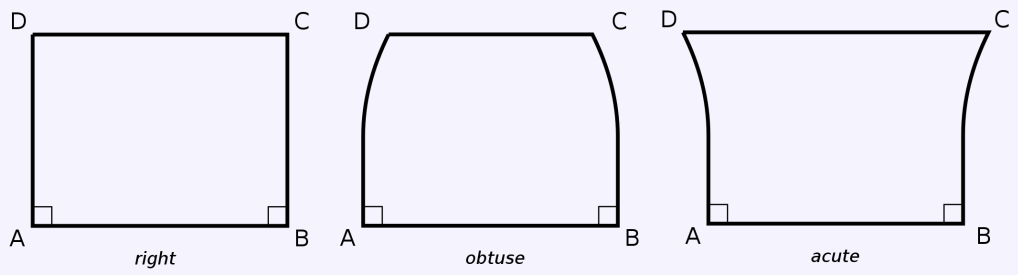


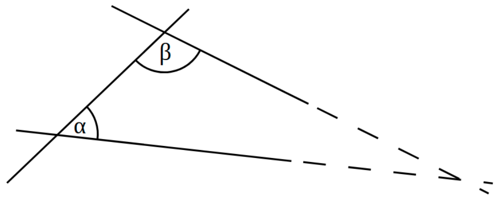
Figure 2.1: Saccheri quadrilaterals. Source: Wikipedia.

In the year 1813, Carl Friedrich Gauss had the germinal ideas of non-Euclidean geometry worked out, but he did not publish any results.[2] In 1830 the Russian mathematician Nikolai Ivanovich Lobachevsky and in 1832 the Hungarian mathematician János Bolyai separately and independently followed the statement a) and published treatises on hyperbolic geometry.[3]

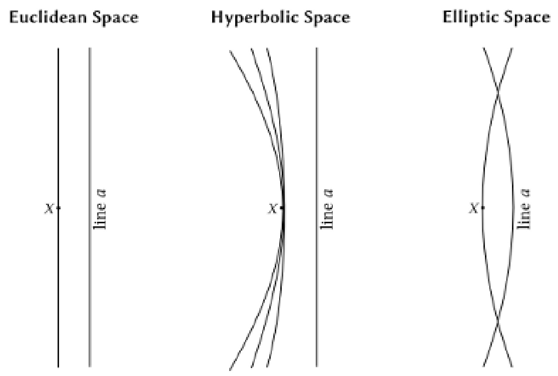
The statement b) was pursued by Bernhard Riemann. In a famous lecture in 1854, he founded the field of *Riemannian geometry*, discussing in particular the ideas now called manifolds, Riemannian metric, and curvature. He constructed an infinite family of non-

[2]In publishing his work, Gauss followed the motto *"Pauca sed matura"* (few, but ripe) which appeared on his seal. Gauss would not publish a result until it was complete and he was entirely satisfied with its presentation. Consequently, much of his work was unpublished with a considerable amount discovered only after his death. Gauss, in defense of his style, said, "A fine building should not show its scaffolding when completed".

[3]Hyperbolic geometry is also called Lobachevskian or Bolyai-Lobachevskian geometry, as both mathematicians, independent of each other, are the basic authors of this geometry.



(a) Euclid's Parallel postulate.



(b) Playfair's axiom and its possible negations

Figure 2.2: Playfair's axiom is equivalent to the Parallel postulate in the Euclidean case. Negating it by allowing infinite parallels gives rise to Hyperbolic geometry. In the case that no parallel line can be drawn, we get Elliptic geometry.

Euclidean geometries by giving a formula for a family of Riemannian metrics on the unit ball in Euclidean space. The simplest of these was the elliptic geometry. A brief overview of some properties of the three geometries is given in Table 2.1 and Figure 2.3.

Before the models of a non-Euclidean plane were presented, Euclidean geometry stood unchallenged as the mathematical model of space. The discovery of the non-Euclidean geometries is an example of a scientific revolution in the history of science It had a ripple effect which went far beyond the boundaries of mathematics and physics, reaching even philosophy or theology, changing the way they viewed their subjects.

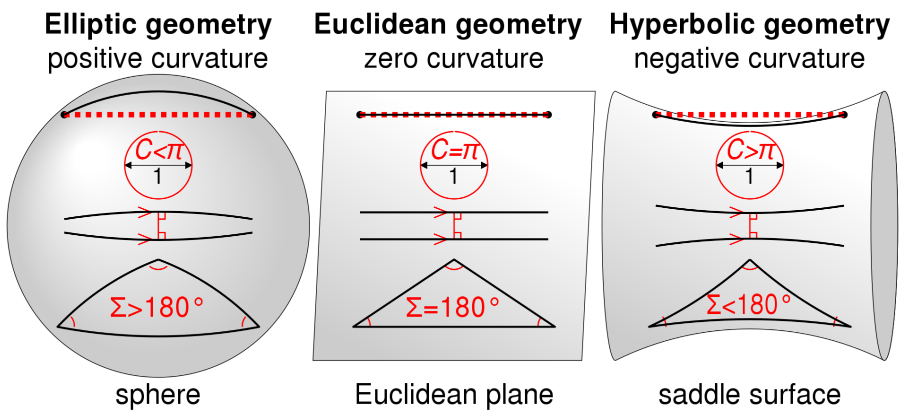


Figure 2.3: Comparison of elliptic, Euclidean and hyperbolic geometries. Source: Wikipedia.

Property	**Euclidean**	**Elliptic**	**Hyperbolic**
Curvature K	0	> 0	< 0
Parallel lines	1	0	∞
Circle length	$2\pi r$	$2\pi \sin(\epsilon\pi)$	$2\pi \sinh(\epsilon\pi)$
Disk area	πr^2	$2\pi(1 - \cos(\epsilon\pi))$	$2\pi(1 - \cosh(\epsilon\pi))$
Sum of triangle angles	π	$> \pi$	$< \pi$

Table 2.1: Some properties of Euclidean, elliptic and hyperbolic geometry. $\epsilon = \sqrt{|K|}$.

2.2 Riemannian Geometry

In this section we recall some definitions from Differential and Riemannian geometry that will be useful for the rest of these thesis. For an in-depth introduction, we refer the reader to Lee (2012).

Definition 2.1 (Manifold)**.** An n-dimensional manifold $\mathcal{M}$ is a topological space that locally resembles the topological space $\mathbb{R}^n$. This is, it is a space that can locally be approximated by $\mathbb{R}^n$. It generalizes the notion of a 2D surface to higher dimensions. For each point x on $\mathcal{M}$, we can find a *homeomorphism* (continuous bijection with continuous inverse) between a neighbourhood of x and $\mathbb{R}^n$.

Definition 2.2 (Tangent space)**.** Intuitively, if we think of $\mathcal{M}$ as a n-dimensional manifold embedded in $\mathbb{R}^{n+1}$, the tangent space $T_x\mathcal{M}$ at point x on M is a n-dimensional hyperplane in $\mathbb{R}^{n+1}$ that best approximates $\mathcal{M}$ around x. Another possible interpretation for $T_x\mathcal{M}$ is that it contains all the possible directions of curves on $\mathcal{M}$ passing through x. The elements of $T_x\mathcal{M}$ are called *tangent vectors* and the union of all tangent spaces is called the *tangent bundle* $T\mathcal{M} = \cup_{x\in\mathcal{M}} T_x\mathcal{M}$

Definition 2.3 (Riemannian metric)**.** A *Riemannian metric* $g = (g_x)_{x\in\mathcal{M}}$ on $\mathcal{M}$ is a collection of inner-products $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ varying smoothly with x on tangent spaces. Riemannian metrics can be used to measure distances on manifolds.

Definition 2.4 (Riemannian manifold)**.** A *Riemannian manifold* is a pair $(\mathcal{M}, g)$, where $\mathcal{M}$ is a smooth manifold and $g = (g_x)_{x\in\mathcal{M}}$ is a Riemannian metric, that is a family of smoothly varying inner products on tangent spaces, $g_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$

Definition 2.5 (Distances and Geodesics)**.** Let $(\mathcal{M}, g)$ be a Riemannian manifold. For $v \in T_x\mathcal{M}$, define the norm of v by $||v||_g := \sqrt{g_x(v, v)}$. Suppose $\gamma : [a, b] \to \mathcal{M}$ is a smooth curve on $\mathcal{M}$. Define the length of γ by:

$$L(\gamma) := \int_a^b ||\gamma'(t)||_g dt$$

Now with this definition of length, every connected Riemannian manifold becomes a metric space and the distance $d : \mathcal{M} \times \mathcal{M} \to [0, \infty)$ is defined as:

$$d(x, y) := \inf_\gamma \{L(\gamma) : \gamma \text{ is a continuously differentiable curve joining x and y}\}$$

Geodesic distances are a generalization of straight lines (or shortest paths) to non-Euclidean geometry. A curve $\gamma : [a, b] \to \mathcal{M}$ is *geodesic* if $d(\gamma(t), \gamma(s)) = L(\gamma|_{[t,s]}) \forall (t, s) \in [a, b](t < s)$.

Definition 2.6 (Parallel transport)**.** *Parallel transport* defined as $P_{x\to y} : T_x\mathcal{M} \to T_y\mathcal{M}$, is a linear isometry between tangent spaces that corresponds to moving tangent vectors along geodesics. Given a smooth manifold $\mathcal{M}$, parallel transport $P_{x\to y}(\cdot)$ maps a vector $v \in T_\mathbf{x}\mathcal{M}$ to $P_{x\to y}(v) \in T_\mathbf{y}\mathcal{M}$. It is a generalization of translation to non-Euclidean geometry, and it defines a canonical way to connect tangent spaces.

Definition 2.7 (Curvature)**.** At a high level, curvature measures how much a geometric object such as surfaces deviate from a flat plane. For instance, the Euclidean space has zero curvature while spheres have positive curvature. We illustrate the concept of curvature in Figure 2.4a.

Definition 2.8 (Exponential and logarithmic maps)**.** The *exponential map* $\exp_x$ at x gives a way to project back a vector v of the tangent space $T_x\mathcal{M}$ at x, to a point $\exp_x(v) \in \mathcal{M}$ on the manifold. This map is often used to parametrize a geodesic γ starting from $\gamma(0) := x \in \mathcal{M}$ with unit-norm direction $\dot{\gamma}(0) := v \in T_x\mathcal{M}$ as $t \mapsto \exp_x(tv)$. Conversely, the *logarithmic map* $\log_x$ at a point x gives a way to project a point y on the manifold $\mathcal{M}$ to a point $\log_x(y)$ on the tangent space $T_x\mathcal{M}$ at x. For geodesically complete manifolds, $\exp_x$ is well-defined on the full tangent space $T_x\mathcal{M}$. A diagram of this operations is showed in Figure 2.4b.

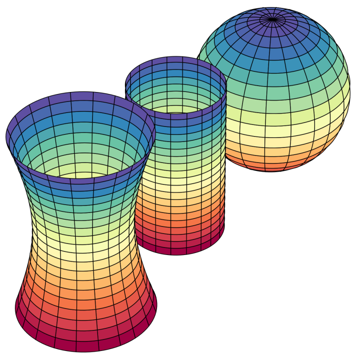
Definition 2.9 (Conformal metric)**.** A metric $\tilde{g}$ is said to be *conformal* to another metric g if it defines the same angles, *i.e.*

$$\frac{\tilde{g}_x(u, v)}{\sqrt{\tilde{g}_x(u, u)}\sqrt{\tilde{g}_x(v, v)}} = \frac{g_x(u, v)}{\sqrt{g_x(u, u)}\sqrt{g_x(v, v)}}$$

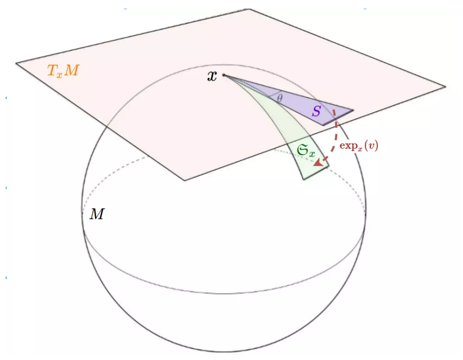
for all $x \in \mathcal{M}, u, v \in T_x\mathcal{M}\backslash\{0\}$. This is equivalent to the existence of a smooth function $\lambda : \mathcal{M} \to \mathbb{R}$, called *conformal factor*, such that $\tilde{g}_x = \lambda_x^2 g_x$ for all $x \in \mathcal{M}$

2.3 Hyperbolic Geometry

The hyperbolic space of dimensions $n \geq 2$ is the unique complete, simply connected Riemannian manifold with constant negative sectional curvature (Cannon et al., 1997).



(a) Curvatures. Source: Wikipedia.



(b) Exponential and Logarithmic maps. Source: Ganea et al. (2018a).

Figure 2.4: a) From left to right: a surface of negative curvature (hyperboloid), a surface of zero curvature (cylinder), and a surface of positive curvature (sphere).

There exist five models of hyperbolic space isometric to each other and conformal to the Euclidean space (see Figure 2.6b). Here we review the Poincaré and Lorentz models of hyperbolic space, as they are well-suited for gradient-based optimization (Nickel & Kiela, 2017, 2018) and offers closed-form expression of geodesics and exponential map (Ganea et al., 2018b).

2.3.1 Poincaré Model

The **Poincaré model** $(\mathbb{D}, g^{\mathbb{D}})$ is defined by the manifold $\mathbb{D}^n = \{x \in \mathbb{R}^n : ||x|| < 1\}$ equipped with the following Riemannian metric

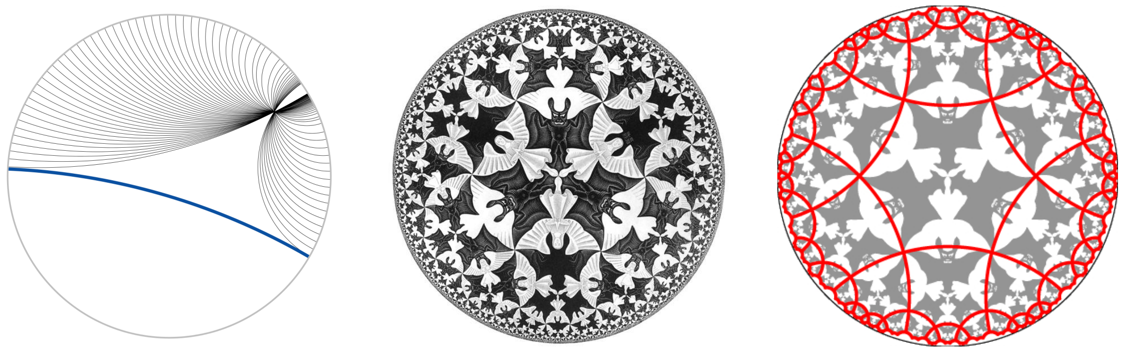
$$g_x^{\mathbb{D}} = \lambda_x^2 g^E, \text{ where } \lambda_x := \frac{2}{1 - ||x||^2} \tag{2.1}$$

and g^E is the Euclidean metric tensor with components $\mathbf{I}_n$ of the standard space $\mathbb{R}^n$ with the usual Cartesian coordinates (see Figure 2.5a).

As the above model is a Riemannian manifold, its metric tensor is fundamental in order to uniquely define most of its geometric properties like distances, inner products (in tangent spaces), straight lines (geodesics), curve lengths or volume elements. In the Poincaré ball model, the Euclidean metric is changed by a simple scalar field, hence the model is conformal (*i.e.* angle preserving), yet it distorts distances.

The **distance** between two points $x, y \in \mathbb{D}^n$ is given by:

$$d_{\mathbb{D}}(x, y) = \cosh^{-1}\left(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)}\right) \tag{2.2}$$



(a) Poincaré disk: All black lines are parallel to the blue one. Source: Wikipedia.

(b) "Circle Limit IV", by M.C. Escher illustrates the Poincaré disc model. Each angel and devil is of constant area in hyperbolic space.

(c) Tessellation {6,4} superimposed on the "Circle Limit IV" pattern. Source: Dunham (1999).

Figure 2.5: In the Poincaré model, straight lines consist of all circular arcs contained within that disk that are orthogonal to the boundary of the disk, plus all diameters of the disk.

The **angle** between two tangent vectors $u, v \in T_x\mathbb{D}$ is given by:

$$\cos(\angle(u,v)) = \frac{g_x^{\mathbb{D}}(u,v)}{\sqrt{g_x^{\mathbb{D}}(u,u)}\sqrt{g_x^{\mathbb{D}}(v,v)}} = \frac{\langle u,v\rangle}{||u||||v||} \tag{2.3}$$

The second equality happens since $g^{\mathbb{D}}$ is conformal to g^E

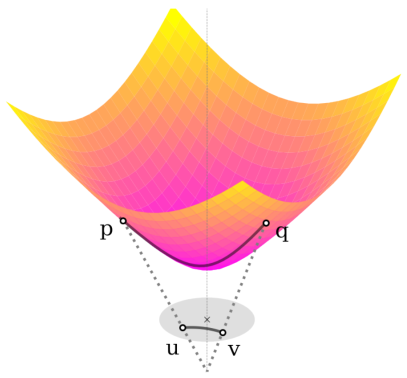
2.3.2 Lorentz Model

The Lorentz (also known as hyperboloid or Minkowski) model is an n-dimensional model of hyperbolic geometry in which points are represented by the points on the forward sheet of a two-sheeted hyperboloid (Figure 2.6). In the following, let $x, y \in \mathbb{R}^{n+1}$ and let:

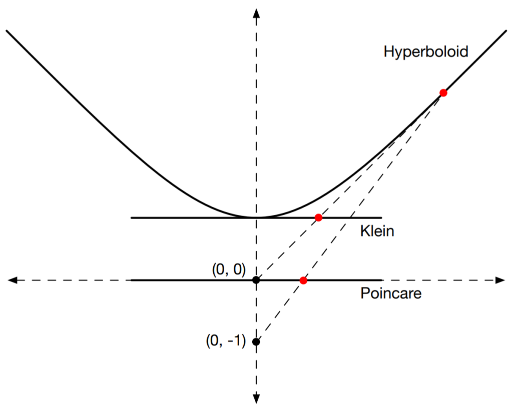
$$\langle x,y\rangle_{\mathbb{L}} = -x_0y_0 + \sum_{i=1}^{n} x_iy_i$$

denote the Lorentzian scalar product. The Lorentz model of n-dimensional hyperbolic space is then defined as the Riemannian manifold $(\mathbb{L}^n, g^{\mathbb{L}})$, where:

$$\mathbb{L}^n = \{x \in \mathbb{R}^{n+1} : \langle x,x\rangle_{\mathbb{L}} = -1, x_0 \geq 0\} \tag{2.4}$$



(a) Lorentz model of hyperbolic geometry. Source: Nickel & Kiela (2018).



(b) Relations between models of hyperbolic space. Source: Gulcehre et al. (2019).

Figure 2.6: Through different projections we can relate the hyperboloid (Lorentz), Klein and Poincaré models of hyperbolic space.

denotes the upper sheet of a two-sheeted n-dimensional hyperboloid and where

$$g^{\mathbb{L}}(x) = \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$$

The associated distance function on $\mathbb{L}$ is then given as:

$$d_{\mathbb{L}}(x, y) = \operatorname{arcosh}(-\langle x, y \rangle_{\mathbb{L}}) \tag{2.5}$$

Furthermore, it holds for any point $x = (x_0, x') \in \mathbb{R}^{n+1}$

$$x \in \mathbb{L}^n \Leftrightarrow x_0 = \sqrt{1 + ||x'||}$$

2.4 Space of Symmetric Positive Definite Matrices

In this section we provide an overview of spaces of symmetric positive-definite matrices, which is another type of Riemannian manifold that we employ in Chapters 9 and 8. More proofs and definitions can be found in Appendix D.

The space SPD_n is a Riemannian manifold of non-positive curvature of $n(n+1)/2$ dimensions. Points in SPD_n are positive definite real symmetric $n \times n$ matrices, with the identity matrix I being a natural basepoint. The tangent space to any point of SPD_n can be identified with the vector space S_n of all real symmetric $n \times n$ matrices. SPD_n contains

n-dimensional Euclidean subspaces, $(n-1)$-dimensional hyperbolic subspaces as well as products of $\lfloor \frac{n}{2} \rfloor$ hyperbolic planes.

Orthogonal diagonalization: Every real symmetric matrix may be orthogonally diagonalized. For every point $P \in \mathrm{SPD}_n$ we may find a positive diagonal matrix D and an orthogonal matrix K such that $P = KDK^T$. This diagonalization has two practical consequences: it allows efficient computation of important SPD_n operations, and provides another means of generalizing Euclidean notions to SPD_n.

Metric and Distance

Metric: The Riemannian metric on SPD_n is defined as follows: if $U, V \in S_n$ are tangent vectors based at $P \in \mathrm{SPD}_n$, their inner product is:

$$\langle U, V\rangle_P = \mathrm{tr}(P^{-1}UP^{-1}V).$$

Note that at the basepoint, this is just the standard matrix inner product $\langle U, V\rangle_I = \mathrm{tr}(UV^T)$ as U, V are symmetric.

Riemannian Distance: This Riemannian metric allows the computation of the length of curves $\gamma\colon [0,1] \to \mathrm{SPD}_n$ which induces a distance function $d : \mathrm{SPD}_n \times \mathrm{SPD}_n \to \mathbb{R}$, by taking the infimum of the lengths of all paths joining two points. While for general Riemannian manifolds such a distance function may be impossible to explicitly compute, the symmetries of SPD_n provide a readily computable formula.

Proposition 2.1. The Riemannian distance between two arbitrary points $P, Q \in \mathrm{SPD}_n$ is given by

$$d(P,Q) = \sqrt{\sum_{i=0}^{n} \log(\lambda_i(P^{-1}Q))} \tag{2.6}$$

where $\{\lambda_i(P^{-1}Q)\}$ are the eigenvalues of of $P^{-1}Q$.

This and the following propositions are proved in Appendix D.

Exponential and Logarithmic Maps

The Riemannian exponential map gives a connection between the Euclidean geometry of the tangent space S_n and the curved geometry of SPD_n. It assigns the tangent vector U to the point $Q = \exp(U)$ of SPD_n reached by traveling along the geodesic starting from the basepoint I in direction U for distance $\|U\|$.

As a consequence of non-positive curvature, $\exp$ is a diffeomorphism of S_n onto SPD_n, and so has an inverse: the Riemannian logarithm $\log\colon \mathrm{SPD}_n \to S_n$. See Ballmann et al. (1985) for a review of the general theory of manifolds of non-positive curvature. Together, this pair of functions allows one to freely move between between 'tangent space coordinates' or the original 'manifold coordinates'.

Secondly, the geometry of SPD_n is so tightly tied to the algebra of $n \times n$ matrices that the Riemannian exponential agrees exactly with the usual matrix exponential, and the Riemannian logarithm is the matrix logarithm (because of this, we do not distinguish the two notationally), as we verify in the proposition below. Both of these are readily computable via orthogonal diagonalization. This is in stark contrast to general Riemannian manifolds, where the exponential map may have no simple formula.

Proposition 2.2. Let $\exp_{\text{Riem}} \colon S_n \to \mathrm{SPD}_n$ be the Riemannian exponential map based at $I \in \mathrm{SPD}_n$, and $\exp$ be the matrix exponential. Then $\exp_{\text{Riem}} = \exp$.

Proposition 2.3. Let $\log_{\text{Riem}} \colon \mathrm{SPD}_n \to S_n$ be the Riemannian logarithm map based at $0 \in S_n$, and $\log$ be the matrix logarithm (note that while the matrix logaritm is multivalued in general, it is uniquely defined on S_n). Then $\log_{\text{Riem}} = \log$.

2.5 Symmetric Spaces, Distances, and Metrics

In this section we provide a brief overview of symmetric spaces. Although all spaces studied throughout this work are particular cases of symmetric spaces, they will be of special interest in Chapter 8. Furthermore, we also introduce the vector-valued distance and Finsler metrics, which are employed in chapters 8 and 9.

2.5.1 Symmetric Spaces

Riemannian symmetric spaces have been extensively studied by mathematicians, and there are many ways to characterize them. They can be described as simply connected Riemannian manifolds, for which the curvature is covariantly constant, or Riemannian manifolds, for which the geodesic reflection in each point defines a global isometry of the space. A key consequence is that symmetric spaces are homogeneous manifolds, which means in particular that the neighbourhood of any point in the space looks the same, and moreover that they can be efficiently described by the theory of semisimple Lie groups.

To be more precise a *symmetric space* is a Riemannian manifold (M, g) such that for any point $p \in M$, the geodesic reflection at p is induced by a global isometry of M. A direct consequence is that the group of isometries $\mathrm{Isom}(M, g)$ acts transitively on M, i.e. given $p, q \in M$ there exists $g \in \mathrm{Isom}(M, g)$ such that $g(p) = q$. Thus symmetric spaces are homogeneous manifolds, which means in particular that the neighbourhood of any point in the space looks the same. This leads to an efficient description by the theory of semisimple Lie groups: $M = \mathsf{G}/\mathsf{K}$ where $\mathsf{G} = \mathrm{Isom}_0(\mathsf{M})$ and K, a compact Lie group, is the stabilizer of a point $p \in M$.

Classification

Every symmetric space (M, g) can be decomposed into an (almost) product $M = M_1 \times \cdots \times M_k$ of symmetric spaces. A symmetric space is *irreducible*, if it cannot be further decomposed into a Riemannian product $M = M_1 \times M_2$. We restrict our discussion to these fundamental building blocks, the irreducible symmetric spaces.

Irreducible symmetric spaces can be distinguished in two classes, the symmetric spaces of compact type, and the symmetric spaces of non-compact type, with an interesting **duality** between them. Apart from twelve exceptional examples, there are eleven infinite families of pairs of symmetric spaces X of compact and non-compact type (they are summarized in Chapter 8, Table 8.1). We refer the reader to Helgason (1978) for more details and a list of the exceptional examples.

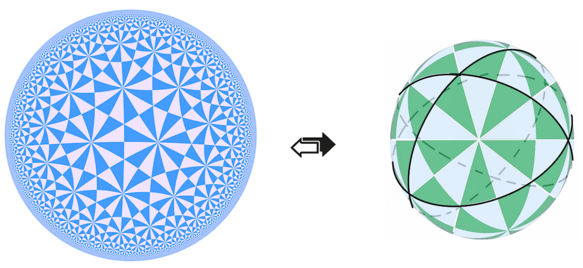
Rank: An important invariant of a symmetric space M is its rank, which is the maximal dimension of an (isometrically embedded) Euclidean submanifold. In a rank r non-compact symmetric space, such submanifolds are isometric to $\mathbb{R}^n$, and called *maximal flats*. In a compact symmetric space, they are compact Euclidean manifolds such as tori.

Some of the rich symmetry of symmetric spaces is visible in the distribution of flats. As homogeneous spaces, each point of a symmetric space M must lie in *some* maximal flat, but in fact for every pair p, q of points in M, one may find some maximal flat containing them. The ability to move any pair of points into a fixed maximal flat by symmetries renders many quantities (such as the metric distances described below) computationally feasible.

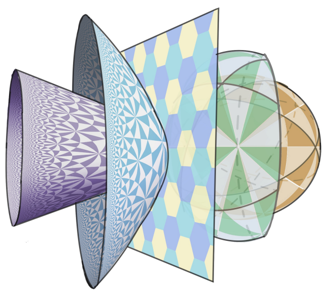
Duality

Compactness provides a useful dichotomy for irreducible symmetric spaces. Symmetric spaces of compact type are compact and of non-negative sectional curvature. The basic example being the sphere S^n. Symmetric spaces of non-compact type are non-compact, in fact they are homeomorphic to $\mathbb{R}^n$ and of non-positive sectional curvature. The basic example being the hyperbolic spaces $\mathbb{H}^n$.

There is a duality between the symmetric spaces of non-compact type and those of compact type, pairing every noncompact symmetric space with its compact 'partner' or dual (see Figure 2.7). Duality for symmetric spaces generalizes the relationship between spheres and hyperbolic spaces, as well as between classical and hyperbolic trigonometric functions. In the reference Table 8.1 (in Chapter 8), we provide for each family of symmetric spaces



(a) The duality between the hyperbolic plane and sphere is the basic example of the duality between symmetric spaces of compact and noncompact type.



(b) Interpolating between the Siegel space (a type of symmetric space) and its compact dual

Figure 2.7: Duality of symmetric spaces. Right: A 1-parameter family of spaces interpolating between the Siegel space and its compact dual, here illustrated in rank 1 ($\mathbb{H}^2$ transitioning S^2 through the Euclidean plane with $k = 0$).

an explicit realization of both the noncompact symmetric space and its compact dual as coset spaces G/K.

2.5.2 Vector-valued Distance

The familiar geometric invariant of pairs of points is simply the distance between them, represented by a scalar. For rank n symmetric spaces, this one dimensional invariant is superseded by an n-dimensional invariant: the *vector valued distance*.

In Euclidean or hyperbolic spaces, given two points A, B at distance k from each other, and two other points C, D also at distance k from each other, there exists an isometry of the space, i.e. a transformation preserving all geometric quantities, that will map A to C and B to D. In other words, the relative position between two points in Euclidean or hyperbolic space is completely determined by a number, namely their distance.

For rank n symmetric spaces this is not true, since there exists pairs of points A, B at distance k from each other, and C, D also at distance k from each other, such that there is no isometry mapping A to C and B to D. In these symmetric spaces the relative position between two points is determined by a vector, which we refer to as the vector-valued distance (VVD). Only if the VVD between two points A and B is the vector v, and the VVD between C and D is also v, then there exists an isometry mapping A to C and B to D. The formulas and procedures to compute the vector-valued distance on general symmetric spaces, is detailed in Appendix B. For Siegel Spaces in particular, it is outlined in Chapter 8, and for SPD manifolds in Chapter 9. For a more in-depth review of the vector-valued distance in symmetric spaces see Kapovich et al. (2017) §2.6.

This vector is an invariant of the relative position of two points up to isometry. This means that we can recover completely the relative position of two points in a rank n symmetric space from this vector, and any pair of points can be mapped to each other by an isometry if and only if they have the same vector-valued distance. Thus, the VVD contains much more information than just the distance. Out of the VVD between two points, one can immediately read the regularity of the unique geodesics joining these two points. This is not possible knowing just the distance as a scalar. Moreover, the VVD contains the full information of the Riemannian distance and of all invariant Finsler distances.

2.5.3 Finsler Metrics

The Riemannian distance function on a manifold is completely determined by its Riemannian metric, a choice of inner product on the tangent bundle. Generalizing this, Finsler metrics are the class of distance functions which may be constructed from a smoothly varying choice of norm $\|\cdot\|_F$ on the tangent bundle (which need not be induced by an inner product). The basic theory proceeds in direct analogy to the Riemannian case. The length of a curve γ is defined via integration of this norm along the path:

$$\mathrm{Length}_F(\gamma) = \int_I \|\gamma'\|_F dt,$$

and the distance between points by the infimum of this over all rectifiable curves joining them:

$$d_F(p,q) = \inf\{\mathrm{Length}_F(\gamma) \mid \gamma(0) = p,\ \gamma(1) = q\}$$

The geometry of symmetric spaces allows the computation of Finsler distances, like much else, to take place in a chosen maximal flat. On such flat spaces, the ability to identify all tangent spaces allow particularly simple Finsler metrics to be defined by choosing a single norm on $\mathbb{R}^n$.

Finsler metrics allow us to generalize other notions of distances, such as the ℓ^1 or ℓ^∞ metrics. While affine lines are geodesics in Finsler geometry, they need not be the unique geodesics between a pair of points. Consider the Figure 2.8 where we plot the shortest path between two points in ℓ^1 geometry. The distance minimizing geodesics connecting the points are not unique, unlike the unique diagonal connecting both points, which would be the shortest path using Riemannian distance. That is, in ℓ^1 geometry different paths traveling along the union of a vertical and horizontal side of a square are distance minimizing paths of the same length. The ℓ^1 metric is often called the 'taxicab' metric for this reason: much as in a city with a grid layout of streets, there are many shortest paths between a generic pair of points, as you may break your path into different choices of horizontal and vertical segments without changing its length.

Additional information about Finsler metrics for symmetric spaces can be found in Planche (1995), and also in Appendix B. They will be of particular interest to us in Chapters 8 and 9.

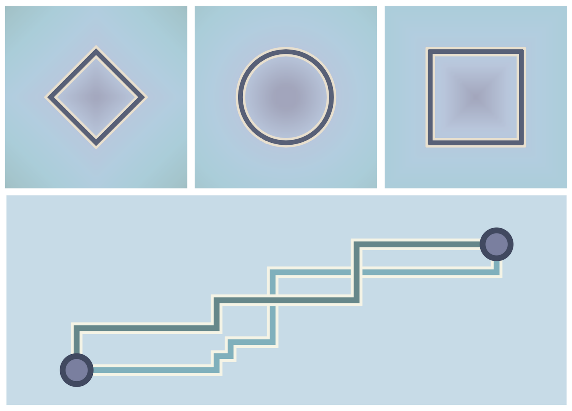


Figure 2.8: Above, from left to right: the unit spheres for the ℓ^1, ℓ^2 (Euclidean), and ℓ^∞ metrics on the plane. Below: Distance minimizing geodesics are not necessarily unique in Finsler geometry. The two paths shown have the same (minimal) ℓ^1 length.

2.6 Gyrovector Spaces

In the Euclidean space, natural operations inherited from the vectorial structure, such as vector addition, subtraction and scalar multiplication are often useful. The framework of gyrovector spaces provides an elegant non-associative algebraic formalism for hyperbolic geometry just as vector spaces provide the algebraic setting for Euclidean geometry (Ungar, 2005, 2008b,a). In particular, these operations are used in special relativity, allowing to add speed vectors belonging to the Poincaré ball of radius c (the celerity, *i.e.* the speed of light) so that they remain in the ball, hence not exceeding the speed of light (Ganea et al., 2018b).

In this section we provide some general notions about the formalism of gyrovector spaces that will be useful in development of operations on SPD in Chapter 9. After that, we describe the implementation of Gyrovector spaces to hyperbolic geometry, together with the work of Ganea et al. (2018b) which we build upon in Chapter 7.

We only provide the general definitions of *gyrogroups* and *gyrovector spaces*. More proofs and definitions can be found in Appendix A. For an in-depth description see Ungar (2008a).

Definition 2.10 (Gyrogroups)**.** A groupoid $(G, \oplus)$ is a gyrogroup if its binary operation satisfies the following axioms.

1. In G there is at least one element, $\mathbf{0}$, called a left identity, satisfying $\mathbf{0} \oplus a = a$ for all $a \in G$
2. There is an element $\mathbf{0} \in G$ satisfying axiom 1 such that for each $a \in G$ there is an element $\ominus a \in G$, called a left inverse of a, satisfying $\ominus a \oplus a = \mathbf{0}$
3. Moreover, for any $a, b, c \in G$ there exists a unique element $\mathrm{gyr}[a, b]c \in G$ such that

the binary operation obeys the left gyroassociative law: $a \oplus (b \oplus c) = (a \oplus b) \oplus \mathrm{gyr}[a, b]c$.

4. The map $\mathrm{gyr}[a, b] : G \to G$ given by $c \mapsto \mathrm{gyr}[a, b]c$ is an automorphism of the groupoid $(G, \oplus)$, that is $\mathrm{gyr}[a, b] \in Aut(G, \oplus)$, and the automorphism $\mathrm{gyr}[a, b]$ of G is called the gyroautomorphism, or the gyration, of G generated by $a, b \in G$. The operator $\mathrm{gyr} : G \times G \to Aut(G, \oplus)$ is called the gyrator of G.

5. Finally, the gyroautomorphism $\mathrm{gyr}[a, b]$ generated by any $a, b \in G$ possesses the left loop property $\mathrm{gyr}[a, b] = \mathrm{gyr}[a \oplus b, b]$.

Definition 2.11 (Real Inner Product Gyrovector Spaces)**.** A real inner product gyrovector space $(G, \oplus, \otimes)$ (gyrovector space, in short) is a gyrocommutative gyrogroup $(G, \oplus)$ that obeys the following axioms:

1. G is a subset of a real inner product vector space $\mathbb{V}$ called the carrier of G, $G \subset \mathbb{V}$, from which it inherits its inner product, $\cdot$, and norm, $|| \cdot ||$, which are invariant under gyroautomorphisms, that is: $\mathrm{gyr}[u, v]a \cdot \mathrm{gyr}[u, v]b = a \cdot b$ for all points $a, b, u, v \in G$

2. G admits a scalar multiplication, $\otimes$, possessing the following properties. For all real numbers $r, r_1, r_2 \in \mathbb{R}$ and all points $a \in G$:

   (a) $1 \otimes a = a$, Identity Scalar Multiplication

   (b) $(r_1 + r_2) \otimes a = r_1 \otimes a \oplus r_2 \otimes a$, Scalar Distributive Law

   (c) $(r_1 r_2) \otimes a = r_1 \otimes (r_2 \otimes a)$ Scalar Associative Law

   (d) $\frac{|r| \otimes a}{||r \otimes a||} = \frac{a}{||a||}, a \neq 0, r \neq 0$ Scaling Property

   (e) $\mathrm{gyr}[u, v](r \otimes a) = r \otimes \mathrm{gyr}[u, v]a$ Gyroautomorphism Property

   (f) $\mathrm{gyr}[r_1 \otimes v, r_2 \otimes v] = I$ Identity (Trivial) Gyroautomorphism.

3. Real, one-dimensional vector space structure $(||G||, \oplus, \otimes)$ for the set $||G||$ of one-dimensional “vectors”

   (a) $||G|| = \{\pm||a|| : a \in G\} \subset \mathbb{R}$, Vector Space, with vector addition $\oplus$ and scalar multiplication $\otimes$, such that for all $r \in \mathbb{R}$ and $a, b \in G$,

   (b) $||r \otimes a|| = |r| \otimes ||a||$ Homogeneity Property

   (c) $||a \oplus b|| \leq ||a|| \oplus ||b||$ Gyrotriangle Inequality.

2.6.1 Gyrovector Spaces for Hyperbolic Geometry

To implement the framework of gyrogroups and gyrovector spaces in different geometries we need to provide concrete implementations of the operations. Here we describe these operations for the Poincaré model of hyperbolic space.

Möbius addition: It is the hyperbolic analogous to vector addition in Euclidean space. Given two points $x, y \in \mathbb{D}^n$, it is defined as:

$$x \oplus y = \frac{(1 + 2\langle x, y\rangle + \|y\|^2)x + (1 - \|x\|^2)y}{1 + 2\langle x, y\rangle + \|x\|^2\|y\|^2} \tag{2.7}$$

Note that this operation is neither commutative nor associative. The Möbius substraction is then defined by the use of the following notation: $x \ominus y := x \oplus (-y)$. See Vermeer (2005) for a geometric interpretation of the Möbius addition.

Möbius scalar multiplication: for $x \in \mathbb{D}^n \backslash \{0\}$ the Möbius scalar multiplication by $r \in \mathbb{R}$ is defined as:

$$r \otimes x = \tanh(r \tanh^{-1}(\|x\|))\frac{x}{\|x\|} \tag{2.8}$$

and $r \otimes 0 := 0$.

Distance: We can express the distance function with this operations. For two points $x, y \in \mathbb{D}^n$, it is defined as:

$$d(x, y) = 2\tanh^{-1}(-x \oplus y) \tag{2.9}$$

Exponential and Logarithmic map: We know from the Hopf-Rinow theorem that the hyperbolic space is complete as a metric space (Cannon et al., 1997). This guarantees that $\mathbb{D}^n$ is geodesically complete. Thus, the **exponential map** is defined for each point $x \in \mathbb{D}^n$ and any $v \in \mathbb{R}^n (= T_x\mathbb{D}^n)$. The mapping between the tangent space and hyperbolic space is done by the exponential map $\exp_x : T_x\mathbb{D}^n \to \mathbb{D}^n$ and the logarithmic map $\log_x : \mathbb{D}^n \to T_x\mathbb{D}^n$. They are given for $v \in T_x\mathbb{D}^n \backslash \{0\}$ and $y \in \mathbb{D}^n \backslash \{0\}, y \neq x$:

$$\begin{aligned} \exp_x(v) &= x \oplus \left(\tanh\left(\frac{\lambda_x\|v\|}{2}\right)\frac{v}{\|v\|}\right) \\ \log_x(y) &= \frac{2}{\lambda_x}\tanh^{-1}(\|-x \oplus y\|)\frac{-x \oplus y}{\|-x \oplus y\|} \end{aligned} \tag{2.10}$$

These expressions become more appealing when $x = 0$, that is, at the origin of the space. They are given for $v \in T_{\mathbf{0}}\mathbb{D}^n \backslash \{0\}$ and $y \in \mathbb{D}^n \backslash \{0\}$:

$$\begin{aligned} \exp_{\mathbf{0}}(v) &= \tanh(\|v\|)\frac{v}{\|v\|} \\ \log_{\mathbf{0}}(y) &= \operatorname{arctanh}(\|y\|)\frac{y}{\|y\|} \end{aligned} \tag{2.11}$$

Chapter 3

Graphs

"It all begins with something very simple and very structureless. We can think of it as a collection of abstract relations between abstract elements. Or we can think of it as a graph."
– Stephen Wolfram
'A Project to Find the Fundamental Theory of Physics'

A paper written by Leonhard Euler on the Seven Bridges of Königsberg and published in 1736 is regarded as the first paper in the history of graph theory (Euler, 1736). In Euler's investigation of the bridges of Königsberg, he discovered that it was the general arrangement of features that was important, not their exact locations. This observation laid the foundations of graph theory and prefigured the idea of topology (see Figure 3.1).

The power of the graph formalism lies both in its focus on relationships between points (rather than the properties of individual points), as well as in its generality. The same graph formalism can be used to represent social networks, interactions between drugs and proteins, the interactions between atoms in a molecule, or the connections between terminals in a telecommunications network, to name just a few examples.

3.1 Definitions from Graph Theory

Here we introduce definitions from graph theory that are used throughout this thesis and complement the ones provided in Section 1.1.

Definition 3.1 (Path)**.** A path P is a sequence of edges $(u_{i1}, u_{i2}), (u_{i2}, u_{i3}), ..., (u_{ik}, u_{ik+1})$ of length k. A path is called simple if all u_{ij} are distinct from each other. Otherwise, if a path visits a node more than once, it is said to contain a cycle.

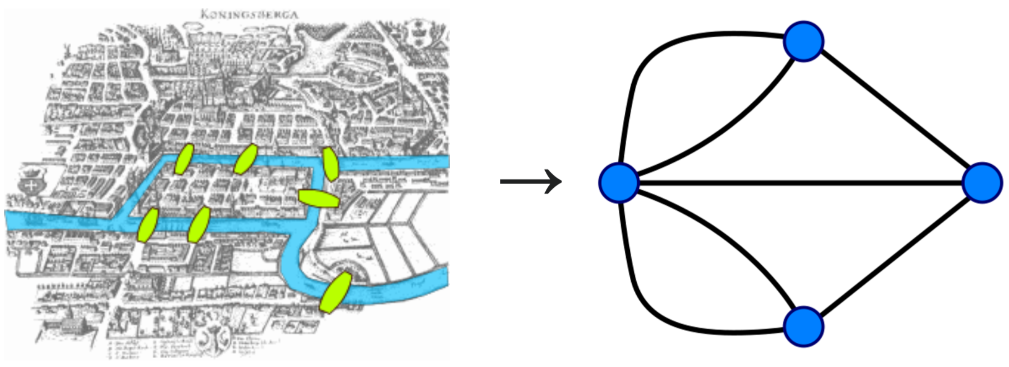


Figure 3.1: Map of Königsberg in Euler's time showing the actual layout of the seven bridges, and the graph induced from it. Source: Wikipedia.

Definition 3.2 (Distance)**.** Given two nodes (u, v) in a graph $\mathcal{G}$, we define the distance from u to v, denoted $d_{\mathcal{G}}(u, v)$, to be the length of the shortest path from u to v, or ∞ if there exist no path from u to v.

Definition 3.3 (Vertex Degree)**.** The *degree*, $\deg(v_i)$, of a vertex v_i in an unweighted graph is the number of edges incident to it. Similarly, the degree of a vertex v_i in a weighted graph is the sum of incident edges weights.

Definition 3.4 (Adjacency Matrix)**.** A finite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be represented as a square $|\mathcal{V}| \times |\mathcal{V}|$ adjacency matrix, where the elements of the matrix indicate whether pairs of nodes are adjacent or not. The adjacency matrix is binary for unweighted graph, $A \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$, and non-binary for weighted graphs $W \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$.

Definition 3.5 (Tree)**.** A *tree* is a particular type of undirected graph in which any two vertices are connected by exactly one path.

Definition 3.6 (Laplacian)**.** The unnormalized *Laplacian* of an undirected graph is the $|\mathcal{V}| \times |\mathcal{V}|$ matrix $L = D - W$. The symmetric normalized *Laplacian* is $\tilde{L} = I - D^{-1/2} W D^{-1/2}$.

3.2 Measures for Graph Analysis

The goal of this thesis is to improve the representation capacity of models by using continuous embedding spaces that resemble the underlying graph-structured data. Hence, we first need to characterize the topology of the data to facilitate choosing an appropriate Riemannian manifold to embed discrete data into. In this section, we present a toolkit formed by different metrics that we apply to study topological properties of graphs.

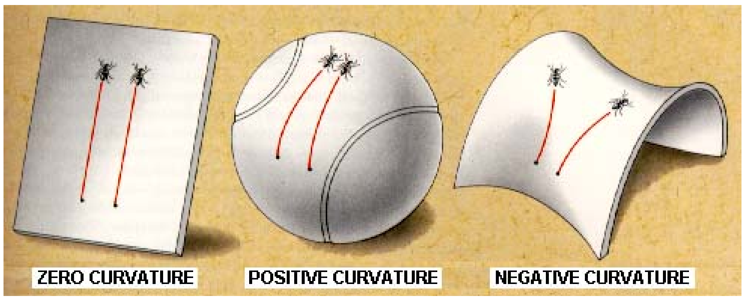


Figure 3.2: Bugs walking on parallel paths on surfaces with different curvature. Source: Wikipedia.

3.2.1 Curvature Analysis

Curvature is a geometric property that describes the local shape of an object. If we draw two parallel paths on a surface with positive curvature like a sphere, these two paths move closer to each other while for a negatively curved surface like a saddle, these two paths tend to be apart (see Figure 3.2). There are multiple notions of curvature in Riemannian manifolds, with varying granularity (for an in-depth treatment see Lee (1997)). We only recall a key notion: hyperbolic spaces have constant negative curvature, Euclidean spaces have zero curvature (flat) and spherical spaces are positively curved.

Discrete data such as graphs do not have manifold structure. Thus, curvature analogs are necessary to provide a measure that satisfies similar properties (Cruceru et al., 2020). In the following we describe two types of curvature analogs that we will use to characterize topological properties of the graphs.

Sectional Curvature

The sectional curvature of a manifold gives a fine-grained notion defined over all two-dimensional subspaces passing through a point p. Intuitively, this captures the rate that geodesics on the surface emanating from p spread apart, which relates to volume growth.

Gu et al. (2019) provides a way to estimate an analog to the sectional curvature for graphs. The estimation technique employs a triangle comparison theorem following from Toponogov's theorem and the law of cosines, which characterizes sectional curvature through the behavior of small triangles (note that a triangle determines a 2-dimensional submanifold). Let abc be a geodesic triangle in manifold (or metric space) $\mathcal{M}$ and m be the (geodesic) midpoint of bc, and consider the quantity:

$$\xi_{\mathcal{M}}(a,b,c) := d_{\mathcal{M}}(a,m)^2 + d_{\mathcal{M}}(b,c)^2/4 - (d_{\mathcal{M}}(a,b)^2 + d_{\mathcal{M}}(a,c)^2)/2 \tag{3.1}$$

This is non-negative (resp. non-positive) when the curvature is non-negative (resp. non-positive), and the equality case occurs when the curvature is 0 (see Figure 3.3)

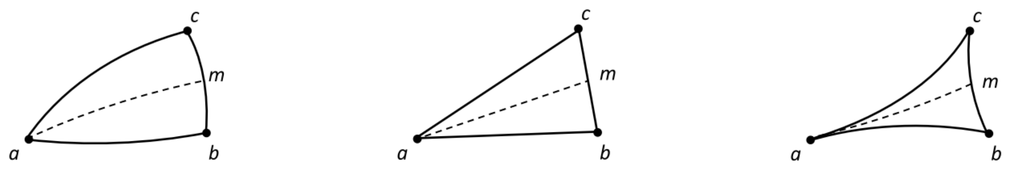


Figure 3.3: Geodesic triangles in differently curved spaces: compared to Euclidean geometry in which it satisfies the parallelogram law (center), the median am is longer in cycle-like positively curved space (left), and shorter in tree-like negatively curved space (right). The relative length of am can be used as a heuristic to estimate discrete curvature. Source: Gu et al. (2019).

Analogous to sectional curvature, which is a function of a point p and two directions x, y from p, in an undirected graph $\mathcal{G}$ we define an analog for every node m and two neighbors b, c. Given a reference node a we set: $\xi_{\mathcal{G}}(m; b, c; a) = \frac{1}{2d_{\mathcal{G}}(a,m)} \xi_{\mathcal{G}}(a, b, c)$. This is exactly the expression from equation 3.1, normalized suitably so as to yield the correct scaling for trees and cycles. The curvature estimation is then a simple average $\xi_{\mathcal{G}}(m; b, c) = \frac{1}{|\mathcal{V}|-1} \sum_{a \neq m} \xi_{\mathcal{G}}(m; b, c; a)$. $\xi_{\mathcal{G}}$ recovers the right curvature for graph atoms: the curvature is zero for lines, positive for cycles, and negative for trees.

Ollivier-Ricci Curvature

The Ricci curvature (Ricci, 1904) of a tangent vector v at p is the average of the sectional curvature over all planes U containing v. Geometrically the Ricci curvature measures how much the volume of a small cone around direction v compares to the corresponding Euclidean cone. Positive curvature implies smaller volumes, and negative implies larger.

Ollivier (2009) generalized the Ricci curvature to metric spaces $(\mathcal{M}, d_{\mathcal{M}})$. It is defined in a way that mimics the interpretation of Ricci curvature on Riemannian manifolds: it is the average distance between two small balls taken relative to the distance between their centers. It yields the curvatures one would expect in several cases: negative curvatures for trees (except for the edges connecting the leaves) and positive for complete graphs and hypercubes. However, it does not capture the curvature of a cycle with more than 5 nodes because it locally looks like a straight line. Since the Ollivier-Ricci curvature characterizes the space only locally, we plot the results over diagrams of the graphs.

3.2.2 δ-hyperbolicity

Also known as Gromov hyperbolicity (Gromov, 1987), δ-hyperbolicity quantifies with a single number the hyperbolicity of a given metric space. The smaller the δ is, the more hyperbolic-like or negatively-curved the space is. The definition that makes it easier to picture it is via the slim triangles property: a metric a δ-slim triangle if any point on the geodesic segment between any two of them is within distance δ space $(\mathcal{M}, d_{\mathcal{M}})$ is

δ-hyperbolic if all geodesic triangles are δ-slim. Three points $x, y, w \in \mathcal{M}$ form from the other two geodesics (i.e., "sides" of the geodesic triangle).

For discrete metric spaces such as graphs, an equivalent definition using the so-called "4-points condition" can be used to devise algorithms that look at quadruples of points. Both exact and approximating algorithms exist that run fast enough to analyze graphs with tens of thousands of nodes within minutes (Fournier et al., 2015; Cohen et al., 2015).

Chapter 4

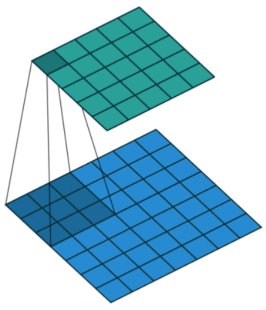
Geometric Deep Learning

"Mighty is geometry; joined with art, resistless."
– Euripides

Deep learning aims to learn complicated concepts by building them from simpler ones in a hierarchical or multi-layer manner. One of the key reasons for the success of deep neural networks is their ability to leverage statistical properties of the data such as stationarity and compositionality through local statistics, which are present in natural images, video, and speech (Simoncelli & Olshausen, 2001). For instance, in images, each pixel has the same neighborhood structure, thus one can consider images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift-invariance, locality is due to the local connectivity, and compositionality stems from the multi-resolution structure of the grid. These properties are exploited by convolutional architectures (CNN, LeCun et al. (1989)).

However, in graphs, one can not define an ordering of nodes since each node might have a different neighborhood structure (see Figure 4.1). Furthermore, Euclidean convolutions strongly rely on geometric priors (e.g. shift invariance) which do not generalize to non-Euclidean domains (e.g. translations might not even be defined on non-Euclidean domains). These challenges led to the development of Geometric deep learning research. *Geometric deep learning* is an umbrella term for emerging techniques attempting to generalize structured deep neural models to non-Euclidean domains. In particular, given the widespread prevalence of graphs in real-world applications, there has been a surge of interest in applying machine learning methods to graph-structured data.

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems, the goal is to *characterize the structure* of the


(a) Blue: Original image defined as a grid. Green: Result of applying a CNN layer.

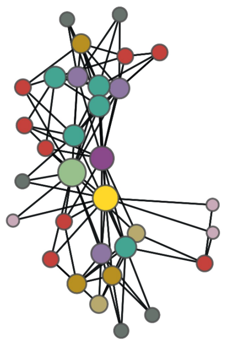

(b) Arbitrary graph.

Figure 4.1: Examples of image processing as a grid (Euclidean) vs. an arbitrary graph (non-Euclidean).

data. The second class of problems deals with *analyzing functions* defined on a given non-Euclidean domain. These two classes are related, since understanding the properties of functions defined on a domain conveys certain information about the domain, and vice-versa, the structure of the domain imposes certain properties on the functions on it (Bronstein et al., 2017). The first class of problems is also called *Manifold learning*, and it is the predominant type of problem that we aim to address in this thesis.

4.1 Manifold Learning

Assume to be given a set of data points with some underlying lower dimensional structure embedded into a high-dimensional Euclidean space. Recovering that lower dimensional structure is often referred to as *manifold learning* or *non-linear dimensionality reduction*, and is an instance of unsupervised learning.

These tasks rely in the *manifold hypothesis*, a premise according to which real-world data is expected to concentrate in the vicinity of a manifold $\mathcal{M}$ of much lower dimensionality $d_{\mathcal{M}}$, embedded in high-dimensional input space $\mathbb{R}^n$ (see Figure 4.2). Finding this lower dimensional manifold helps us to avoid overfitting and generalize to unseen data points. We further assume this manifold has an useful geometry in the sense that it is a metric space and has a smooth differentiable structure that allows learning via function optimization. Thus, we are interested in *Riemannian manifolds* of intrinsic dimension $d_{\mathcal{M}} \leq n$.

Many methods for manifold learning consist of two steps: first, they start with constructing a representation of local affinity of the data points, typically, a sparsely connected *graph*. Second, the data points are embedded into a low-dimensional space trying to preserve some criterion of the original affinity. Hence, the **graph embedding problem**

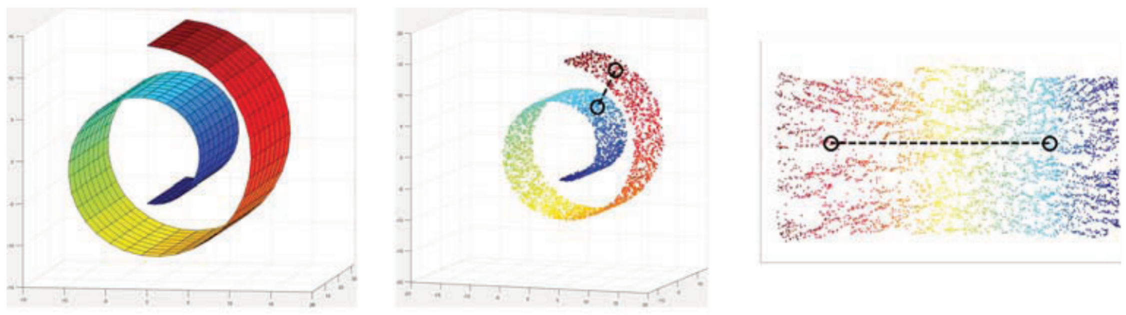


Figure 4.2: An example of the data set including a “swiss roll” structure. The Euclidean distance (center) cannot accurately model the data, as the distance measured in the low-dimensional manifold (right). Source: Wang et al. (2017).

can be viewed as a dimensionality reduction technique for graph-structured data, where the input data is defined on a non-Euclidean, high-dimensional, discrete domain. In this work we study non-Euclidean manifolds with similar structural properties to the shape of the data, in order to achieve graph embeddings with improved representation capacity.

Examples of non-linear dimensionality reduction include different flavors of multidimensional scaling (MDS) (Cox & Cox, 2008), locally linear embedding (LLE) (Roweis & Saul, 2000), stochastic neighbor embedding (t-SNE) (van der Maaten & Hinton, 2008), spectral embeddings such as Laplacian eigenmaps (Belkin & Niyogi, 2001) and deep models (Perozzi et al., 2014; Tang et al., 2015), among others. We review these works in Chapter 5.

4.2 Why Should We Go non-Euclidean?

In this section we analyze theoretical arguments about the universality of Euclidean representations. However, these arguments rely on assumptions that are hard to satisfy in practical scenarios in Deep Learning. Moreover, some of the necessary conditions have associated complications that we discuss in this section.

4.2.1 Universal Approximation Theorem

The capacity of neural networks to approximate almost arbitrary functions is the subject of various Universal Approximation Theorems (Hornik et al., 1989; Cybenko, 1989). In simple terms, the theorem states that a neural network with one hidden layer *of enough neurons* can approximate any continuous function with arbitrary precision.

Although the theorem affirms that such a simple choice of neural architecture yields a dense class of functions, it does not provide a construction for the weights, but merely state that such a construction is possible. Moreover, the single layer may require an infeasibly large amount of neurons, and may fail to learn and generalize correctly (Goodfellow et al., 2016).

4.2.2 Nash Embedding Theorem

The Nash embedding theorem states that any d-dimensional Riemannian manifold can be "isometrically" embedded in an Euclidean space of dimension $d' \leq \frac{d(3d+11)}{2}$ for compact manifolds, and $d' \leq \frac{d(d+1)(3d+11)}{2}$ for non-compact manifolds (Nash, 1956). However, "isometric" in this context means preserving the metric tensor (i.e. the inner-product in each tangent space), meaning that the original manifold is isometric with its embedding image, but not with the entire ambient Euclidean space. This does not imply that the Euclidean distances would match the geodesic distances (Ganea, 2019).

Furthermore, we see the respective squared and cubic growth of the required dimensionality of the target Euclidean space. This poses the challenge of working with very high-dimensional data. In the following section we present some of the issues that arise when working with high dimensions in Deep Learning.

4.2.3 Curse of Dimensionality

Recalling the Universal approximation theorem and the Nash embedding theorem, in theory we can approximate any function or embed any Riemannian manifold into an Euclidean space, *given enough dimensions* in our network. Nonetheless, increasing the dimensionality of a neural model to very large values poses problems, both at training and inference times. We refer to issues caused by working in high-dimensional spaces as the *curse of dimensionality*.

Modern machine learning operates with datasets, by trying to learn a function f with the capacity to interpolate such data. In order to obtain a statistically sound and reliable result, the amount of data needed to support the result often grows exponentially with the dimensionality. This is due to the fact that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse.

Furthermore, organizing and searching data often relies on detecting areas where objects form groups with similar properties. In high-dimensional data, however, all objects appear to be sparse and dissimilar in many ways, which prevents common data organization strategies from being efficient. For example, the minimum and maximum Euclidean distances between N points become indiscernible as the space dimension goes to infinity, which poses a problem for methods such as nearest neighbor search, k-nearest neighbor classification or clustering (Aggarwal et al., 2001).

4.2.4 Limitations of Euclidean Representations

Euclidean geometry has historically been the workhorse for machine learning applications due to its power and simplicity. Nevertheless, the quality of the representations achieved by embeddings is determined by how well the geometry of the embedding space matches the structure of the data. Hence, in many cases Euclidean geometry is not the optimal choice as target embedding space, regardless of the dimensionality. We review some of such cases.

Graphs are a typical example of non-Euclidean metrics. If the input data has a graph structure, the resulting graph metric might be very different from the Euclidean metric, resulting in potentially large embedding distortion, no matter how many dimensions are used.

Another case is when data points are sampled from intrinsically curved Riemannian manifolds. As a consequence, points cannot be embedded in the Euclidean space without potentially large distortion when using the Euclidean distance as an approximation to the geodesic distance. Manifold learning approaches focus on preserving metric properties and are based on a nearest-neighbor search that generally uses straight-line Euclidean distances to model both global and local similarities. This is consistent with a locally "flat" approximation of a manifold, but becomes problematic when a global isometric embedding is desired (e.g. for link prediction or graph generative models). For example, methods like IsoMap (explained in Chapter 5) has zero distortion only for intrinsically flat manifolds, i.e. that can be isometrically embedded onto an Euclidean vector space. Only in such cases geodesic distances would correspond to Euclidean distances in the target embedding space. Otherwise, they would exhibit embedding distortion and error when the data comes from an intrinsically curved manifold.

4.3 Geometric Inductive Bias

We have seen that increasing the dimensionality of the models is a convenient theoretical result, but unattainable in many cases due to lack of enough data to properly train neural networks. Thus, to build models that are able to learn and generalize to inputs that have not been encountered during training, we need to introduce a set of assumptions. The goal of these assumptions is to narrow the output space (without discarding useful hypothesis) and allow a learning algorithm to prioritize one solution or interpretation over another. This is called an *inductive bias or prior* in the model.

Moreover, Euclidean space has been employed as the conventional embedding space but this flat geometry is not the natural geometry of all data structures. The choice of a metric space where to embed the data can be understood as a powerful *geometric inductive bias*. For example, from a theoretical point of view, Gromov (1987) shows that arbitrary tree structures cannot be embedded with arbitrary low distortion in the Euclidean space with any number of dimensions, but this task becomes possible in the hyperbolic space with only 2 dimensions, where the exponential volume growth matches the exponential growth of nodes with the tree depth. Furthermore, the sphere and cycle graphs exhibit a similar behaviour, since they can only be isometrically embedded in spaces of strictly positive curvature (Wilson et al., 2014). Hence, to improve representations for a variety of data types, we propose to explore non-Euclidean spaces of diverse curvatures. We expect these spaces to match the geometry of the data and thus provide higher quality representations.

Finally, traditional neural networks struggle at manipulating non-Euclidean manifolds. Even if we were to set weights by hand, it would be challenging to compactly represent the transformations we want. Therefore, we also propose means to adapt conventional Euclidean operations to other domains, and new neural layers that work by respecting the geometry of different manifolds. Our proposals have several advantages, and also pose some challenges that we detail in the following sections.

4.3.1 Advantages

Imposing inductive biases on models through geometric priors in the metric spaces has several advantages:

- Reduction of the parameter footprint: choosing the appropriate space allows us to operate with lower dimensional embeddings. This results in better generalization capabilities, less overfitting, better quality with less training data, improved computational complexity and better clustering behavior due to less effect of the curse of dimensionality.
- Lower embedding distortion: it implies better preserving local and global geometric information through improved representations.
- Superior arrangement of the embedding space: it results in improved clustering, classification, performance.
- Higher interpretability: deep learning models can be hard to interpret, but a different geometry gives rise to new perspectives.
- New techniques for data analysis: more complex geometries are equipped with tools that provide new insights about the data.

4.3.2 Challenges

We also face challenges associated with moving away from Euclidean geometry:

- Availability of tools that support non-Euclidean geometries: many "trivial" data processing methods, such as dimensionality reduction or unsupervised clustering, are built under Euclidean premises, and it is not easy to generalize them or to integrate non-Euclidean metrics or notions into them.
- Availability of neural layers: most neural architectures employ Euclidean operations that might not be defined in different geometries. The few implementations of neural networks in different geometries are hard to integrate in general pipelines or to combine with conventional Euclidean layers.
- Optimization: if the parameters of the models are lying on different manifolds, we need to optimize them respecting their corresponding geometry. There are very few tools for optimization on Riemannian manifolds.
- Access to closed-form expressions: for generic manifolds, geometric objects such as geodesic equations, exponential map, distance function or parallel transport can easily lose their appealing closed form expressions.
- Computational efficiency: operating in non-Euclidean geometries usually requires more complex operations, which can be orders of magnitude slower than their Euclidean counterparts.

Chapter 5

Related Work and Applications

"All models are wrong, but some are useful."
– George E. P. Box

In this chapter we review related work to graph embedding methods that is relevant to this thesis. Moreover, we also explain tasks and applications where these methods are found useful. Finally, we enumerate their limitations.

5.1 Graph Embedding Methods

Shallow embeddings of nodes can be learned as a mapping function from a discrete graph to a continuous embedding space, such that the structure of the data in the space corresponds to the underlying graph structure. We recall the formal definition from Section 1.1.1: given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, (optionally with weighted adjacency matrix $W \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$), the goal is to learn low-dimensional representations $\{Z_i\}_{i\in\mathcal{V}}$ (embeddings) for nodes in the graph $\{v_i\}_{i\in\mathcal{V}}$, such that important graph properties (e.g. local or global structure) are preserved in the embedding space.

At a high level, the graph embedding problem is similar to dimensionality reduction methods, except that the input data might not have a linear structure. Here, we briefly discuss two major types of shallow graph embedding methods, namely *outer product-based* and *distance-based* methods. The overview of these methods is summarized in Table 5.1.

5.1.1 Outer Product-based Methods

Outer product-based methods rely on pairwise dot-products $\langle\cdot,\cdot\rangle$ to compute node similarities. These methods measure some notion of similarity in the graph, thus higher values mean more similar pairs.

	Method	Model	Publication
Product	Matrix Factorization	GF	Ahmed et al. (2013)
		GraRep	Cao et al. (2015)
	Random Walk	DeepWalk	Perozzi et al. (2014)
		node2vec	Grover & Leskovec (2016)
		LINE	Tang et al. (2015)
Distance	Euclidean	MDS	Cox & Cox (2008)
		IsoMap	Tenenbaum et al. (2000)
		LLE	Roweis & Saul (2000)
		LE	Belkin & Niyogi (2001)
	Non-Euclidean	Poincaré	Nickel & Kiela (2017)
		Lorentz	Nickel & Kiela (2018)
		h-MDS	Sala et al. (2018)
		Mixed Curvature	Gu et al. (2019)
		SPD	Cruceru et al. (2020)

Table 5.1: Overview of shallow graph embedding methods reviewed in this work. We focus on the highlighted non-Euclidean models.

Matrix Factorization Methods

These approaches learn embeddings that lead to a low-rank representation of some similarity matrix S, where $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, is a the weighted adjacency matrix, the Laplacian matrix or more complex similarities derived from proximity measures such as the Katz Index[1]. The matrix factorization method is a simple outer product $\widehat{S} = ZZ^T$, where $Z \in \mathbb{R}^{|\mathcal{V}|\times d}$ are the learned embeddings. These methods learn representations that preserve structural information as defined by the similarity matrix S.

Graph Factorization (GF): Learns a low-rank factorization for the adjacency matrix (Ahmed et al., 2013).

Graph representation with global structure information (GraRep): Most methods operate under symmetric assumptions (undirected graphs). To overcome this limitation, GraRep learns two embeddings per node, a source embedding Z^s and a target embedding Z^t, which capture asymmetric proximity in directed networks (Cao et al., 2015). GraRep learns embeddings that preserve k-hop neighborhoods by approximating the k power of the adjacency matrix S.

[1] https://en.wikipedia.org/wiki/Katz_centrality

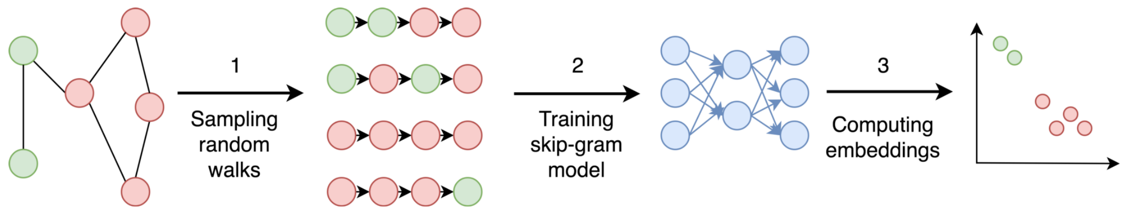


Figure 5.1: Phases of DeepWalk approach. Source: Godec (2018).

Random Walk (or Skip-gram) Methods

The inner-product methods discussed in the previous section all employ deterministic measures of node similarity. They often define the similarity matrix S as some polynomial function of the adjacency matrix. Building on the success of this methods, recent years have seen a surge in models that adapt the inner-product approach to use *stochastic* measures of neighborhood overlap. The key innovation in these approaches is that graph embeddings are optimized so that two nodes have similar embeddings if they tend to co-occur on short random walks over the graph. Perozzi et al. (2014) empirically showed that the frequency statistics induced by random walks also follow Zipf's law, thus motivating the development of skip-gram graph embedding methods. Skip-gram graph embedding models were inspired by efficient NLP methods modeling probability distributions over words for learning word embeddings (Mikolov et al., 2013; Pennington et al., 2014). These embeddings are optimized to predict context words, or surrounding words, for each target word in a sentence. Hence, graph embedding methods exploit random walks on graphs and produce node sequences that are similar in positional distribution, as to words in sentences

DeepWalk: It was the first attempt to generalize skip-gram models to graph-structured data (Perozzi et al., 2014). DeepWalk draws analogies between graphs and language. Specifically, writing a sentence is analogous to performing a random walk, where the sequence of nodes visited during the walk, is treated as the words of the sentence. DeepWalk trains neural networks by maximizing the probability of predicting context nodes for each target node in a graph, namely nodes that are close to the target node in terms of hops and graph proximity. For this purpose, node embeddings are decoded into probability distributions over nodes using row-normalization of the decoded matrix with softmax.

node2vec: It is a random-walk based approach for unsupervised network embedding, that extends DeepWalk's sampling strategy (Grover & Leskovec, 2016). The authors introduce a technique to generate biased random walks on the graph, by combining graph exploration through breadth-first search (BFS) and through depth-first search (DFS). Intuitively, node2vec also preserves high order proximities in the graph but the balance between BFS and DFS allows node2vec embeddings to capture local structures in the graph, as well as global community structures, which can lead to more informative embeddings.

Large scale Information Network Embedding (LINE): The basic idea in LINE is to learn embeddings that preserve first and second order proximity (Tang et al., 2015). The first objective aims to encode first-order adjacency information with an adjacency-based similarity measure. The second objective is more similar to the random walk approaches. It encodes two-hop adjacency information.

5.1.2 Euclidean Distance-based Methods

Distance-based methods optimize embeddings such that points that are close in the graph (as measured by their graph distances for instance) stay as close as possible in the embedding space using a predefined distance function. Formally, the model computes pairwise distance for some distance function $d(\cdot, \cdot)$. Note that while outer-product methods measure some notion of similarity in the graph, in distance-based methods the distance function measures dissimilarity between nodes. Hence, higher values mean less similar pairs of nodes.

Most distance-based methods optimize Euclidean embeddings by minimizing Euclidean distances between similar nodes. Among these, we find linear embedding methods such as principal component analysis[2] (PCA) or MDS, which learn low-dimensional linear projection subspaces, or nonlinear methods such as Laplacian eigenmaps, IsoMAP and Local linear embedding. Note that all these methods have originally been introduced for dimensionality reduction or visualization purposes, but can easily be interpreted in the context of graph embedding.

Multi-Dimensional Scaling (MDS): Refers to a set of embedding techniques used to map objects to positions while preserving the distances between these objects (Kruskal, 1964). In particular, metric MDS (mMDS) (Cox & Cox, 2008) minimizes the difference between the distance of the embeddings in the space and a distance matrix measuring the dissimilarity between objects. That is, mMDS finds an embedding configuration where distances in the low-dimensional embedding space are preserved by minimizing a residual sum of squares called the *stress* cost function. Note that if the dissimilarities are computed from Euclidean distances of a higher-dimensional representation, then mMDS is equivalent to the PCA dimensionality reduction method.

Isometric Mapping (IsoMap): It is an algorithm for non-linear dimensionality reduction which estimates the intrinsic geometry of a data lying on a manifold (Tenenbaum et al., 2000). This method is similar to MDS, except for a different choice of the distance

[2] `https://en.wikipedia.org/wiki/Principal_component_analysis`

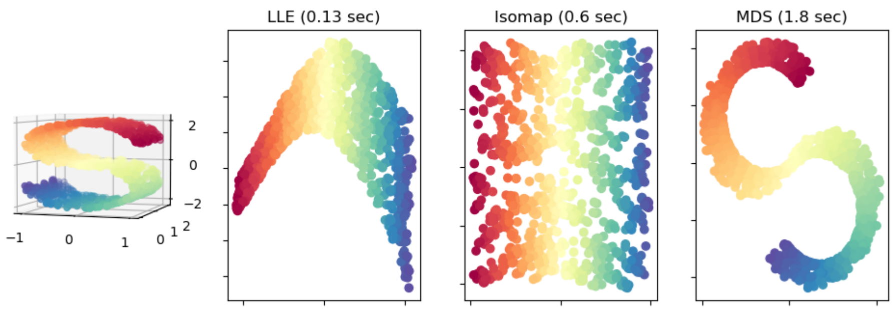


Figure 5.2: Comparison of 2D embeddings applying LLE, IsoMap and MDS for points sampled from the manifold on the left. Source: Pedregosa et al. (2011).

matrix. IsoMap approximates manifold distances (in contrast with straight-line Euclidean geodesics) by first constructing a discrete neighborhood graph $\mathcal{G}$, and then using the graph distances (length of shortest paths computed using Dijkstra's algorithm for example) to approximate the manifold geodesic distances. IsoMap then uses the an MDS algorithm to compute representations that preserve these graph geodesic distances. Different from MDS, IsoMap works for distances that do not necessarily come from a Euclidean metric space (e.g. data defined on a Riemannian manifold). However, it has zero distortion only for intrinsically flat manifolds, i.e. that can be isometrically embedded onto an Euclidean vector space, that have zero intrinsic curvature. In this case, geodesic distances would correspond to Euclidean distances in the target embedding space.

Locally Linear Embedding (LLE): It is another non-linear dimensionality reduction technique which was introduced around the same time as IsoMap and improves over its computational complexity via sparse matrix operations (Roweis & Saul, 2000). Different from IsoMAP which preserves the global geometry of manifolds via geodesics, LLE is based on the local geometry of manifolds and relies on the assumptions that when locally viewed, manifolds are approximately linear. The main idea behind LLE is to approximate each point using a linear combination of embeddings in its local neighborhood (linear patches). These local neighborhoods are then compared globally to find the best non-linear embedding. See a comparison of methods in Figure 5.2.

Laplacian Eigenmaps (LE): It is a non-linear dimensionality reduction methods that seeks to preserve local distances (Belkin & Niyogi, 2001). Spectral properties of the graph Laplacian matrix capture important structural information about graphs. In particular, eigenvectors of the graph Laplacian provide a basis for smooth functions defined on the graph vertices (the "smoothest" function being the constant eigenvector corresponding to eigenvalue zero). LE is a non-linear dimensionality reduction technique which builds on this intuition. LE first constructs a graph from datapoints (e.g. k-NN graph

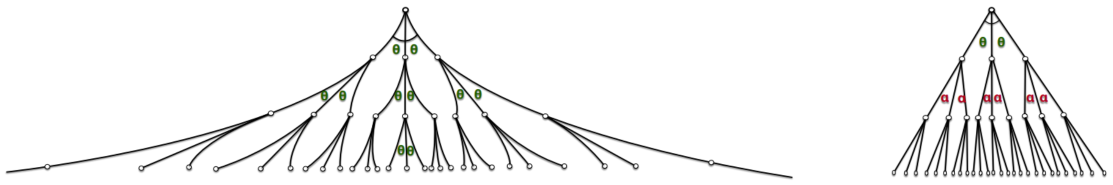


Figure 5.3: Distances and angles for a tree embedded in hyperbolic and Euclidean spaces. In hyperbolic space, as the tree grows the angles between the edges (θ) can be preserved from one level to the next. In Euclidean space, since the number of nodes in the tree grows faster than the rate that the volume grows, angles may not be preserved (θ to α). Lines in the left diagram are straight in hyperbolic space, but appear curved in this Euclidean diagram. Source: Gulcehre et al. (2019).

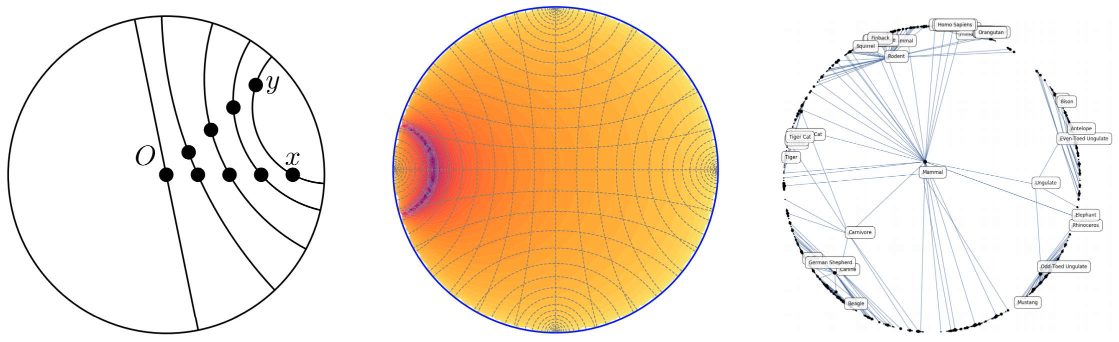
or ϵ-neighborhood graph) and then represents nodes in the graphs via the Laplacian's eigenvectors corresponding to smaller eigenvalues. The high-level intuition for LE is that points that are close on the manifold (or graph) will have similar representations, due to the "smoothness" of Laplacian's eigenvectors with small eigenvalues.

5.1.3 Non-Euclidean Distance-based Methods

The distance-based methods described so far assumed embeddings are learned in a Euclidean space. Graphs are non-Euclidean discrete data structures, and several works proposed to learn graph embeddings into non-Euclidean spaces instead of conventional Euclidean space. Examples of such spaces include hyperbolic space, spherical space, and symmetric positive definite matrices spaces. Note that since these spaces have a manifold structure, embeddings need to be optimized using Riemannian optimization techniques (Bonnabel, 2011; Bécigneul & Ganea, 2019) to ensure that they remain on the manifold. We thoroughly discuss these methods given that they are the closest to our work, and that they are employed as baselines in different experiments.

Hyperbolic space

Before its use in machine learning applications, hyperbolic geometry has been extensively studied and used in network science research (Gromov, 1987; Krioukov et al., 2009, 2010). As already explained, while the volume of the Euclidean space grows polynomially with the distance to the origin, in the hyperbolic space the volume grows exponentially. This is analog to the number of nodes of a full tree growing exponentially with its depth (see Figure 5.3). Moreover, if we consider the origin O and two points, x and y, moving towards the outside of the disk, i.e. $\|x\|, \|y\| \to 1$, the distance $d_H(x, y)$ grows exponentially, and it tends to $d_H(x, O) + d_H(O, y)$ (see Figure 5.4a). This is, the path between x and y converges to a path through the origin. This behaviour can be seen as the continuous analogue to a discrete tree-like hierarchical structure, where the shortest path between



(a) As x and y move towards the outside of the disk, the distance $d_{\mathbb{D}}(x, y) \to d_{\mathbb{D}}(x, 0) + d_{\mathbb{D}}(0, y)$. Source: Sala et al. (2018).

(b) Plot of the $\log$ of geodesic distances to the hyperplane defined by the purple line. Source: Kochurov et al. (2020).

(c) 2d Poincaré embeddings of the WordNet mammals subtree. Source: Nickel & Kiela (2017).

Figure 5.4: As points move towards the boundary of the Poincaré disk, the distance between them grows exponentially. This is in stark contrast with the Euclidean intuition.

two sibling nodes goes through their common ancestor. Thus, hyperbolic spaces can be thought of as continuous versions of trees or vice versa, trees can be thought of as "discrete hyperbolic spaces" (Nickel & Kiela, 2017). Due to all these properties there has been a recent interest in learning hyperbolic representations of hierarchical graphs or trees, via gradient-based optimization.

Poincare Embeddings: Nickel & Kiela (2017) learn Poincaré embeddings of hierarchical graphs such as lexical databases (e.g. WordNet) in the Poincaré model of hyperbolic space. Embeddings are then learned by minimizing hyperbolic distances between connected nodes while maximizing distances between disconnected nodes.

To embed hierarchies in the Poincaré ball, items near the top of the hierarchy are placed near the origin of the space, while lower items near the boundary of the ball, intuitively, embedding the "vertical" structure. Furthermore, items sharing a parent in the hierarchy are close to each other, embedding the "horizontal" structure. In this manner, the learned embeddings capture notions of both similarity, through the relative distance among each other, and hierarchy, through the distance to the origin, i.e. the norm. Figure 5.4c shows as an example the result of embedding the WordNet mammal hierarchy in a 2-dimensional Poincaré disk.

Formally, let $\mathcal{D} = \{(u, v)\}$ be the set of observed directed edges between node pairs. The model learns embeddings of all symbols in $\mathcal{D}$ by minimizing the loss function

$$\mathcal{L}(\Theta) = \sum_{(u,v)\in\mathcal{D}} \log \frac{e^{-d(\mathbf{u},\mathbf{v})}}{\sum_{\mathbf{v'}\in\mathcal{Q}(u)} e^{-d(\mathbf{u},\mathbf{v'})}} \tag{5.1}$$

Where $d(\cdot,\cdot)$ is the hyperbolic distance and $\mathcal{Q}(u) = \{v|(u, v) \notin \mathcal{D}\} \cup \{u\}$ is the set of negative samples for u (including u). Since they work in the Poincaré model of hyperbolic space, they need to constrain embeddings to remain with the unit ball. To do so they employ the projection:

$$\mathrm{proj}(\theta) = \begin{cases} \theta/||\theta|| - \epsilon & \text{if } ||\theta|| \geq 1 \\ \theta & \text{otherwise} \end{cases}$$

where ϵ is a small constant to ensure numerical stability.

Lorentz embeddings: While the Poincaré embeddings are designed for embedding unweighted undirected graphs, in this case the same authors extend the approach to a more general setting: inferring continuous hierarchies from pairwise similarity measurements (Nickel & Kiela, 2018). Formally, let $\mathcal{C} = \{c_i\}_{i=1}^m$ be a set of concepts and $X \in \mathbb{R}^{m \times m}$ be a dataset of pairwise similarity scores between these concepts. Moreover, the concepts can be organized according to an unobserved hierarchy $(\mathcal{C}, \preceq)$, where $c_i \preceq c_j$ defines a partial order over the elements of $\mathcal{C}$. Given this setting, the goal is then to recover the partial order $(\mathcal{C}, \preceq)$ from X, by learning embeddings whose objective is again to reflect semantic similarities and hierarchical information. They explore a different model of hyperbolic space, namely the Lorentz model (also known as the hyperboloid model), and show that it provides better numerical stability than the Poincaré model.

h-MDS: Sala et al. (2018) propose to learn hyperbolic embeddings by translating MDS to the hyperbolic domain, in particular to the Poincaré model. A key technical step in the algorithm is normalizing the matrix of hyperbolic distances so that the center of mass of the points is at the origin. This mirrors the Euclidean solution to the problem. They show that h-MDS solutions that involve points near the edge of the disk are more sensitive to noise than traditional Euclidean MDS.

Cartesian Products of Spaces

Another line of work extends non-Euclidean embeddings to mixed-curvature product spaces (Tifrea et al., 2019; Skopek et al., 2020). They simultaneously contain Euclidean, spherical and hyperbolic spaces, providing a space of heterogeneous curvature suitable for a wide variety of structures. In this manner, they allow to accommodate graphs of very dissimilar structure, bringing more flexibility for different types of graphs with compound geometries (e.g. ring of trees). In the work of Gu et al. (2019) specifically, they form a Riemannian product manifold combining hyperbolic, spherical, and Euclidean components and equip it with a decomposable Riemannian metric. While each component space in the product has constant curvature (positive for spherical, negative for hyperbolic, and zero for

Euclidean), the resulting mixed space has non-constant curvature. Moreover, they learn the appropriate curvature for the hyperbolic and spherical submanifolds through Riemannian optimization.

Formally, they consider a sequence of smooth manifolds $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$. The product manifold is defined as the Cartesian product $\mathcal{M} = \mathcal{M}_1 \times \mathcal{M}_2 \times \ldots \times \mathcal{M}_k$. Points $p \in \mathcal{M}$ have coordinates $p = (p_1, \ldots, p_k) : p_i \in \mathcal{M}_i$. If the $\mathcal{M}_i$ are equipped with metric tensor g_i, then the product $\mathcal{M}$ is also Riemannian with a metric tensor $g(u, v) = \sum_{i=1}^{k} g_i(u_i, v_i)$. That is, the product metric decomposes into the sum of the constituent metrics. The same occurs for exponential maps and squared distances. To compute embeddings, they optimize points with the following loss function. Given graph distances $\{d_{\mathcal{G}}(X_i, X_j)\}_{ij}$ between all pairs of connected nodes, the loss is defined as:

$$\mathcal{L}(x) = \sum_{1 \leq i \leq j \leq n} \left| \left(\frac{d_{\mathcal{M}}(x_i, x_j)}{d_{\mathcal{G}}(X_i, X_j)} \right)^2 - 1 \right| \tag{5.2}$$

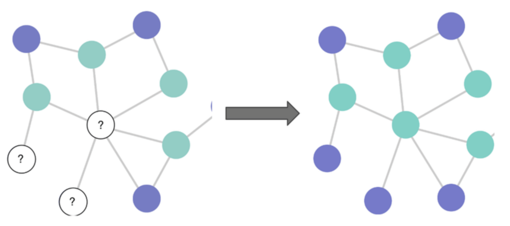
where $d_{\mathcal{M}}(x_i, x_j)$ is the distance between the corresponding node representations in the embeddings space. This formulation of the loss function seeks to minimize the average distortion. For optimization, they adapt RSGD (Bonnabel, 2011) to decompose per component, and update the corresponding coordinates of the points respecting the geometry of each submanifold.

Symmetric Positive Definite Matrix Space

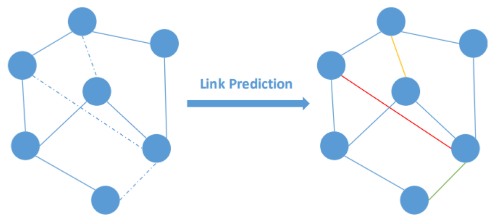
The SPD space is a manifold of non-positive curvature, where points are represented as positive definite real symmetric matrices. It contains Euclidean and hyperbolic subspaces as well as products of hyperbolic planes. Cruceru et al. (2020) extends the approach of Gu et al. (2019), and takes the hypothesis of varying curvature yielding higher flexibility further by exploring the representation properties of several irreducible spaces of non-constant sectional curvature. They use, in particular, SPD and Grasmannian manifolds, *i.e.* Riemannian manifolds where points are represented as specific types of matrices that offer a convenient trade-off between semantic richness and algorithmic tractability. Since these manifolds are very well studied (Bridson & Häfliger, 2011; Edelman et al., 1999), the geometric tools required to compute distances and perform Riemannian optimization are already available through closed-form expressions.

5.2 Tasks and Applications

We seek to build graph embedding models that can learn from data in order to solve particular tasks. These models can be applied to a wide range of applications. Here we



(a) Node classification. Source: Mishra et al. (2021).



(b) Link prediction. Source: Wang et al. (2021).

Figure 5.5: Applications of graph embeddings.

highlight the most common ones.

Node classification: It is an important supervised graph application, where the goal is to learn node representations that can accurately predict node labels, which could be types, categories, or attributes (see Figure 5.5a). For instance, node labels could be scientific topics in citation networks, or categorical attributes in social networks. Often, to train we assume that we have label information only for a very small subset of the nodes in a single graph: $\mathcal{V}_{train} \subset \mathcal{V}$. To classify the remaining ones, models that try to leverage local information by assigning similar labels to neighboring nodes in a graph exploit *homophily*, which is the tendency for nodes to share attributes with their neighbors in the graph (McPherson et al., 2001). Other models work under the hypothesis of *structural equivalence* (Donnat et al., 2018), which is the idea that nodes with similar local neighborhood structures will have similar labels. Note that node features can significantly boost the performance on node classification tasks if these are descriptive for the target label. Thus, graph neural networks employing this information usually achieve state-of-the-art performance in most benchmarks (Wu et al., 2019).

Examples of applications of node classification include classifying the function of proteins (Hamilton et al., 2017), document topics (Kipf & Welling, 2017) and detecting bots in social networks (Zhou et al., 2020).

Link prediction: It is the task of predicting links or, more generally, relations in a graph (see Figure 5.5b). In other words, the goal in link prediction tasks is to predict missing or unobserved links (e.g. links that may appear in the future for dynamic and temporal networks). Link prediction can also help identifying spurious link and remove them. A common approach for training link prediction models is to mask some edges in the graph (positive and negative edges), train a model with the remaining edges ($\mathcal{E}_{train} \subset \mathcal{E}$) and then test it on the masked set of edges ($\mathcal{E} \setminus \mathcal{E}_{train}$). For undirected graphs, models that apply simple heuristics based on how many neighbors two nodes share can achieve strong performance. On the other hand, in more complex multi-relational graph datasets, such

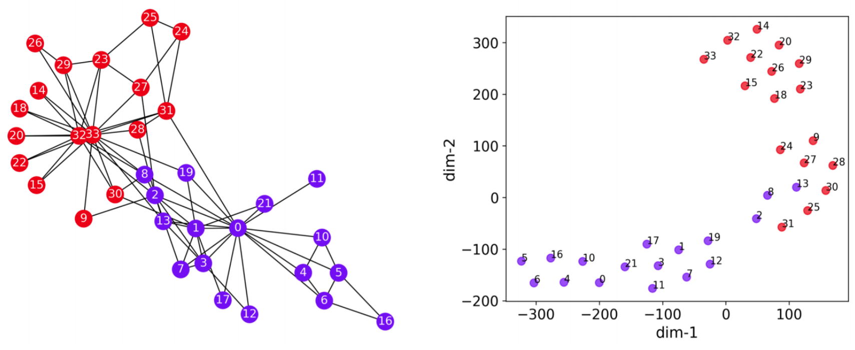


Figure 5.6: Example of applying graph embedding method for community detection in the Zachary's karate club dataset. Source: Xu (2020).

as knowledge graphs, relation prediction can require complex reasoning and inference strategies (Nickel et al., 2016).

It is a major application of graph embedding models in industry, and common example of applications include predicting friendships in social networks (Fan et al., 2019), content recommendation (Ying et al., 2018), predicting drug-side effects (Zitnik et al., 2018) or inferring new facts in a relational databases (Bordes et al., 2013).

Clustering and Community Detection: The challenge of community detection is to infer latent community structures, *i.e.* where nodes are much more likely to form edges with nodes that belong to the same community, given only the input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. It is the graph analogue of unsupervised clustering. Real-world applications include uncovering functional modules in neuroscience (Martinet et al., 2020), detecting anomalies in traffic analysis (Chen et al., 2012) and fraudulent groups of users in financial transaction networks (Pandit et al., 2007).

Graph Classification or Regression: In these tasks we seek to learn over graph data, but instead of making predictions over the individual components of a single graph (*i.e.*, the nodes or the edges), we are instead given a dataset of multiple different graphs and our goal is to make independent predictions specific to each graph. Each graph is considered as a datapoint associated with a label, and the goal is to use a labeled set of training points to learn a mapping from datapoints. They are particularly challenging task because they require some notion of pooling, in order to aggregate node-level information into graph-level information. Generalizing this notion of pooling to arbitrary graphs is non trivial because of the lack of regularity in the graph structure.

Common applications include predicting molecule's toxicity or solubility (Gilmer et al., 2017), malware detection (Chau et al., 2011) or vulnerability detection in software systems (Li et al., 2019).

5.3 Limitations

In this chapter we have reviewed shallow embedding methods related to our work. Most of these models map each node in the graph to one node embedding that is optimized in the target space. Although this approach has achieved many successes it is also important to note some important drawbacks.

- Since models directly optimizes a unique embedding for each node, there is no part of the network with shared parameters. This is a drawback, given that parameter sharing can improve the efficiency of learning and also act as a powerful form of regularization. Moreover, the number of parameters in shallow embedding methods necessarily grows as $\mathcal{O}(|\mathcal{V}|)$, which can be intractable in massive graphs.

- Shallow embedding approaches do not leverage node features. Many graph datasets have rich feature information, which could potentially be informative in the encoding process, and this information is discarted by these methods.

- Shallow embedding methods are transductive. These approaches can only generate embeddings for nodes that are present during the training phase. Therefore, they fail to generalize to new nodes (e.g. evolving graphs) or new graph instances. Generating embeddings for new nodes is not possible unless additional training is performed. We note, however, that recent theoretical results by Srinivasan & Ribeiro (2020) show that models previously assumed to be capable of only one setting (e.g. only transductive) can be used in both.

Part II

Embeddings Graphs in Hyperbolic Space

Chapter 6

Constructing and Exploiting Hierarchical Graphs

"All animals are equal, but some animals are more equal than others."
– George Orwell, 'Animal Farm'

In previous chapters we have reviewed the limitations of Euclidean methods and the advantages of representing graphs in non-Euclidean manifolds. We established that hyperbolic spaces are a natural fit for representing graphs with hierarchical structures, such as trees (see §1.2 and §5.1.3). While the intrinsic advantages of hyperbolic embeddings are well-established, their usefulness in downstream tasks is, so far, less clear. We believe this is due to three difficulties: First, incorporating hyperbolic embeddings into a neural model is non-trivial since training involves optimization in hyperbolic space. Second, it is not clear how models for downstream tasks can profit from the hierarchical information encoded into these embeddings. And third, it is often difficult to determine what the best hierarchy for the task at hand is.

To investigate possible solution to these issues we choose the downstream task of Entity typing. Entity typing classifies textual mentions of entities according to their semantic class or type. These types tend to be organized into inventories that exhibit hierarchical arrangements (see Figure 6.1 for an example). Thus, we consider it as an adequate application where the integration of hyperbolic embeddings could yield noticeable benefits.

In this chapter we study the ability of hyperbolic embeddings to capture hierarchical relations between mentions in context and their target types in a shared vector space. To do so, we pose the entity typing task as a graph embedding problem, where the graph is derived from the type inventory, followed by a nearest neighbor classifier. We propose

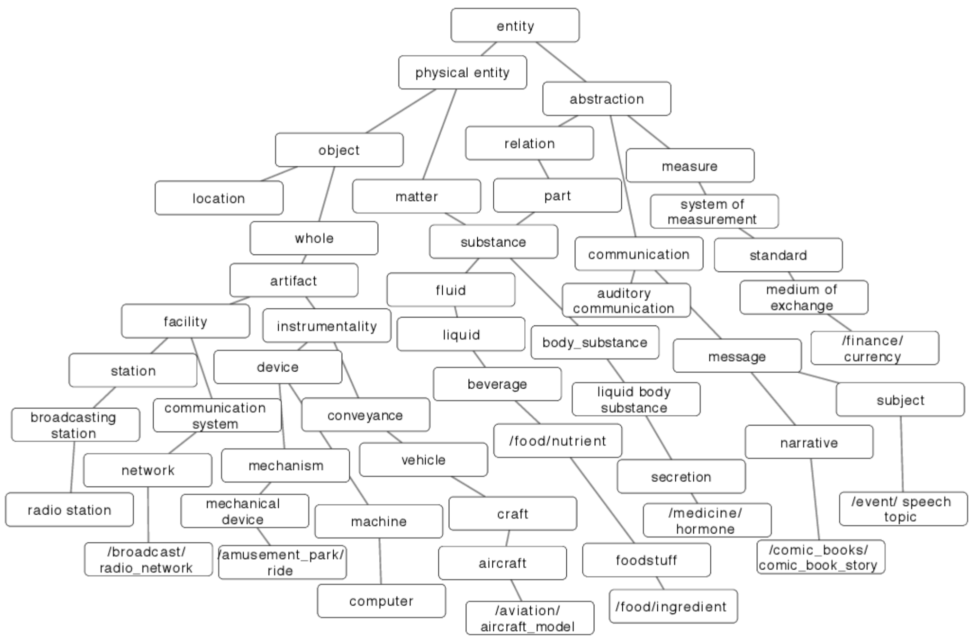


Figure 6.1: Sampled types from the TypeNet inventory. Source: Murty et al. (2017).

two different techniques for creating a large hierarchical entity type inventory: from an expert-generated ontology and by automatically mining type co-occurrences. We model the inventories as graphs, and perform a thorough graph analysis that help us to understand their fitness into hyperbolic spaces. We learn and compare graph embeddings for entity types in both Euclidean and hyperbolic space. Our evaluation on two datasets shows that the hyperbolic model yields improvements over its Euclidean counterpart in some, but not all cases. Our analysis suggests that the adequacy of this geometry depends on the granularity of the type inventory, and the way hierarchical relations are inferred.

6.1 Entity Typing in Hyperbolic Space

Entity typing is a Natural Language Processing (NLP) task that classifies textual mentions of entities according to their semantic class (see Figure 6.2). Formally, we define it as:

Definition 6.1 (Fine-grained Entity Typing)**.** The task we consider is, given a context sentence c containing an entity mention m, predict the correct type labels t_m that describe m from a predefined type inventory T. The mention m can be a named entity, a nominal, or a pronoun. The ground-truth type set t_m may contain multiple types, making the task a multi-label classification problem.

The task has progressed from finding company names (Rau, 1991), to recognizing coarse classes (*person*, *location*, *organization*, and *other*, Sang & De Meulder (2003)), to fine-grained inventories of about one hundred types, with finer-grained types proving beneficial in applications such as relation extraction (Yaghoobzadeh et al., 2017) and question

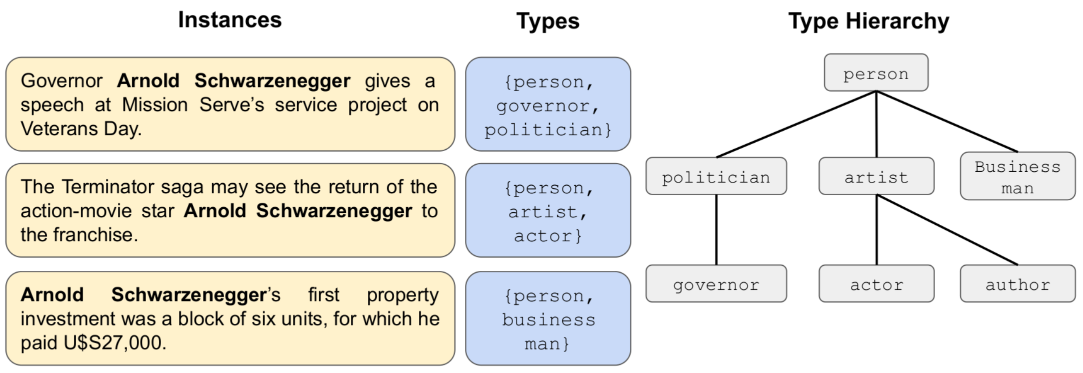

Instances
Types
Type Hierarchy
Governor **Arnold Schwarzenegger** gives a speech at Mission Serve's service project on Veterans Day.
{person, governor, politician}
The Terminator saga may see the return of the action-movie star **Arnold Schwarzenegger** to the franchise.
{person, artist, actor}
Arnold Schwarzenegger's first property investment was a block of six units, for which he paid U$S27,000.
{person, business man}
person
politician
artist
Business man
governor
actor
author

Figure 6.2: Example of entity typing where the same entity should be assigned with different types depending on the context. Moreover, the types can have different level of granularity.

answering (Yavuz et al., 2016). The trend towards larger inventories has culminated in *ultra-fine* and *open* entity typing with thousands of classes (Choi et al., 2018; Zhou et al., 2018).

However, large type inventories pose a challenge for the common approach of casting entity typing as a multi-label classification task (Yogatama et al., 2015; Shimaoka et al., 2016), since exploiting inter-type correlations becomes more difficult as the number of types increases. A natural solution for dealing with a large number of types is to organize them in hierarchy ranging from general, *coarse* types such as "object" near the top, to more specific, *fine* types such as "vehicle" in the middle, to even more specific, *ultra-fine* entity types such as "aircraft" at the bottom (see Figure 6.1). By virtue of such a hierarchy, a model learning about aircrafts will be able to transfer this knowledge to related entities such as other vehicles.

6.1.1 Related Work

Type inventories for the task of fine-grained entity typing (Ling & Weld, 2012; Gillick et al., 2014; Yosef et al., 2012) have grown in size and complexity (Del Corro et al., 2015; Murty et al., 2017; Choi et al., 2018). Systems have tried to incorporate hierarchical information on the type distribution in different manners. Ren et al. (2016a) learns embeddings for mentions and type-paths extracted from the hierarchy. Ma et al. (2016) proposed to incorporate prototypical information as well. Abhishek et al. (2017) applies a joint-learning schema into a shared space, but uses hierarchical information only to mitigate noise in the label annotations. Shimaoka et al. (2017) encode the hierarchy through a sparse matrix. Hu et al. (2015) and Xin et al. (2018) incorporate relational information by using knowledge graph-based representation learning. Ren et al. (2016b); Murty et al. (2018) and Xu & Barbosa (2018) model the relations through a hierarchy-aware loss function.

Different models have been proposed where types and feature representations were embedded into a low dimensional space in order to facilitate information sharing among them (Yogatama et al., 2015; Ma et al., 2016; Abhishek et al., 2017). All previous models aim to measure entity-context similarity under Euclidean assumptions. Instead, we impose a hyperbolic geometry to enrich the hierarchical information.

Finally, our work resembles Xiong et al. (2019) since they derive hierarchical information in an unrestricted fashion through type co-occurrence statistics from the dataset. By means of these co-occurrences, we derive a weighted graph that carries information about relationships between the different types.

6.1.2 Objective

We aim to analyze the effects of hyperbolic and Euclidean spaces when modeling hierarchical information present in the type inventory, for the task of fine-grained entity typing. Since hyperbolic geometry is naturally equipped to model hierarchical structures, we hypothesize that this enhanced representation will result in superior performance.

With the goal of examining the relation between the metric space and the hierarchy, we propose a regression model combined with a nearest neighbor classifier. In this case, "nearest" is defined according to the geometry of the target embedding space: Euclidean or hyperbolic. The steps of our proposed approach are:

1. Derive a graph from the type inventory. This can be done automatically by mining type co-ocurrences on the dataset, or by aligning the types to a predefined structure (e.g. WordNet).

2. Embed the hierarchy in the target space $\mathbb{S}^n$ (Euclidean or hyperbolic space). This is, learn type embeddings $t_i \in \mathbb{S}^n$ according to their connections in the graph.

3. Learn a function $f : \mathbb{R}^{n'} \rightarrow \mathbb{S}^n$ that maps feature representations of a mention m and its context c into the embedding $v \in \mathbb{S}^n$ lying on embedding space, such that the instances are embedded closer to their target types.

4. Look for the k nearest type embeddings t_i to v, and assign them as the types that characterize the mention m in the context c.

The ground-truth type set contains a varying number of types per instance. In our setup, however, we aim to predict a fixed amount k of labels for all the instances. This imposes strong upper bounds to the performance of our proposed model. Nonetheless, as the strict accuracy of state-of-the-art methods for the Ultra-Fine dataset is below 40% (Choi et al., 2018; Xiong et al., 2019), the evaluation we perform is still informative in qualitative terms, and enables us to gain better intuitions with regard to embedding hierarchical structures in different metric spaces.

6.2 Hierarchical Type Inventories

In this section, we investigate two methods for deriving a hierarchical structure for a given type inventory. First, we introduce the datasets on which we perform our study since we exploit some of their characteristics to construct a hierarchy.

6.2.1 Data

We focus our analysis on the the Ultra-Fine entity typing dataset introduced in Choi et al. (2018). Its design goals were to increase the diversity and coverage entity type annotations. It contains 10,331 target types defined as free-form noun phrases and divided in three levels of granularity: *coarse*, *fine* and *ultra-fine* (see Figure 6.3). The data consist of 6,000 crowdsourced examples and approximately 6M training samples in the open-source version[1], automatically extracted with distant supervision, by entity linking and nominal head word extraction. Our evaluation is done on the original crowdsourced dev/test splits. Note that besides the division given by the granularity, there is no information in this dataset specifying hierarchical relationships between the types.

[1]Choi et al. (2018) uses the licensed Gigaword to build part of the dataset resulting in about 25.2M training samples.

Split	**Samples**	**Coarse**	**Fine**	**Ultra-fine**
Train	6,240,105	2,148,669	2,664,933	3,368,607
Dev	1,998	1,612	947	1,860
Test	1,998	1,598	964	1,864

Table 6.1: Amount of samples with at least one label of the granularity organized by split on Ultra-Fine Dataset.

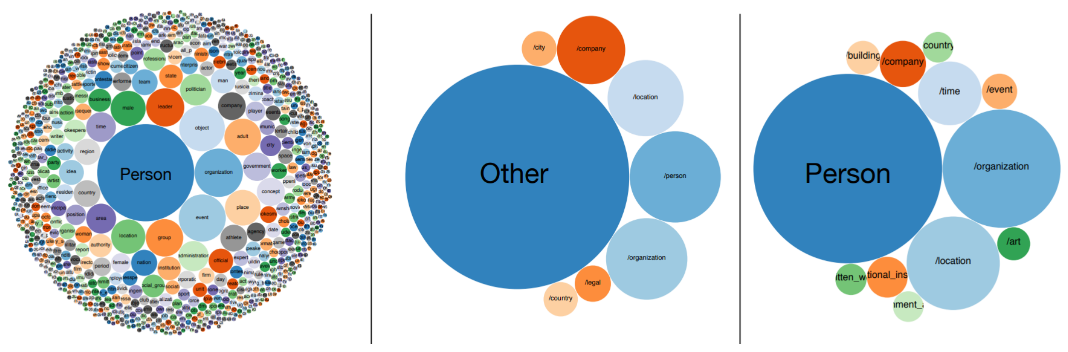


Figure 6.3: Visualization of the type distribution for Ultra-Fine (left) (Choi et al., 2018), OntoNotes (center) (Gillick et al., 2014) and FIGER (right) (Ling & Weld, 2012) datasets. Bubble sizes are proportional to the type frequency. The Ultra-Fine dataset is much more diverse and fine grained. Source: Choi et al. (2018).

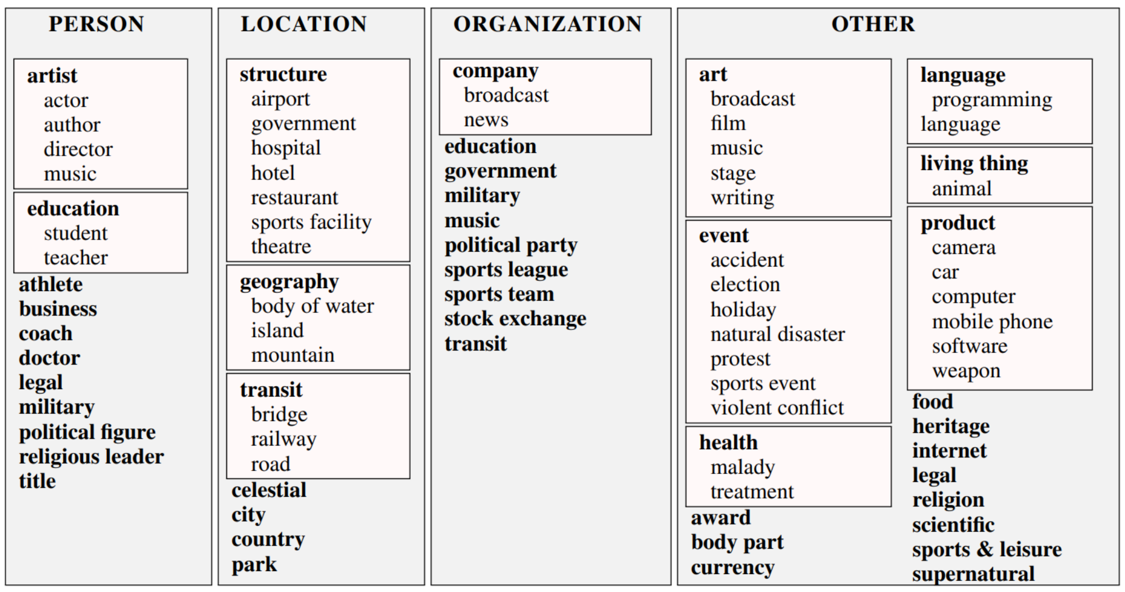


Figure 6.4: Type taxonomy from OntoNotes (Gillick et al., 2014). It includes 89 types at three levels. Source: Gillick et al. (2014).

To gain a better understanding of the proposed model under different geometries, we also experiment on the OntoNotes dataset (Gillick et al., 2014) as it is a standard benchmark for entity typing. From the taxonomy depicted in Figure 6.4 we can observe that there is a predefined hierarchical arrangement between the types of this dataset. Statistics for both datasets are presented in Table 6.2.

6.2.2 Deriving the Hierarchies

The two methods we analyze to derive a hierarchical structure from the type inventory are the following.

Knowledge base alignment: Hierarchical information can be provided explicitly, by aligning the type labels to a knowledge base schema. In this case the types follow the

Dataset	Split	Samples	Coarse	Fine	Ultra-fine
Ultra-Fine	Train	6,240,105	2,416,593	4,146,143	3,997,318
	Dev	1,998	1,918	1,289	7,594
	Test	1,998	1,904	1,318	7,511
OntoNotes	Train	793,487	828,840	735,162	301,006
	Dev	2,202	2,337	869	76
	Test	8,963	9,455	3,521	417

Table 6.2: Type instances for both dataset grouped by split and granularity.

Sentence	**Annotation**
...when the **president** said...	politician, president
...during the negotiation, **he**...	politician, diplomat
...after the last meeting, **she**...	politician, president
...the **president** argued...	politician, president

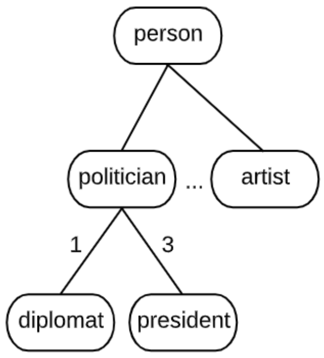


Figure 6.5: Diagram of how we mine type co-occurrences and draw a weighted graph from it.

structure of the ontology (typically tree-like) curated by experts. On the Ultra-Fine dataset, the type vocabulary T (*i.e.* noun phrases) is extracted from WordNet (Miller, 1992). Nouns in WordNet are organized into a deep hierarchy, defined by hypernym or "IS A" relationships. By aligning the type labels to the hypernym structure existing in WordNet, we obtain a type hierarchy. In this case, all paths lead to the root type *entity*, similar to Figure 6.1. In the OntoNotes dataset the annotations follow a pre-established, much smaller, hierarchical taxonomy based on "IS A" relations, as well (see Figure 6.4).

Type co-occurrences: Although in practical scenarios hierarchical information may not always be available, the distribution of types has an implicit hierarchy that can be inferred automatically. If we model the ground-truth labels as nodes of a graph, its adjacency matrix can be drawn and weighted by considering the co-occurrences on each instance. That is, if t_1 and t_2 are annotated as true types for a training instance, we add an edge between both types. To weigh the edge we explore two variants: the frequency of observed instances where this co-relation holds, and the *pointwise mutual information* (pmi), as a measure of the association between the two types.[2] By mining type co-occurrences present in the dataset as an affinity score, the hierarchy can be inferred (see Figure 6.5). This method alleviates the need for a type inventory explicitly aligned to an ontology or pre-defined label correlations.

[2]We adapt pmi in order to satisfy the condition of non-negativity.
This is: $pmi(x, y) \geq 0 \wedge pmi(x, y) = pmi(y, x) \; \forall \; x, y \in T$.

On a Universal Taxonomy

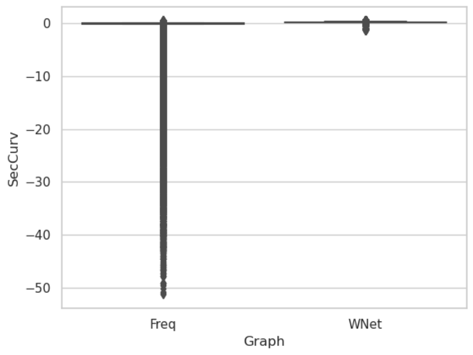
John Wilkins (1614–1672) is an English philosopher who proposed a universal language based on a classification system that would encode a description of the thing a word describes into the word itself. For example, *Zi* identifies the genus *beasts*, *Zit* denotes the "difference" *rapacious beasts of the dog kind*, and finally *Zit*α specifies *dog*. In response to this proposal, and in order to illustrate the arbitrariness and cultural specificity of any attempt to categorize the world, the Argentine writer Jorge Luis Borges (1899–1986) wrote the short essay "The Analytical Language of John Wilkins" (Borges, 1964). In this essay, Borges believes he finds "ambiguities, redundancies and deficiencies" on Wilkins' language, and describes an alternate taxonomy, supposedly taken from an ancient Chinese encyclopedia entitled "Celestial Emporium of Benevolent Knowledge". The list divides all animals into 14 categories:

- Those belonging to the Emperor
- Embalmed ones
- Trained ones
- Suckling pigs
- Mermaids
- Fabled ones
- Stray dogs
- Those included in this classification
- Those that tremble as if they were mad
- Innumerable ones
- Those drawn with a very fine camel hair brush
- Et cetera
- Those that have just broken the flower vase
- Those that from afar look like flies

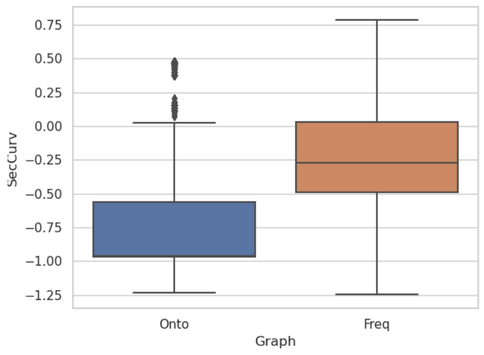
Borges concludes that *"there is no description of the universe that is not arbitrary and conjectural for a simple reason: we do not know what the universe is"*.

6.2.3 Graph Analysis

In this section we analyze the graphs derived from the type inventories with the two aforementioned techniques. Our goal is to examine their fitness to learn representations in Euclidean or hyperbolic space. Statistics for the generated graph can be found in Table 6.3. We can observe that the frequency-based type co-occurrence graphs (FREQ) have significantly more edges that the ones derived from the knowledge based alignments (WORDNET and ONTO respectively). This is due to the fact that even if different types co-occur only once in the dataset annotations, we add an edge between them, therefore many of these edges come from noisy annotations. Nonetheless, the FREQ graphs are



(a) Ultra-Fine dataset. *WNet* refers to the graph aligned with WordNet.



(b) In the OntoNotes dataset, *Onto* refers to the dataset taxonomy.

Figure 6.6: Distribution of sectional curvature for the analyzed datasets. In both plots, *Freq* refers to the graph automatically mined from type co-occurrences.

weighted by the frequency of the co-ocurrences, hence the edges with very low weights do not play a significant role in the analysis.

In the table we report the δ-hyperbolicity. Note that the lower this value is, the graph is more tree-like (ergo, hyperbolic-like). We see that both FREQ graphs exhibit lower values of maximum δ-hyperbolicity when compared to the knowledge base ones, suggesting that the FREQ graphs could be better accommodated in a hyperbolic space.

The mean sectional curvature of the graphs can also be seen in the table. We display the distribution of this value over the sampled triangles in the graph in Figure 6.6. We see that the distribution of the sectional curvature for the FREQ graph in the UltraFine dataset is very skewed towards the negative side, in line with the observations about the hyperbolicity. For the WORDNET graph, the sectional curvature is very close to zero. On the OntoNotes dataset we see that both distributions of the sectional curvatures tend to be negative.

	Ultra-Fine		OntoNotes	
	WORDNET	FREQ	ONTO	FREQ
Nodes	8,662	8,779	89	89
Edges	37,889	170,674	132	1,068
Weighted	No	Yes	No	Yes
δ-hyperbolicity mean	0	0	0	0
δ-hyperbolicity max	0.5	0	0.5	0
Sectional Curvature	0.17±0.27	-0.08±0.83	-0.74±0.42	-0.22±0.35

Table 6.3: Statistics collected from the generated graph for the UltraFine and OntoNotes datasets.

Since the Ollivier-Ricci curvature characterizes the space locally, we plot the graphs for the Ultra-Fine dataset in Figure 6.7. We can observe that nodes and edges in the WORDNET graph exhibit both positive (in blue color) and negative curvature (in red color). The size of the nodes in the plots reflects their degree. We see highly connected nodes to be more on the red spectrum, thus showing negative curvature. This can be explained by the fact that negatively curved nodes and edges are highly related to graph connectivity, and removing them would result in a disconnected graph (Ni et al., 2015). As we add more edges, the FREQ graph becomes much more connected, therefore negatively-curved edges play a less important role. Nonetheless, given that the Ollivier-Ricci curvature accounts not only for connections but also for distances on the weighted graph[3], the overall curvature shown by the vast majority of nodes and edges remains negative. The correspondence with the negative curvature of hyperbolic space suggests that the FREQ graph would profit from a representation in that geometry, rather than in an Euclidean one.

6.2.4 Embedding the Hierarchies

Since the goal of this chapter is to study how to derive hierarchical information and its incorporation into different models, we do not focus in developing effective ways to learn graph embeddings. We cover this topic in following chapters. To embed the target type representations into the different metric spaces we make use of the library *Hype*.[4] This library allows us to embed graphs into low-dimensional continuous spaces with different metrics by implementing Poincaré embeddings (Nickel & Kiela, 2017), Lorentz embeddings (Nickel & Kiela, 2018) (both explained in §5.1.3) and the same algorithm also for Euclidean embeddings. The method aims at keeping related objects close to each other in the space. The learned embeddings capture notions of both similarity, through the relative distance among each other, and hierarchy, through the distance to the origin, *i.e.* the norm.

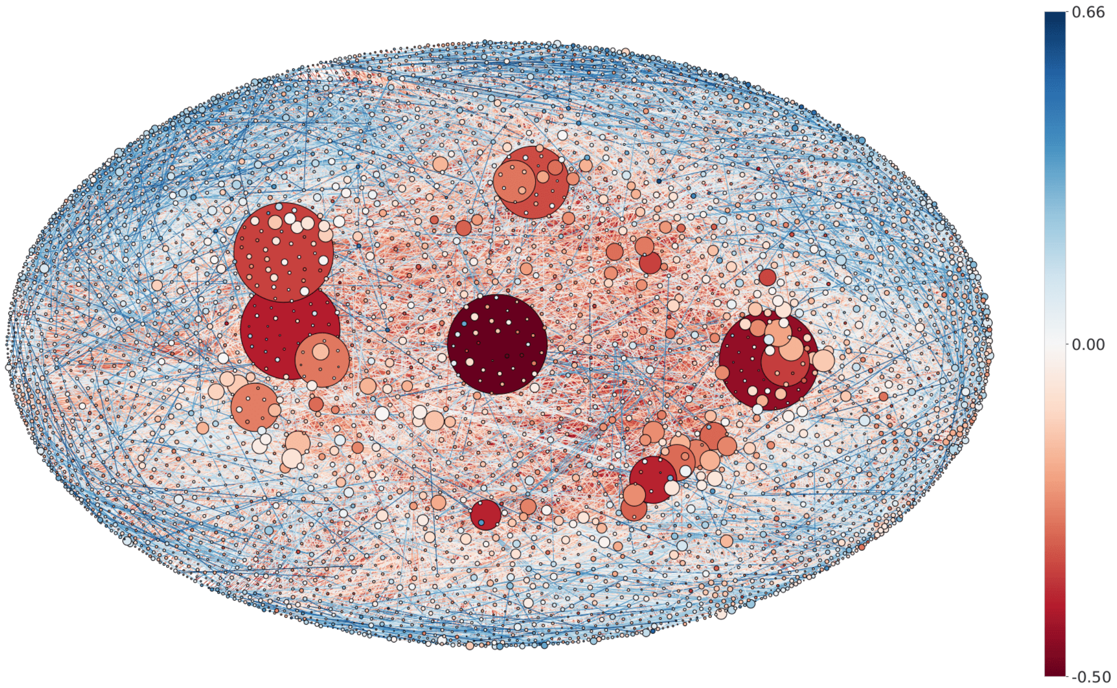
In this work we choose to compare the representation capacity of the Poincaré model of hyperbolic space ($\mathbb{D}$) and the Euclidean space ($\mathbb{R}$). The projection of the hierarchy derived from WordNet is depicted in Figure 6.8.

6.3 Model

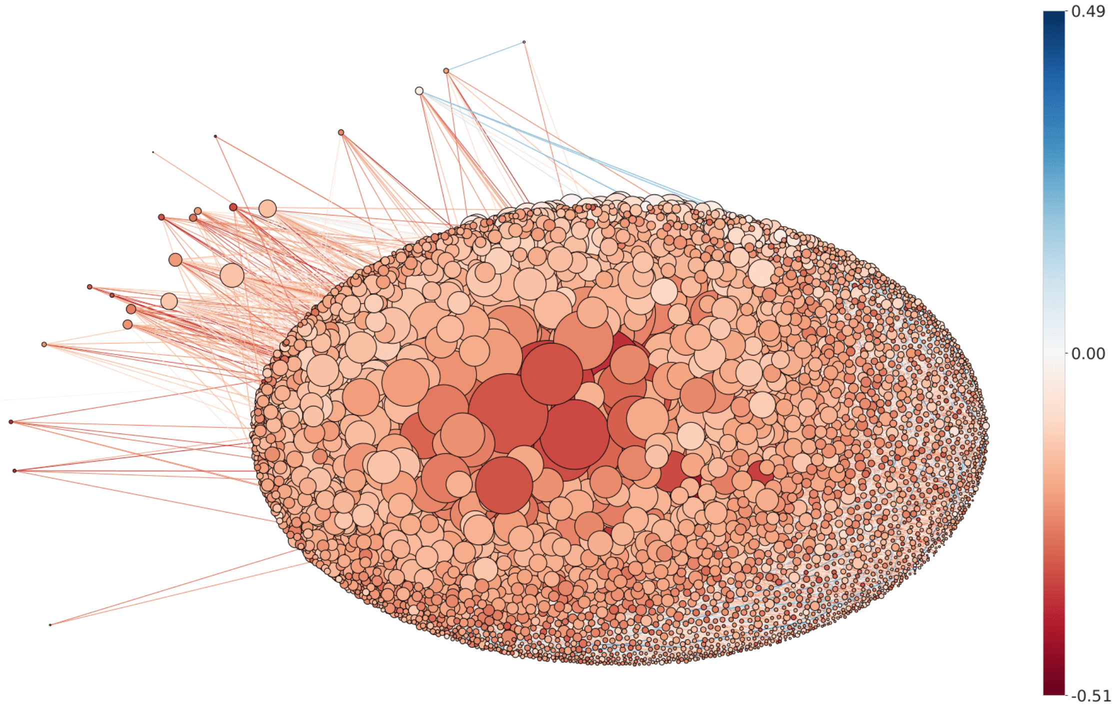
Given the encoded feature representations of a mention m and its context c, noted as $e(m, c) \in \mathbb{R}^{n'}$ our goal is to learn a mapping function $f : \mathbb{R}^{n'} \rightarrow \mathbb{S}^n$, where $\mathbb{S}^n$ is the target vector space. We intend to approximate embeddings of the type labels t_m, previously

[3]Note that the weights in the graph are given by the frequency of the types co-occurring, so we could say that they reflect *similarity*. To obtain a distance metric we apply the transformation $distance = {}^{1}\!/_{similarity}$.

[4]https://github.com/facebookresearch/poincare-embeddings/



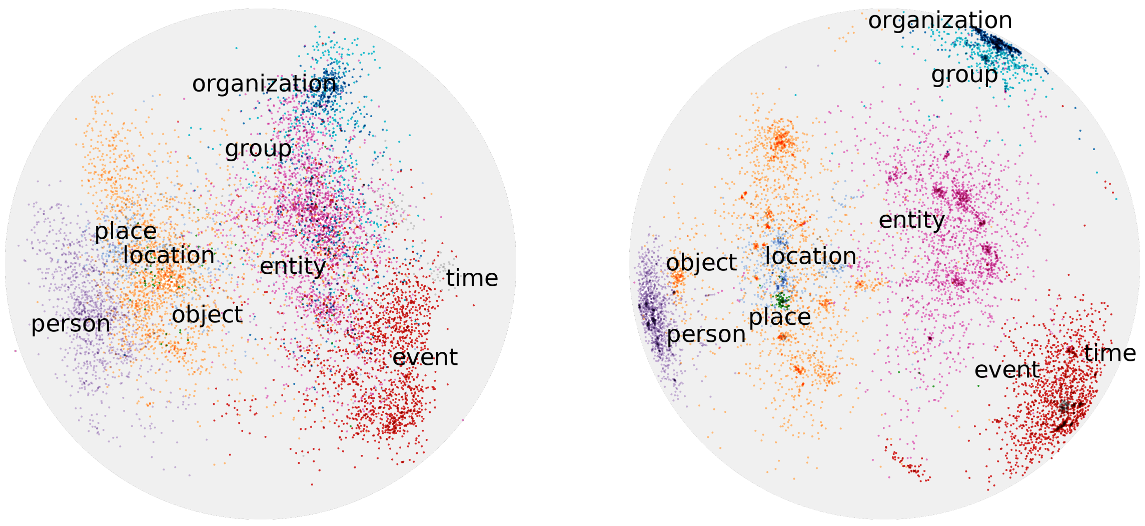
(a) WORDNET alignment.



(b) FREQ: Frequency-based type co-occurrences.

Figure 6.7: Visualization of Ollivier-Ricci curvature for different graphs from the UltraFine dataset. The size of the nodes reflects the degree.

projected into the space. Subsequently, we perform a search of the nearest type embeddings of the embedded representation in order to assign the categorical label corresponding to the mention within that context. Figure 6.9 presents an overview of the model.



(a) Euclidean Space. (b) Hyperbolic Space.

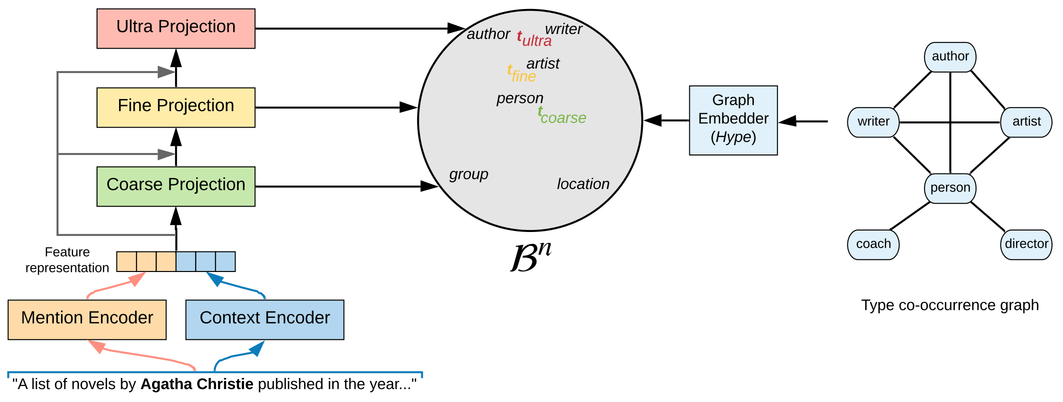
Figure 6.8: Type inventory of the Ultra-Fine dataset aligned to the WordNet noun hierarchy and projected on two dimensions in different spaces. We can see how the types are better clustered in the hyperbolic space. Moreover, in this space, types that are high in the hierarchy, such as `entity`, are placed close to the center of the space, whereas more fine-grained types and leaf nodes are located closer to the boundary of the disk.

The label distribution on the dataset is diverse and fine-grained. Each instance is annotated with three levels of granularity, namely *coarse*, *fine* and *ultra-fine*, and on the development and test set there are, on average, five labels per item. This poses a challenging problem for learning and predicting with only one projection. As a solution, we propose three different projection functions, f_{coarse}, f_{fine}, and f_{ultra}, each one of them fine-tuned to predict labels of a specific granularity.

We hypothesize that the complexity of the projection increases as the granularity becomes *finer*, given that the target label space per granularity increases. Inspired by Sanh et al. (2019), we arrange the three projections in a hierarchical manner that reflects these difficulties. The *coarse* projection task is set at the bottom layer of the model and more complex (*finer*) interactions at higher layers. With the projected embedding of each layer, we aim to introduce an inductive bias in the next projection that will help to guide it into the correct region of the space. Nevertheless, we use shortcut connections so that top layers can have access to the encoder layer representation.

6.3.1 Mention and Context Representations

To encode the context c containing the mention m, we apply the encoder schema of Choi et al. (2018) based on Shimaoka et al. (2016). Given a sentence of tokens $x_1, ..., x_n$, Choi et al. (2018) concatenates an additional location embedding l_i to the pre-trained GloVe



(a) Projection layers. (b) Incorporation of hierarchical information.

Figure 6.9: Overview of the proposed model to predict types of a mention within its context.

(Pennington et al., 2014) word embedding w_i of each word, which indicates whether x_i is before, inside, or after the mention. This technique drives the attention layer to focus excessively on the mention m. Since in most datasets the mentions are named entities, they provide important information to recognize the entity types. However, the dataset we are interested in contains a large quantity of entity mentions which are pronouns, thus the prediction should be based on to the context. We replace the location embedding for a word position embedding p_i to reflect relative distances between the i-th word and the entity mention. This modification induces a bias on the attention layer to focus less on the mention and more on the context. As a result of this, we represent each token in the context sentence by $[w_i; p_i]$. Finally we apply a standard Bi-LSTM (*Long short-term memory*, Hochreiter & Schmidhuber (1997)) and a self-attentive encoder (McCann et al., 2017) on top to get the context representation $C \in \mathbb{R}^{d_c}$.

For the mention representation we derive features from a character-level CNN, concatenate them with the Glove word embeddings of the mention, and combine them with a similar self-attentive encoder. The mention representation is denoted as $M \in \mathbb{R}^{d_m}$. The final representation is achieved by the concatenation of mention and context $[M; C] \in \mathbb{R}^{d_m+d_c}$.

6.3.2 Projecting into the Ball

To learn a projection function that embeds our feature representation in the target space, we apply a variation of the re-parameterization technique introduced in Dhingra et al. (2018). The re-parameterization involves computing a direction vector r and a norm magnitude λ

from $e(m, c)$ as follows:

$$\begin{aligned} \overline{r} &= \varphi_{dir}(e(m,c)), \quad r = \frac{\overline{r}}{\|\overline{r}\|}, \\ \overline{\lambda} &= \varphi_{norm}(e(m,c)), \quad \lambda = \sigma(\overline{\lambda}), \end{aligned} \tag{6.1}$$

where $\varphi_{dir} : \mathbb{R}^{n'} \rightarrow \mathbb{R}^n$, $\varphi_{norm} : \mathbb{R}^{n'} \rightarrow \mathbb{R}$ can be arbitrary functions, whose parameters will be optimized during training, and σ is the sigmoid function that ensures the resulting norm $\lambda \in (0, 1)$. The re-parameterized embedding is defined as $v = \lambda r$, which lies in $\mathbb{S}^n$.

By making use of this simple technique, the embeddings are guaranteed to lie in the unit ball (Poincaré ball or Euclidean). This avoids the need to correct the gradient or the utilization of Riemannian SGD (Bonnabel, 2011). Instead, it allows the use of any optimization method in deep learning, such as Adam (Kingma & Ba, 2014).

We parameterize the direction function $\varphi_{dir} : \mathbb{R}^{d_m+d_c} \rightarrow \mathbb{R}^n$ as a multi-layer perceptron (MLP) with a single hidden layer, using rectified linear units (ReLU) as nonlinearity, and dropout. We do not apply the ReLU function after the output layer in order to allow negative values as components of the direction vector. For the norm magnitude function $\varphi_{norm} : \mathbb{R}^{d_m+d_c} \rightarrow \mathbb{R}$ we use a single linear layer.

6.3.3 Optimization of the Model

We aim to find projection functions f_i that embed the instance representations closer to the respective target types, in a given vector space $\mathbb{S}^n$. As target space $\mathbb{S}^n$ we use the Poincaré Ball $\mathbb{D}^n$ and compare it with the Euclidean unit ball $\mathbb{R}^n$. Both $\mathbb{D}^n$ and $\mathbb{R}^n$ are metric spaces, therefore they are equipped with a distance function, namely the hyperbolic distance $d_{\mathbb{D}}$ defined in Equation 2.2, and the Euclidean distance $d_{\mathbb{R}}$ respectively, which we intend to minimize. Moreover, since the Poincaré Model is a conformal model of the hyperbolic space (see Equation 2.3), *i.e.* the angles between Euclidean and hyperbolic vectors are equal, the cosine distance $d_{\cos}$ can be used, as well.

We propose to minimize a combination of the distance defined by each metric space and the cosine distance to approximate the embeddings. Although formally this is not a distance metric since it does not satisfy the Cauchy-Schwarz inequality, it provides a very strong signal to approximate the target embeddings accounting for the main concepts modeled in the representation: *relatedness*, captured via the distance and orientation in the space, and *generality*, via the norm of the embeddings.

To mitigate the instability in the derivative of the hyperbolic distance[5] we follow the approach proposed in Sala et al. (2018) and minimize the square of the distance, which does have a continuous derivative in $\mathbb{D}^n$. Thus, in the Poincaré Model we minimize the

[5]$\lim_{y \rightarrow x} \partial_x |d_H(x, y)| \rightarrow \infty \; \forall x \in \mathbb{D}^n$

distance for two points $u, v \in \mathbb{D}^n$ defined as:

$$d_{\mathbb{H}}(u, v) = \alpha(d_{\mathbb{D}}(u, v))^2 + \beta d_{\cos}(u, v) \tag{6.2}$$

Whereas in the Euclidean space, for $x, y \in \mathbb{R}^n$ we minimize:

$$d_{\mathbb{E}}(x, y) = \alpha d_{\mathbb{R}}(x, y) + \beta d_{\cos}(x, y) \tag{6.3}$$

The hyperparameters α and β are added to compensate the bounded image of the cosine distance function in $[0, 1]$.

6.4 Experiments

We perform experiments on the Ultra-Fine Choi et al. (2018) and OntoNotes Gillick et al. (2014) datasets to evaluate which kind of hierarchical information is better suited for entity typing, and under which geometry the hierarchy can be better exploited.[6]

6.4.1 Setup

For evaluation we run experiments on the Ultra-Fine dataset with our model projecting onto the hyperbolic space, and compare to the same setting in Euclidean space. The type embeddings are created based on the following hierarchical structures derived from the dataset: the type vocabulary aligned to the WordNet hierarchy (WORDNET), type co-occurrence frequency (FREQ), *pointwise mutual information* among types (PMI), and finally, the combination of WordNet's transitive closure of each type with the co-occurrence frequency graph (WORDNET + FREQ).

As baseline, we compare our model to the multi-task model of Choi et al. (2018) trained on the open-source version of their dataset (MULTITASK). Our final type predictions consist of the nearest neighbor from the *coarse* and *fine* projections, and the three nearest neighbors from the *ultra-fine* projection. Thus, for every instance we always predict **five** candidate labels. We report Loose Macro-averaged and Loose Micro-averaged F1 metrics computed from the precision/recall scores over the same three granularities established by Choi et al. (2018). For all models we optimize Macro F1 on *coarse* types on the dev set, and evaluate on the test set. All experiments project onto a target space of 10 dimensions.

[6]Code available at: `https://github.com/nlpAThits/figet-hyperbolic-space`

Model	**Space**	**Coarse**		**Fine**		**Ultra-fine**		**Coarse + Ultra**		**Variation**	
		MaF1	**MiF1**	**MaF1**	**MiF1**	**MaF1**	**MiF1**	**MaF1**	**MiF1**	**MaF1**	**MiF1**
MULTITASK	-	60.6	58.0	37.8	34.7	13.6	11.7	-	-	-	-
WORDNET	$\mathbb{D}$	45.9	44.3	22.5	21.5	7.0	6.7	41.8	37.2	-4.1	-7.1
	$\mathbb{R}$	56.1	54.2	26.6	25.3	7.2	6.5	56.6	48.5	0.6	-5.7
WORDNET + FREQ	$\mathbb{D}$	54.6	52.8	18.4	18.0	11.3	10.8	46.5	40.6	-8.0	-12.2
	$\mathbb{R}$	**56.7**	**54.9**	**27.3**	**26.0**	12.1	11.5	55.8	49.1	-0.9	-5.8
FREQ	$\mathbb{D}$	56.5	54.6	26.8	25.7	**16.0**	15.2	59.7	**53.5**	3.2	**-1.1**
	$\mathbb{R}$	56.1	54.2	25.8	24.4	12.1	11.4	**60.0**	53.0	**3.9**	-1.3
PMI	$\mathbb{D}$	54.7	53.0	26.9	25.8	**16.0**	**15.4**	57.5	51.8	2.8	-1.2
	$\mathbb{R}$	56.5	54.6	26.9	25.6	12.2	11.5	59.7	53.0	3.2	-1.5

(a) Results on the same three granularities analyzed by Choi et al. (2018).

(b) Comparison to previous *coarse* results.

Table 6.4: Results on the test set for different hierarchies and spaces. The best results of our models are marked in bold. On (b) we report the comparison of adding the closest *coarse* label to the *ultra-fine* prediction, with respect to the *coarse* results on (a).

6.5 Results and Discussion

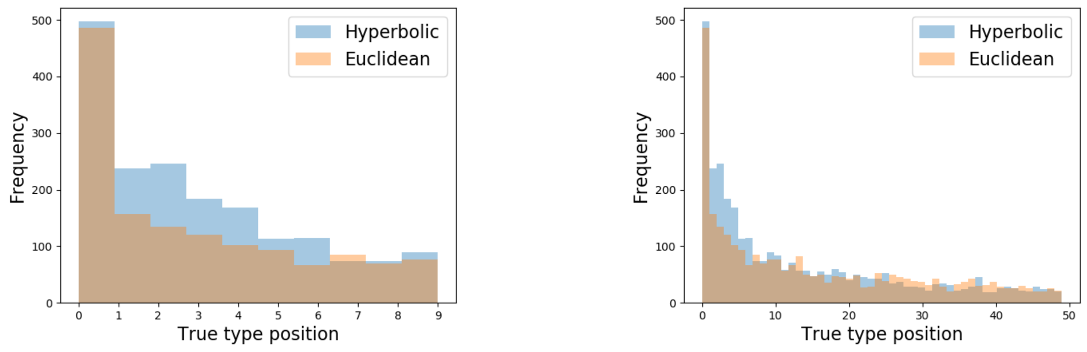
6.5.1 Comparison of the Hierarchies

Results on the test set are reported in Table 6.4. From comparing the different strategies to derive the hierarchies, we can see that FREQ and PMI substantially outperform MULTITASK on the *ultra-fine* granularity (17.5% and 29.8% relative improvement in Macro F1 and Micro F1, respectively, with the hyperbolic model). Both hierarchies show a substantially better performance over the WORDNET hierarchy on this granularity as well (MaF1 16.0 and MiF1 15.4 for PMI vs 7.0 and 6.7 for WORDNET on the Hyperbolic model), indicating that these structures, created solely from the dataset statistics, better reflect the type distribution in the annotations. On FREQ and PMI, types that frequently co-occur on the training set are located closer to each other, improving the prediction based on nearest neighbor.

All the hierarchies show very low performance on *fine* when compared to the MULTITASK model. This exhibits a weakness of our regression setup. On the test set there are 1,998 instances but only 1,318 *fine* labels as ground truth (see Table 6.2). By forcing a prediction on the *fine* level for all instances, precision decreases notably. More details in Section 6.5.3.

The combined hierarchy WORDNET + FREQ achieves marginal improvements on *coarse* and *fine* granularities, while it degrades the performance on *ultra-fine* when compared to FREQ.

By imposing a hierarchical structure over the type vocabulary we can infer types that are located higher up in the hierarchy from the predictions of the lower ones. To analyze



(a) Top 10. (b) Top 50.

Figure 6.10: Histogram of ground-truth type neighbor positions for *ultra-fine* predictions in Hyperbolic and Euclidean spaces on the test set.

this, we add the closest *coarse* label to the *ultra-fine* prediction of each instance. Results are reported in Table 6.4b. The improvements are noticeable on the Macro score (up to 3.9 F1 points difference on FREQ) whereas Micro decreases. Since we are adding types to the prediction, this technique improves recall and penalizes precision. Macro is computed on the entity level, while Micro provides an overall score, showing that per instance the prediction tends to be better. The improvements can be observed on FREQ and PMI given that their predictions over *ultra-fine* types are better.

6.5.2 Comparison of the Spaces

When comparing performances with respect to the metric spaces, the hyperbolic models for PMI and FREQ outperform all other models on *ultra-fine* granularity. Compared to its Euclidean counterpart, PMI brings considerable improvements (16.0 vs 12.2 and 15.4 vs 11.5 for Macro and Micro F1 respectively). The reason for this is that, as already explained, types that co-occur more often on the training set are placed closer together in the embedding space. Moreover, *ultra-fine* types tend to be lower in the hierarchy, therefore they are located near the boundary of the Poincaré ball. Since in the hyperbolic space the amount of space grows exponential with the norm, in that region it becomes easier for the model to separate the type labels and perform classification. Figure 6.10 shows a histogram of the distribution of ground-truth types as nearest neighbors to the prediction.

On both Euclidean and hyperbolic models, the type embeddings for *coarse* and *fine* labels are located closer to the origin of the space. In this region, the spaces show a much more similar behavior in terms of the distance calculation, and this similarity is reflected on the results as well.

The low performance of the hyperbolic model of WORDNET on *coarse* can be explained by the fact that `entity` is the root node of the hierarchy, therefore it is located closer to the center of the space. Elements placed in the vicinity of the origin have a norm closer to zero, thus their distance to other types tends to be shorter (does not grow exponentially). This often misleads the model into assign `entity` as the *coarse*. See Table 6.5c for an example. This issue is alleviated on WORDNET + FREQ. Nevertheless, it appears again when using the *ultra-fine* prediction to infer the *coarse* label. The drop in performance can be seen in Table 6.4b: Macro F1 decreases by 8.0 and Micro F1 by 12.2.

6.5.3 Error Analysis

" There is nothing worse than a model doing the right thing for the wrong reasons."
– Anette Frank
Computational Linguistic Colloquium, 29.01.2019

We perform an error analysis on samples from the development set and predictions from two of our proposed hyperbolic models. We show three examples in Table 6.5. Overall we can see that predictions are reasonable, suggesting synonyms or related words, although some of them are considered errors for not being part of the annotations.

In the proposed regression setup, we predict a fixed amount of labels per instance. This

a) Example	Rin, Kohaku and Sesshomaru Rin befriends Kohaku, the demonslayer Sango's younger brother, while Kohaku acts as her guard when Naraku is using her for bait to lure Sesshomaru into **battle**.
Annotation	*event, conflict, war, fight, battle, struggle, dispute, group_action*
Prediction	FREQ: event, conflict, war, fight, battle;
	WORDNET: event, conflict, difference, engagement, assault
b) Example	**The UN mission in Afghanistan** dispatched its own investigation, expressing concern about reports of civilian casualties and calling for them to be properly cared for.
Annotation	*organization, team, mission*
Prediction	FREQ: organization, team, mission, activity, operation;
	WORDNET: group, institution, branch, delegation, corporation
c) Example	Brazilian President Luiz Inacio Lula da Silva and Turkish Prime Minister Recep Tayyip Erdogan talked about designing a strategy different from sanctions at **a meeting** Monday, Amorim said.
Annotation	*event, meeting, conference, gathering, summit*
Prediction	FREQ: event, meeting, conference, film, campaign;
	WORDNET: entity, meeting, gathering, structure, court

Table 6.5: Qualitative analysis of instances taken from the development set. The predictions are generated with the hyperbolic models of FREQ and WORDNET. The mention is in **bold** and correct predictions are marked in blue color.

Model	Dev			Test		
	P	R	F1	P	R	F1
ATTNER	**53.7**	15.0	23.5	**54.2**	15.2	23.7
FREQ	24.8	**25.9**	25.4	25.6	**26.8**	26.2
MULTI	48.1	23.2	**31.3**	47.1	24.2	**32.0**

Table 6.6: Combined performance over the three granularities. Results are extracted from Choi et al. (2018).

schema has drawbacks as shown in example a), where all predicted types by the FREQ model are correct though we can not predict more, and b), where we predict more related types that are not part of the annotations.

In examples b) and c) we see how the FREQ model predicts the *coarse* type correctly whereas the model that uses the WordNet hierarchy predicts `group` and `entity` since these labels are considered more general (`organization` *is-a* `group`) thus located closer to the origin of the space.

To analyse precision and recall more accurately, we compare our model to the one of Shimaoka et al. (2016) (ATTNER) and the multi-task model of Choi et al. (2018) (MULTI). We show the results for macro-averaged metrics in Table 6.6. Our model is able to achieve higher recall but lower precision. Nonetheless we are able to outperform ATTNER with a regression model even though they apply a classifier to the task.

6.5.4 Analysis Case: OntoNotes

To better understand the effects of the hierarchy and the metric spaces we also perform an evaluation on OntoNotes (Gillick et al., 2014). We compare the original hierarchy of the dataset (ONTO), and one derived from the type co-occurrence frequency extracted from the data augmented by Choi et al. (2018) with this type inventory. The results for the three granularities are presented in Table 6.7.

The FREQ model on the hyperbolic geometry achieves the best performance for the *ultra-fine* granularity, in accordance with the results on the Ultra-Fine dataset. In this case

Model	Sp	Coarse		Fine		Ultra	
		Ma	Mi	Ma	Mi	Ma	Mi
ONTO	$\mathbb{D}$	**83.0**	81.9	24.0	23.9	2.0	2.0
	$\mathbb{R}$	82.2	**82.2**	28.8	28.7	2.4	2.4
FREQ	$\mathbb{D}$	81.7	81.8	27.1	27.1	**4.2**	**4.2**
	$\mathbb{R}$	81.7	81.7	**30.6**	**30.6**	3.8	3.8

Table 6.7: Macro and micro F1 results on OntoNotes.

the improvements of the frequency-based hierarchy are not so remarkable when compared to the ONTO model given that the type inventory is much smaller, and the annotations follow a hierarchy where there is only one possible path for every label to its *coarse* type.

The low results on the *ultra-fine* granularity are due to the reduced multiplicity of the annotated types (See Table 6.2). Most instances have only one or two types, setting very restrictive upper bounds for this setup.

6.6 Conclusions

There has been a remarkable increase in the size of types inventories for fine-grained entity typing. Hence, incorporating the hierarchical information present in the inventories into the models has become critical to improve performance. Hyperbolic spaces are a natural fit to model hierarchical structures, therefore they arise as a prominent candidate to contribute in this regard. They have been applied mostly on complex and social networks modeling (Krioukov et al., 2010; Verbeek & Suri, 2016). In the field of Natural Language Processing, they have been employed to learn embeddings for Question Answering (Tay et al., 2018), in Neural Machine Translation (Gulcehre et al., 2019), and to model language (Leimeister & Wilson, 2018; Tifrea et al., 2019).

In our case, we pose the entity typing task as a graph embedding problem combined with a nearest neighbor classifier. To do so, we derive expert-generated and data-driven hierarchies from the type inventories. We model the hierarchies with graphs, and analyze diverse geometrical properties under the lens of graph theory. This analysis justifies the choice of the hyperbolic space as a suitable metric space to embed the data. We build upon the work of Nickel & Kiela (2017) on modeling hierarchical link structure of symbolic data by learning graph embeddings, and adapt it with the parameterization method proposed by Dhingra et al. (2018) to cope with feature representations of text. Experiments on two different datasets show consistent improvements of hyperbolic embedding over Euclidean baselines on very fine-grained labels when the hierarchy reflects the annotated type distribution.

The improved results of the data-driven approach suggest the advantages of learning from the data instead of imposing a external taxonomy. Nonetheless, in the work presented in this chapter, an explicit definition of the graph to be embedded is required before projecting the mentions with their context. In the next chapter, we devise ways to perform this task (extensible also to many others), in an end-to-end setup, without the need to derive the hierarchy in advance.

Chapter 7

Inferring the Hierarchy with a Fully Hyperbolic Model

"The road itself tells us far more than signs do."
– Tom Vanderbilt

In the previous chapter we have devised explicit means to organize large label inventories into hierarchical graphs. Moreover, noticing a perfect match between hierarchical label inventories in the task at hand and the benefits of hyperbolic spaces, we endowed a classification model with a suitable geometry to capture this fundamental property of the data distribution. We showed that downstream tasks profit from the geometric inductive bias given by representing hierarchies in hyperbolic space. Nevertheless, the proposed approach suffers from different drawbacks. First, we have to provide the model with the label inventory arranged as an explicit predefined hierarchical graph. Experiments demonstrated that the choice of this hierarchy has a significant impact on the results. Second, the k-nearest neighbor classifier imposes strong upper bounds to the performance of the model given the varying number of types per instance. And third, the integration of hyperbolic neural components into Euclidean pipelines remains unclear.

In this chapter, we deepen our study in different directions, in order to tackle all the aforementioned issues. We propose a fully hyperbolic neural model for hierarchical multi-class multi-label classification. The proposed approach is able to automatically infer the latent hierarchy arising from the class distribution, instead of imposing a predefined one. By virtue of the hyperbolic representations, the model achieves a meaningful and interpretable organization of the label space. This arrangement captures implicit relations in the inventory and enables the model to excel at fine-grained classification.

Recent work has proposed hyperbolic neural components, such as word embeddings (Dhingra et al., 2018; Tifrea et al., 2019), recurrent neural networks (Ganea et al., 2018b), attention layers (Gulcehre et al., 2019) and classifiers (Cho et al., 2019). However, researchers have incorporated these isolated components into neural models, whereas the rest of the layers and algorithms operate under Euclidean assumptions. This impedes models from fully exploiting the properties of hyperbolic geometry. Furthermore, there are different analytic models of hyperbolic space, and not all previous work operates in the same one, which hinders their combination, and hampers their adoption for downstream tasks (e.g. Tifrea et al. (2019) learn embeddings in the Poincaré model, Gulcehre et al. (2019) aggregate points in the Klein model, or Nickel & Kiela (2018) perform optimization in the Lorentz model). We address these issues. Our model encodes textual inputs, applies a novel attention mechanism, and performs multi-class multi-label classification, executing all operations in the Poincaré model of hyperbolic space.

The model is proposed in a generic fashion such that it can be applied on different hierarchical classification problems with sequential data as input. Since it is particularly well-suited for large inventories that exhibit a hierarchical structure due to the geometric inductive bias provided by the hyperbolic representation, we apply it on the concrete task of Entity typing. We evaluate the model on the same two datasets from the previous chapter, namely UltraFine (Choi et al., 2018) and OntoNotes (Gillick et al., 2014), and compare to Euclidean baselines as well as to state-of-the-art methods for the task (Xiong et al., 2019; Onoe & Durrett, 2019). The hyperbolic system has competitive performance when compared to an ELMo model (Peters et al., 2018) and a BERT model (Devlin et al., 2019) on very fine-grained types, with remarkable reduction of the parameter size. Instead of relying on large pre-trained models, we impose a suitable geometric inductive bias by choosing an adequate metric space to embed the data, which does not introduce extra burden on the parameter footprint.

Finally, by means of the exponential and logarithmic maps (explained in §2.6.1) we are able to mix hyperbolic and Euclidean components into one model, aiming to exploit their strengths at different levels of the representation. We perform a thorough ablation that allows us to understand the impact of each hyperbolic component in the final performance of the system, and showcases its ease of integration with Euclidean layers.

7.1 Label Embeddings as Graph Embeddings

Label inventories for multi-class multi-label classification have grown in size and complexity (Del Corro et al., 2015; Choi et al., 2018). In this setup, labels are not mutually exclusive, therefore exploiting inter-label correlations becomes critical to improve performance. Large inventories tend to exhibit a hierarchical structure, either by an explicit tree-like arrangement of the labels with *coarse* labels at the top, *fine-grained* at the bottom, (such as in TypeNet, see Figure 6.1, or in OntoNotes, Figure 6.4), or implicitly through

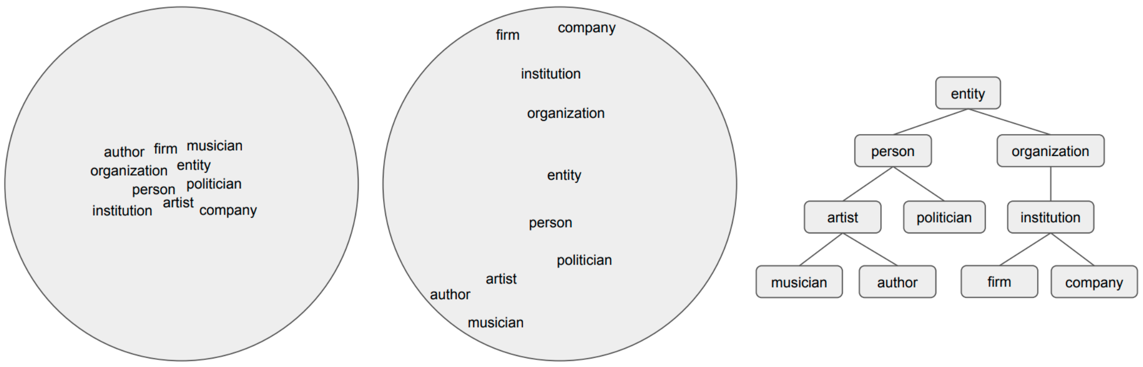


Figure 7.1: Left: initially the label embeddings are randomly initialized near the origin of the hyperbolic space. Center: after the model converges, the arrangement of the label embeddings in the space reflects semantic similarity through their relative distance, and hierarchy through the norm. Right: the hierarchical graph that we can derive from the label embeddings, which reflects implicit hyponymic relations in the inventory.

the label distribution in the dataset, where *coarse* labels appear more frequently than *fine-grained* ones (such as in UltraFine).

Prior work has integrated only explicit hierarchical information by formulating a hierarchy-aware loss (Murty et al., 2018; Xu & Barbosa, 2018) or by representing instances and labels in a joint Euclidean embedding space (Shimaoka et al., 2017; Abhishek et al., 2017). However, the resulting space is hard to interpret, and these methods fail to capture implicit relations in the label inventory.

Instead of conditioning the model with an explicit hierarchical arrangement, we propose to infer this information from the class distribution in the dataset while jointly performing classification in a hyperbolic space. As the system is trained, it mines co-occurrences of the labels relevant for the task of classifying. During this process, the final layer of the model is implicitly learning an embedding for each label (more details on this in §7.3.5). The resulting embedding space is meaningful and interpretable thanks to the self-organizing properties of the hyperbolic geometry (Ontrup & Ritter, 2002). The final arrangement automatically captures implicit hyponymic relations (*is-a*) in the inventory without requiring additional annotated data.

Conceptually, this is equivalent to learning graph embeddings for the nodes of a graph whose edges are unknown. Initially, we only have the labels, but we are not aware of how they relate to each other. As we train the model, it accommodates the node embeddings in the space in way that it reflects semantic similarity, through the relative distance among each other, and hierarchy, through the distance to the origin. The relationships between the labels arise from the distribution in the dataset, and the objective of the classification task. Finally, from the node embeddings we can reconstruct the induced graph. A diagram of this process is depicted in Figure 7.1.

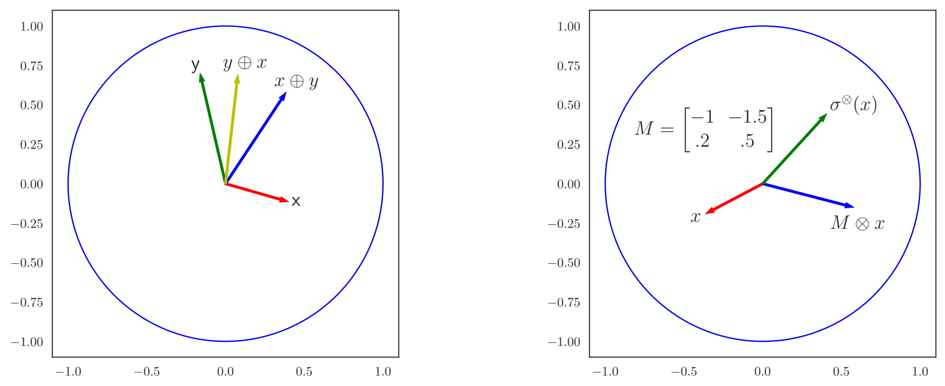


Figure 7.2: Visualization of Möbius operations. Left: Möbius addition (noncommutative). Right: Matrix-vector multiplication and pointwise non-linearity.

7.2 Hyperbolic Neural Networks

In this section we briefly review the necessary background on hyperbolic neural components. The terminology and formulas used here follow the formalism of Möbius gyrovector spaces introduced in §2.6, and the definitions of hyperbolic neural components of Ganea et al. (2018b). Throughout this chapter we work in the Poincaré model of hyperbolic space, as defined in §2.3.1, where $\lambda_x := 2/1-\|x\|^2$ is called the *conformal factor*.

In order to implement the equivalent of feed-forward neural networks (FFNN) in hyperbolic space we need to translate the operations of matrix-vector multiplication, bias addition, and the application of pointwise non-linearities. For bias addition we utilize the formula of Möbious addition defined in Equation 2.7. In the following we provide the definition for the two operations remaining.

Möbius matrix-vector multiplication: Given a linear map $M : \mathbb{R}^n \to \mathbb{R}^m$, which we identify with its matrix representation, and a point $x \in \mathbb{D}^n$, $Mx \neq 0$, it is defined as:

$$M \otimes x = \tanh\left(\frac{\|Mx\|}{\|x\|}\tanh^{-1}(\|x\|)\right)\frac{Mx}{\|Mx\|} \tag{7.1}$$

Pointwise non-linearity: If we model it as $\varphi : \mathbb{R}^n \to \mathbb{R}^n$, then its Möbius version $\varphi^{\otimes}$ can be applied using the same formulation of the matrix-vector multiplication. A visualization of the aforementioned operations can be seen in Figure 7.2.

By combining these operations we obtain a one-layer feed-forward neural network in hyperbolic space, described as

$$y = \varphi^{\otimes}(M \otimes x \oplus b)$$

with $M \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{D}^m$ as trainable parameters. Note that the parameter b lies in the

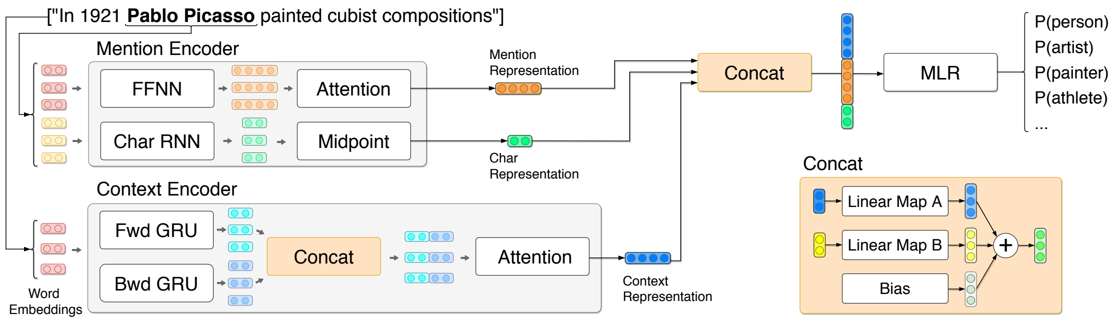


Figure 7.3: Overview of the proposed model. The mention encoder extracts word and char-level entity representations. The context encoder is based on a bidirectional-GRU with attention. The outputs of both encoders are concatenated and passed to a classifier based on a multinomial logistic regression.

hyperbolic space, thus its updates during training need to be corrected for this geometry.

Finally, we also make use of the exponential and logarithmic maps to map points in the hyperbolic space to the Euclidean space, and vice-versa. The definition of these operations at the origin of the space can be found in §2.6.1.

7.3 Fully Hyperbolic Classification Model

In this section we propose a general hyperbolic neural model for classification with sequential data as input. The building blocks are defined in a generic manner such that they can be applied to different tasks, or integrated with regular Euclidean layers. Our proposed architecture resembles recent neural models applied to entity typing (Choi et al., 2018). For the encoders we employ the neural networks introduced in Ganea et al. (2018b), we propose a novel attention mechanism operating entirely in the Poincaré model, and we extend the hyperbolic classifier to multi-class multi-label setups. An overview of the model can be seen in Figure 7.3.

7.3.1 Mention Encoder

To represent the mention, we combine word and char-level features, similar to Lee et al. (2017). Given a sequence of k tokens in a mention span, we represent them using pre-trained word embeddings $w_i \in \mathbb{D}^n$ which we assume to lie in hyperbolic space. We apply a hyperbolic FFNN, described as:

$$m_i = \tanh^{\otimes}(W^M \otimes w_i \oplus b^M) \tag{7.2}$$

with $m_i \in \mathbb{D}^{d_M}$, and where $W^M \in \mathbb{R}^{d_M \times n}, b^M \in \mathbb{D}^{d_M}$ are parameters of the model. We combine the resulting $m_1, ..., m_k$ into a single mention representation $\mathbf{m} \in \mathbb{D}^{d_M}$ by computing a weighted sum of the token representations in hyperbolic space with the attention mechanism explained in §7.3.4.

Moreover, we extract features from the sequence of characters in the mention span with a recurrent neural network (RNN) (Lample et al., 2016). We represent each character with a char-embedding $c_i \in \mathbb{D}^{d_C}$ that we train in the Poincaré ball. An RNN operating in hyperbolic space is defined by:

$$h_{t+1} = \varphi^{\otimes}(W^C \otimes h_t \oplus U^C \otimes c_t \oplus b^C) \tag{7.3}$$

where $W^C, U^C \in \mathbb{R}^{d_C \times d_C}, b^C, h_t \in \mathbb{D}^{d_C}$, and φ is a pointwise non-linearity function. Finally, we obtain a single representation $\mathbf{c} \in \mathbb{D}^{d_C}$ by taking the midpoint of the states h_i using Equation 7.7.

7.3.2 Context Encoder

To encode the context we apply a hyperbolic version of gated recurrent units (GRU) (Cho et al., 2014) proposed in Ganea et al. (2018b).[1]

$$\begin{aligned}
r_t &= \sigma\left(\log_0(W^r \otimes h_{t-1} \oplus U^r \otimes x_t \oplus b^r)\right) \\
z_t &= \sigma\left(\log_0(W^z \otimes h_{t-1} \oplus U^z \otimes x_t \oplus b^z)\right) \\
\tilde{h}_t &= \tanh^{\otimes}((W \operatorname{diag}(r_t)) \otimes h_{t-1} \oplus U \otimes x_t \oplus b) \\
h_t &= h_{t-1} \oplus \operatorname{diag}(z_t) \otimes (-h_{t-1} \oplus \tilde{h}_t)
\end{aligned} \tag{7.4}$$

where $W \in \mathbb{R}^{d_S \times d_S}, U \in \mathbb{R}^{d_S \times n}, x_t \in \mathbb{D}^n$ and $b \in \mathbb{D}^{d_S}$ (superscripts are omitted). r_t is the reset gate, z_t is the update gate, $\operatorname{diag}(x)$ denotes a diagonal matrix with each element of the vector x on its diagonal, and σ is the $\operatorname{sigmoid}$ function.

Given a sequence of l tokens, we represent them with a pre-trained word embedding $w_i \in \mathbb{D}^n$, and apply a forward and backward GRU, producing contextualized representations $\overrightarrow{h_i}, \overleftarrow{h_i} \in \mathbb{D}^{d_S}$ for each token. We concatenate the resulting states into a single embedding $s_i = \operatorname{concat}(\overrightarrow{h_i}, \overleftarrow{h_i})$ (see concat in §7.3.3), where $s_i \in \mathbb{D}^{2d_S}$. Ultimately, we combine $s_1, ..., s_l$ into a single context representation $\mathbf{s} \in \mathbb{D}^{2d_S}$ with the distance-based attention mechanism.

7.3.3 Concatenation

If we model the concatenation of two vectors in the Poincaré ball as appending one to the other, this does not guarantee that the result remains inside the ball. Thus, we apply a generalized version of the concatenation operation. For $x \in \mathbb{D}^k, y \in \mathbb{D}^l$, then

[1]For a complete description of this network see Ganea et al. (2018b) §3.3.

concat : $\mathbb{D}^k \times \mathbb{D}^l \to \mathbb{D}^n$ is defined as:

$$\text{concat}(x, y) = M_1 \otimes x \oplus M_2 \otimes y \oplus b \tag{7.5}$$

where $M_1 \in \mathbb{R}^{n\times k}, M_2 \in \mathbb{R}^{n\times l}, b \in \mathbb{D}^n$ are parameters of the model. In Euclidean architectures, the concatenation of vectors is usually followed by a linear layer, which takes the form of Equation 7.5 when written explicitly.

7.3.4 Distance-based Attention

Previous approaches to hyperbolic attention (Gulcehre et al., 2019; Chami et al., 2019) require mappings of points to different spaces, which hinders their adoption into neural models. We propose a novel attention mechanism (Bahdanau et al., 2015; Vaswani et al., 2017) in the Poincaré model of hyperbolic space. We cast attention as a weighted sum of vectors in this geometry, without requiring any extra mapping of the inputs. In this manner, we make consistent use of the same analytical model of hyperbolic space across all components, which eases their integration.

To obtain the attention weights, we exploit the hyperbolic distance between points (Gulcehre et al., 2019). Given a sequence of states $x_i \in \mathbb{D}^n$, we combine them with a trainable position embedding $p_i \in \mathbb{D}^n$ such that $r_i = x_i \oplus p_i$. We use addition as the standard method to enrich the states with positional information (Vaswani et al., 2017; Devlin et al., 2019). We apply two different linear transformations on r_i to obtain vectors q_i and k_i, both lying in the Poincaré ball. We compute the distance between these two points and finally obtain the weight by applying a softmax over the sequence in the following manner:

$$\begin{gathered} q_i = W^Q \otimes r_i \oplus b^Q, \quad k_i = W^K \otimes r_i \oplus b^K \\ \alpha(q_i, k_i) = \text{softmax}(-\beta d_{\mathbb{D}}(q_i, k_i)) \end{gathered} \tag{7.6}$$

where $W^Q, W^K \in \mathbb{R}^{n\times n}, b^Q, b^K \in \mathbb{D}^n$ and $\beta \in \mathbb{R}$ are parameters of the model. Attention weights will be higher for elements with q_i and k_i vectors placed close to each other.

The positional embeddings are trained along with the model as a hyperbolic parameter. For the context encoder, they reflect relative distances between the i-th word and the entity mention. For the mention encoder, they represent the absolute position of the word inside the mention span.

To aggregate the points as a weighted summation in hyperbolic space we propose to apply the Möbius midpoint, which obeys many of the properties that we expect from a weighted average in Euclidean space (Ungar (2010), Theorem 4.6):

$$m = \frac{1}{2} \otimes \frac{\sum_{i=1}^{n} \alpha_i \gamma(x_i)^2 x_i}{\sum_{i=1}^{n} \alpha_i \left(\gamma(x_i)^2 - \frac{1}{2}\right)} \tag{7.7}$$

where x_i are the states in the sequence, α_i the weights corresponding to each state, and $\gamma(x_i)$ the Lorentz factors.[2] By applying the Möbius midpoint we develop an attention mechanism that operates entirely in the Poincaré model of hyperbolic space.

7.3.5 Classification in the Poincaré Ball

The input of the classifier is the concatenation of mention and context features. To perform multi-class classification in the Poincaré ball, we adapt the generalized multinomial logistic regression (MLR) from Ganea et al. (2018b). Given K classes and $k \in \{1, ..., K\}$, $p_k \in \mathbb{D}^m$, $a_k \in T_{p_k}\mathbb{D}^m \backslash \{0\}$, the formula for the hyperbolic MLR is:

$$p(y = k|x) \propto f\left(\lambda_{p_k}\|a_k\| \sinh^{-1}\left(\frac{2\langle -p_k \oplus x, a_k\rangle}{(1 - \| -p_k \oplus x\|^2)\|a_k\|}\right)\right) \tag{7.8}$$

Where $x \in \mathbb{D}^m$, and p_k and a_k are trainable parameters. It is based on formulating logits as distances to margin hyperplanes. The hyperplanes in hyperbolic space are defined by the union of all geodesics passing through p_k and orthogonal to a_k. Since $a_k \in T_{p_k}\mathbb{D}^m$ and it depends on p_k, it is replaced by $a_k = (\lambda_0/\lambda_{p_k})a'_k$, where $a'_k \in T_{\mathbf{0}}\mathbb{D}^m = \mathbb{R}^m$.[3]

Although this formulation was made for one-label classification, the underlying notion also holds for multi-label setups. In that case, we need to be able to select several classes by considering the distances (logits) to all hyperplanes. To achieve that we employ the $\mathrm{sigmoid}$ function as f, instead of a $\mathrm{softmax}$, and predict the given class if $p(y = k|x) > 0.5$. Figure 7.4 shows examples of the hyperbolic definition of multiple hyperplanes, which follow the curvature of the space.

7.3.6 Optimization

With the proposed classification model, we aim to minimize variants of the binary cross-entropy loss function as the training objective. The model has trainable parameters in both Euclidean and hyperbolic space. We apply the Geoopt implementation of *Riemannian Adam* (Kochurov et al., 2020) as a Riemannian adaptive optimization method (Bécigneul & Ganea, 2019) to carry out a gradient-based update of the parameters in their respective geometry.

7.4 Experiments

We evaluate the proposed hyperbolic model on two different datasets for fine-grained entity typing (task is described in Definition 6.1), and compare to Euclidean baselines as well as

[2]The Lorentz factors are given by: $\gamma(x) = 1/\sqrt{1-\|x\|^2}$

[3]For more details, see Ganea et al. (2018b), §3.1.

state-of-the-art models.[4]

7.4.1 Data

For analysis and evaluation of the model, we focus on the UltraFine entity typing dataset Choi et al. (2018). To gain a better understanding of the proposed model, we also experiment on the OntoNotes dataset Gillick et al. (2014) as it is a standard benchmark for entity typing. Both datasets have been thoroughly described and analyzed in §6.2, where it was observed that the label inventories show an underlying hierarchical structure. Given this characteristic, and according to different metrics, the analysis suggests that both instances would profit from a representation in hyperbolic space, rather than in a Euclidean one.

7.4.2 Setup

The MLR classifier operates in a hyperbolic space of m dimensions with $m = d_M + d_C + 2d_S$. By setting different values, we experiment with three models: BASE $(m = 100)$, LARGE $(m = 250)$ and XLARGE $(m = 500)$.

As word embeddings we employ Poincaré GloVe embeddings (Tifrea et al., 2019), which are pre-trained in the Poincaré model. Hence, the input to the encoders is already in hyperbolic space and all operations can be performed in this geometry. These embeddings are not updated during training. Low values of dropout are used since the model was very sensitive to this parameter given the behaviour of the hyperbolic distance with respect to the norm of the points. Our model is implemented in PyTorch (Paszke et al., 2019).

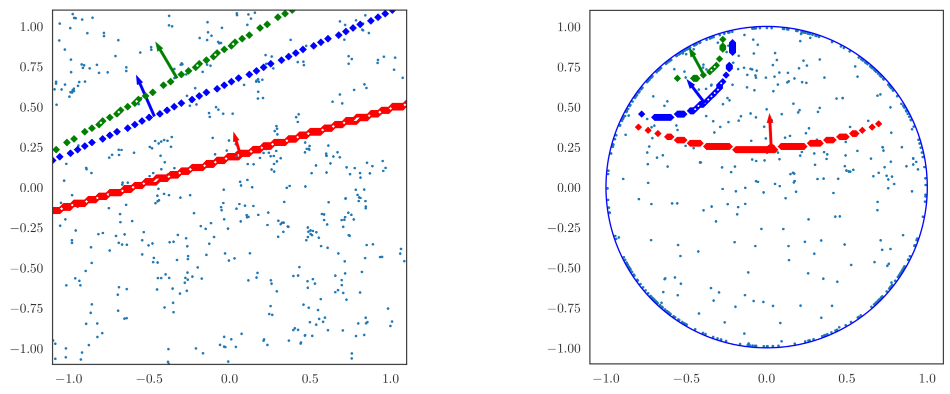
On the UltraFine dataset, for each epoch, we train over the entire training set, and we run extra iterations over the crowdsourced split before evaluating. In this way, the model benefits from the large amount of noisy, automatically-generated data, and is fine-tuned with high-quality crowdsourced samples. As previous work (Xiong et al., 2019; Onoe & Durrett, 2019), we optimize the multi-task objective proposed by Choi et al. (2018).

For evaluation we report Macro-averaged and Micro-averaged F_1 metrics computed from the precision/recall scores over the same three granularities established by Choi et al. (2018). For all models we optimize *Total* Macro-averaged F_1 on the validation set, and evaluate on the test set. Following Ganea et al. (2018b), we report the average of three runs given the highly non-convex spectrum of hyperbolic neural networks.

7.4.3 Baselines

Euclidean baseline: We replace all operations of the hyperbolic model by their Euclidean counterpart. To map the Poincaré GloVe embeddings to the Euclidean space we apply $\log_{\mathbf{0}}$.

[4]Code available at: `https://github.com/nlpAThits/hyfi`



(a) Euclidean Space. (b) Hyperbolic Space.

Figure 7.4: Classification hyperplanes for the types `person` (red), `artist` (blue) and `musician` (green). The hyperbolic formulation of the hyperplanes is better suited for hierarchical inventories.

We do not apply any kind of normalization or correction over the weights to circumscribe them into the unit ball. On the contrary, we grant them freedom over the entire Euclidean space to establish a fair comparison.

Multi-task: Model proposed by Choi et al. (2018), along with the UltraFine dataset.

LabelGCN: Model introduced by Xiong et al. (2019). A label-relational inductive bias is imposed by means of a graph propagation layer that encodes label co-occurrence statistics.

BERT: We compare to the setup of Onoe & Durrett (2019) in which BERT Devlin et al. (2019) is adapted for this task and fine-tuned on the crowdsourced train split.

Denoised: An ELMo-based model Peters et al. (2018) proposed by Onoe & Durrett (2019) trained on raw and denoised distantly-labeled data.

7.5 Results and Discussion

Following previous work (Choi et al., 2018; Onoe & Durrett, 2019), we report results on the development set in Table 7.1. All hyperbolic models outperform MultiTask and LabelGCN baselines on *Total* Macro F_1. Denoised and BERT systems, based on large pre-trained models, show the best *Total* performance. Nonetheless, hy xLarge has a competitive performance, and surpasses both systems on *ultra-fine* F_1. In the hyperbolic model, fine-grained types are placed near the boundary of the ball, where the amount of space grows exponentially. Furthermore, the underlying structure of the type inventory is hierarchical, thus the hyperbolic definition of the hyperplanes is well-suited to improve the classification in this case (see comparison with Euclidean classifiers on Figure 7.4). These properties combined enable the hyperbolic model to excel at classifying hierarchical labels, with outstanding improvements on very fine-grained types.

	Total			Coarse			Fine			Ultra-Fine			
Model	**P**	**R**	**F$_1$**	**P**	**R**	**F$_1$**	**P**	**R**	**F$_1$**	**P**	**R**	**F$_1$**	**# Params**
DENOISED	50.7	33.1	**40.1**	66.9	**80.7**	73.2	41.7	46.2	43.8	45.6	17.4	25.2	31.0M
BERT	**51.6**	32.8	**40.1**	**67.4**	80.6	**73.4**	41.6	**54.7**	**47.3**	**46.3**	15.6	23.4	110.0M
LABELGCN	49.3	28.1	35.8	66.2	68.8	67.5	**43.9**	40.7	42.2	42.4	14.2	21.3	5.1M
MULTITASK	48.0	23.0	31.0	60.0	61.0	61.0	40.0	38.0	39.0	42.0	8.0	14.0	6.1M
HY BASE	48.5	29.1	36.3	64.4	72.2	68.1	39.4	38.5	38.9	39.3	14.5	21.2	1.8M
HY LARGE	42.3	33.5	37.4	63.6	72.1	67.6	36.3	48.3	41.4	33.3	19.7	24.7	4.6M
HY XLARGE	43.4	**34.2**	38.2	61.4	73.9	67.1	35.7	46.6	40.4	36.5	**19.9**	**25.7**	9.5M

Table 7.1: Macro-averaged P, R and F$_1$ on the UltraFine dev set for different baselines and models. We only reproduced LABELGCN. Values for other baselines are taken from the original publications.

The reduction of the parameter size is also remarkable: 70% and 91% versus DENOISED and BERT respectively. This emphasizes the importance of choosing a suitable metric space that fits the data distribution (hierarchical in this case) as a powerful and efficient inductive bias. Through adequate tools and formulations, we are able to exploit this bias without introducing an overload on the parameter cost.

Correspondence of results between HY BASE and LABELGCN suggest that both models capture similar information. LABELGCN requires label co-occurrence statistics represented as a weighted graph, from where a hierarchy can be easily derived (Krioukov et al., 2010). The similarity of results indicates that the hyperbolic model is able to implicitly encode the latent hierarchical information in the label co-occurrences without additional inputs or the burden of the graph layer.

To shed light on this aspect, we inspect the embeddings p_k learned by HY BASE to define the hyperplanes of Equation 7.8. Table 7.2 shows the types corresponding to the closest points to the label `person` and its *subtypes*, measured by hyperbolic distance. The types are highly correlated given that they often co-occur in similar contexts. Moreover, the model captures hyponymic relations (*is-a*) present in the label co-occurrences. An analogous behaviour is observed for other types in Table 7.3. The inductive bias given by

`person`		`artist`		`musician`	
Types	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$
artist	0.26	*musician*	0.25	*singer*	0.24
author	0.28	actor	0.26	actor	0.25
actor	0.30	person	0.26	artist	0.25
speaker	0.30	author	0.26	composer	0.27
leader	0.30	singer	0.28	band	0.27

Table 7.2: Closest p_k points in the Poincaré Ball to different UltraFine entity types. The model is able to capture hierarchical relations such as `singer` *is-a* `musician` *is-a* `artist` *is-a* `person`.

`organization`		`institution`		`firm`		`group`		`unit`		`division`	
Types	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$
institution	0.34	firm	0.24	business	0.23	unit	0.34	division	0.26	subsidiary	0.25
company	0.35	company	0.26	institution	0.24	gathering	0.34	theatre	0.28	unit	0.26
news_agency	0.36	university	0.26	company	0.25	subject	0.34	activist	0.28	track	0.28
business	0.38	operator	0.28	maker	0.27	administration	0.36	position	0.28	half	0.28
administration	0.40	maker	0.28	operator	0.28	affiliation	0.36	half	0.28	activist	0.29
`location`		`state`		`country`		`place`		`space`		`half`	
Types	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$
state	0.33	country	0.29	state	0.31	space	0.40	half	0.28	peak	0.26
cemetery	0.35	half	0.31	nation	0.31	localization	0.40	shopping_mall	0.29	operator	0.26
space	0.35	agency	0.31	agency	0.32	place_name	0.40	venue	0.29	theatre	0.26
half	0.35	activist	0.32	kingdom	0.34	close	0.41	landmark	0.30	placement	0.26
area	0.36	unit	0.32	world	0.35	birthplace	0.41	localization	0.30	summit	0.26
`event`		`conflict`		`war`		`time`		`duration`		`calendar`	
Types	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$
conflict	0.44	war	0.34	guerrilla	0.32	duration	0.40	calendar	0.30	date	0.22
activist	0.45	dispute	0.36	conflict	0.34	period	0.43	peak	0.31	phrase	0.25
election	0.45	series	0.37	military	0.35	length	0.46	half	0.32	second	0.26
activity	0.46	guerrilla	0.38	citizen	0.36	month	0.46	second	0.32	activist	0.27
holiday	0.46	future	0.38	situation	0.36	date	0.46	fantasy	0.32	need	0.28
`object`		`machine`		`computer`		`entity`		`separation`		`placement`	
Types	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$	**Types**	$d_{\mathbb{D}}$
machine	0.37	computer	0.29	version	0.29	separation	0.43	placement	0.27	position	0.25
arrangement	0.39	theatre	0.30	machine	0.30	relative	0.44	missionary	0.27	localization	0.26
medium	0.39	operator	0.30	communication	0.30	meaning	0.44	meaning	0.27	half	0.26
method	0.39	card_game	0.31	activist	0.31	warlord	0.45	variation	0.27	separation	0.27
representation	0.39	core	0.31	maker	0.32	baseball	0.45	phrase	0.27	winner	0.27

Table 7.3: Closest p_k points in the Poincaré Ball to *coarse* entity types, with their hyperbolic distance. In many cases, a hierarchical relation holds with the closest type. For example: `firm` *is-a* `institution` *is-a* `organization`.

the hyperbolic geometry allows the model to capture the hierarchy, deriving a meaningful and interpretable representation of the label space: *coarse* labels near the origin, *fine-grained* labels near the boundary, and hyponymic relations are preserved. It is also noteworthy that the model learns these relations automatically without requiring explicitly annotated data.

Finally, we can mine the positions and distances of the embeddings in the space to recover a discrete graph representation of the inventory. From the tables 7.2 and 7.3 we can observe that relative distances are a good indicator of semantic similarity, and this can be further combined with the norm of the embeddings, which in the hyperbolic representation signals its hierarchy.

7.6 Ablations and Analysis

7.6.1 Comparison of the Spaces

A comparison of the metric spaces for different models on the test set is shown in Table 7.4. It can be seen that the hyperbolic model outperforms its Euclidean variants in all settings. It is notable that this trend holds even in high-dimensional spaces (500 dimensions for

Model		Coarse		Fine		Ultra	
		Ma	Mi	Ma	Mi	Ma	Mi
BASE	HY	**69.6**	**67.3**	**42.0**	**39.7**	21.2	19.1
	EU	68.5	66.1	39.8	36.5	17.8	16.1
LARGE	HY	67.9	65.4	38.4	36.3	24.3	22.3
	EU	67.1	63.8	36.7	34.7	22.0	19.7
XLARGE	HY	69.1	66.2	39.7	37.2	**26.1**	**24.0**
	EU	67.9	65.4	37.8	35.3	22.2	20.0

Table 7.4: Results on Ultra-Fine test set for macro and micro F_1 across metric spaces and dimensions.

XLARGE). Since the label inventory exhibits a clearly hierarchical structure, it perfectly suits the hyperbolic classification method.

The hyperbolic model brings considerable gains as the granularity becomes finer: 5.1% and 16.2% relative improvement in *fine* and *ultra-fine* Macro F_1 respectively for the BASE model over its Euclidean counterpart. We also observe that as the size of the model increases, the Euclidean baseline becomes more competitive for *ultra-fine*. This is due to the Euclidean model gaining enough capacity to accommodate the separation hyperplanes with higher dimensions, thus reducing the gap.

It is noticeable that the BASE model outperforms the larger ones on *coarse* and *fine* granularities. That is due to the larger models overfitting given the low dropout applied. Moreover, Euclidean and hyperbolic models exhibit a similar performance on the *coarse* granularity when compared to each other. A possible explanation is that the separation planes for these labels are located closer to the origin of the space. In this region, the spaces behave alike in terms of the distance calculation, and this similarity is reflected in the results as well.

7.6.2 Word Embeddings Ablation

The input for both the Euclidean and hyperbolic models are Poincaré GloVe embeddings, which are originally trained in hyperbolic space (Tifrea et al., 2019). This might favor the hyperbolic model, despite the application of the $\log_{\mathbf{0}}$ map in the Euclidean case. Thus, we replace the hyperbolic embeddings by the regular GloVe embeddings (Pennington et al.,

BASE	Coarse		Fine		Ultra	
Model	Ma	Mi	Ma	Mi	Ma	Mi
HY GLOVE	**68.7**	**66.6**	**41.5**	**38.8**	**22.1**	**20.1**
EU GLOVE	67.8	65.3	39.7	36.0	20.7	18.6

Table 7.5: Test results on Ultra-Fine. Poincaré GloVe embeddings (Tifrea et al., 2019) are replaced by regular GloVe embeddings. (Pennington et al., 2014).

2014), and use $\exp_{\mathbf{0}}$ on the hyperbolic model to project them into the ball.

Table 7.5 shows that the tendency of the BASE hyperbolic model outperforming the Euclidean one holds, and that the improvement does not come from the embeddings. Also, in this way we showcase how the hyperbolic model can be easily integrated with regular word embeddings.

7.6.3 Component Ablation

With the aim of analyzing the contribution of the different hyperbolic components, we perform an ablation study on the BASE model. We divide the system in *encoder*, *attention* (both in the mention and context encoders), *concatenation*, and *MLR*, and replace them, one at a time, by their Euclidean counterparts. Note that when Euclidean and hyperbolic components are mixed, we convert the internal representations from one manifold to the other with the $\exp_{\mathbf{0}}$ and $\log_{\mathbf{0}}$ maps.

As we see in Table 7.6, MLR is the component that contributes the most to the *ultra-fine* classification. The hierarchical structure of the type inventory combined with the hyperbolic definition of the hyperplanes are the reason of this (see Figure 7.4).

Hyperbolic attention and concatenation are relevant for *coarse* and *fine-grained* classification (here is where the biggest drop appears when they are removed), but do not play a major role in the *ultra-fine* granularity.

Finally, the encoders do not benefit from the hyperbolic representation. As the reason for this we consider that the model is not able to capture tree-like relations among the input tokens such that they can be exploited for the task.

This ablation suggests that the main benefits of hyperbolic layers arise when they are incorporated at deeper levels of representation in the model, and not over low-level features or raw text.

Computing time: Möbius operations are more expensive than their Euclidean counterparts. Due to this, in our experiments we found the hyperbolic encoder to be twice slower, and the MLR 1.5 times slower than their Euclidean versions.

7.6.4 OntoNotes Dataset

To further understand the capabilities of the proposed model we also perform an evaluation on the OntoNotes dataset (Gillick et al., 2014). In this case, we apply the standard binary cross-entropy loss, since fine-grained labels are scarce in this dataset. Following previous work (Xiong et al., 2019), we train over the dataset augmented by Choi et al. (2018). Results for the three granularities for BASE and LARGE models are presented in Table 7.7. The hyperbolic models outperform the Euclidean baselines in both cases, and the difference

Model	Coarse Ma	Coarse Mi	Fine Ma	Fine Mi	Ultra Ma	Ultra Mi
HY BASE	**69.6**	**67.3**	**42.0**	**39.7**	21.2	19.1
EU Encoder	68.8	66.3	41.7	38.9	**22.0**	**20.1**
EU Attention	68.9	66.4	40.8	38.0	20.1	18.4
EU Concat	68.6	66.1	40.6	37.5	21.8	19.8
EU MLR	69.2	67.1	40.8	38.0	17.3	15.8

Table 7.6: Results on Ultra-Fine test set. Ablation of the hyperbolic model, replacing one component by its Euclidean counterpart at a time.

Model		Coarse Ma	Coarse Mi	Fine Ma	Fine Mi	Ultra Ma	Ultra Mi
BASE	HY	82.0	80.2	41.8	**41.4**	23.9	25.0
	EU	81.8	80.3	37.7	36.1	17.5	15.8
LARGE	HY	**83.1**	**81.3**	**42.0**	**41.4**	**24.0**	**25.2**
	EU	82.4	80.9	38.2	36.7	18.9	18.1

Table 7.7: Macro and micro F_1 on OntoNotes test set for different granularities.

is noticeable for *fine* and *ultra-fine* (42.0 vs 38.2 and 24.0 vs 18.9 on Macro F_1 for the LARGE model), in accordance with the results on Ultra-Fine.

We report a comparison with neural systems in Table 7.8. The hyperbolic model, without requiring the explicit hierarchy provided in this dataset, achieves a competitive performance. Nonetheless, the advantages of the hyperbolic model are mitigated by the low multiplicity of fine-grained labels, and the lower hierarchy.

7.7 Conclusions

Modelling inter-label correlations from the label inventory has become critical to improve performance. Hyperbolic spaces offer an exciting approach since they are naturally

Model	Acc.	Ma-F_1	Mi-F_1
Shimaoka et al. (2017)	51.7	70.9	64.9
AFET (Ren et al., 2016a)	55.1	71.1	64.7
PLE (Ren et al., 2016b)	57.2	71.5	66.1
BERT	51.8	76.6	69.1
MULTITASK	59.5	76.8	71.8
LABELGCN	**59.6**	**77.8**	**72.2**
HY LARGE	47.4	75.8	69.4

Table 7.8: Total accuracy, macro and micro F_1 scores on OntoNotes test set.

equipped to model hierarchical structures. However, previous work integrated isolated components into neural systems. In this chapter we propose a fully hyperbolic model and showcase its effectiveness on challenging datasets. We build upon the hyperbolic neural layers introduced in Ganea et al. (2018b), and develop the missing components to perform, not binary, but multi-class multi-label text classification. We test the proposed model not with a synthetic dataset, but on a concrete downstream tasks, such as entity typing. Our work resembles Chen et al. (2019), though they separately learn embeddings for type labels and text representations in hyperbolic space, whereas we do it in an integrated fashion. Experimental evidence suggests that our model encodes similar hierarchical information to the work of Xiong et al. (2019), but without the need to provide it explicitly.

Our hyperbolic model achieves a performance comparable to state-of-the-art systems on very fine-grained labels with a remarkable reduction of the parameter size. This emphasizes the importance of choosing a metric space suitable to the data distribution as an effective inductive bias to capture fundamental properties, such as hierarchical structure. In addition to performing classification, the model simultaneously learns label embeddings. The arrangement of these embeddings in the target space automatically captures implicit hyponymic relations in the inventory, and help us to infer the latent hierarchy from the class distribution. The label embeddings are conceptually equivalent to node embeddings for a graph derived from the inventory, and connected according to the distribution in the dataset and the task-specific objective.

Finally, we illustrate ways to integrate different hyperbolic components with Euclidean layers, showing their strengths and drawbacks. Our ablation suggests that the benefits of hyperbolic layers become evident when they are incorporated at deeper levels of representation in the model. An interesting future direction is to develop hyperbolic adapters in combination with contextualized word embeddings, in order to map the pre-trained representations into the target geometry.

From the component ablation, we can also notice the advantages of combining Euclidean and hyperbolic representations into a unified model. However, up until this point we have had to choose which space to use in advance. In the following chapters, we provide a systematic approach to symmetric spaces. The compound geometry of the matrix manifolds we study simultaneously contains Euclidean as well as hyperbolic or spherical subspaces. This richer structure eliminates the need to specify one fixed geometry in advance, allowing symmetric spaces to automatically adapt to dissimilar features in the graphs without a priori knowledge of their internal structure.

Part III

Embeddings Graphs in Matrix Manifolds

Chapter 8

A Framework for Graph Embeddings on Symmetric Spaces

"Symmetry is one idea by which humanity through the ages has tried to comprehend and create order, beauty, and perfection."
– Hermann Weyl

As we have previously discussed, the goal of representation learning is to embed data, frequently modeled on a graph, into an ambient space to then perform tasks on the discrete graph. In Chapter 5 we have reviewed many methods for this purpose in Euclidean space, and in Chapters 6 & 7 we have developed and applied different techniques in hyperbolic space. Moreover, there is an extensive strand of research that seeks to represent graphs in non-Euclidean domains. In addition to hyperbolic space (Krioukov et al., 2009; Nickel & Kiela, 2017; Sala et al., 2018), embeddings in spherical spaces (Wilson et al., 2014; Liu et al., 2017; Xu & Durrett, 2018) have been developed as well. Recent work proposes to combine different curvatures through several layers (Chami et al., 2019; Bachmann et al., 2020; Grattarola et al., 2020), to enrich the geometry by considering Cartesian products of spaces (Gu et al., 2019; Tifrea et al., 2019; Skopek et al., 2020), or to use Grassmannian manifolds or the space of symmetric positive definite matrices (SPD) as a trade-off between the representation capability and the computational tractability of the space (Huang & Gool, 2017; Huang et al., 2018; Cruceru et al., 2020). A unified framework in which to encompass these various examples is still missing.

In this chapter, we propose the systematic use of symmetric spaces in representation learning: this is a class comprising all the aforementioned spaces. Symmetric spaces are Riemannian manifolds with rich symmetry groups which makes them algorithmically tractable. They have a compound geometry that simultaneously contains Euclidean as well as hyperbolic or spherical subspaces, eliminating the need to choose one fixed geometry in advance, and allowing them to automatically adapt to dissimilar features in the graph (see

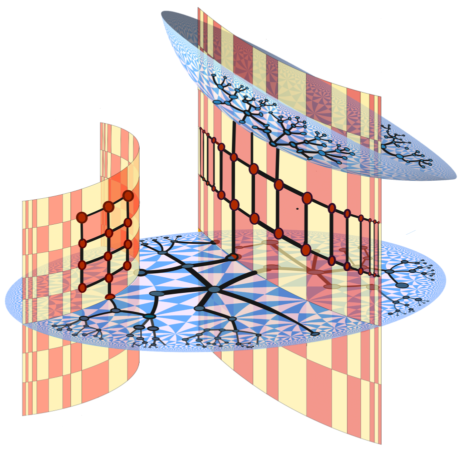


Figure 8.1: Symmetric spaces have a rich structure of totally geodesic subspaces, including flat subspaces (orange) and hyperbolic planes (blue). This compound, yet computationally tractable geometry allows isometric embeddings of many graphs, including those with subgraphs of dissimilar geometry. For example the graph embedded in the picture has both trees and grids as subgraphs.

Figure 8.1).

Algorithms formulated on abstract manifolds are not strictly speaking numerical algorithms since they involve manipulating different geometric objects, instead of numerical calculations (Absil et al., 2009). Turning these abstract algorithms into concrete numerical procedures relies on producing adequate formulas and representations of the geometric objects. In this chapter, we develop a general framework that makes it possible to perform this "geometric-to-numerical" conversion on Riemannian symmetric spaces for graph embedding problems.

We develop a general framework to choose a Riemannian symmetric space and implement the mathematical tools required to compute distances and perform Riemannian optimization. The output of our framework is a neural model capable of learning graph embeddings in the chosen space. Our systematic view enables us to introduce the use of Finsler metrics integrated with a Riemannian optimization scheme as a new method to achieve graph representations. Moreover, we use the vector-valued distance function on symmetric spaces (explained in §8.1.1) to develop new tools for the analysis of the structural properties of the embedded graphs.

In the first part of this chapter we review the main mathematical background, and we outline the steps of our general framework. In the second part, we demonstrate a concrete implementation of our general framework on Siegel spaces (Siegel, 1943). This is a family of symmetric spaces that has not been explored in geometric deep learning, despite them being among the most versatile symmetric spaces of non-positive curvature. Finally, we evaluate the representation capabilities of the proposed approach on different tasks. The results manifest the effectiveness and versatility of the method, particularly for graphs with varying and intricate structures.

8.1 Symmetric Spaces for Embedding Problems

Riemannian symmetric spaces (RSS) are Riemannian manifolds with rich symmetry groups, which makes them amenable to analytical tools as well as to explicit computations. This rich class, that includes many of the spaces previously applied in representation learning, has been extensively studied by mathematicians, who established many general tools that can be used for explicit algorithmic implementation. We refer to §2.5 and Appendix B for a general introduction to the theory, the duality between compact and non-compact RSS, general algorithms to compute the distance function in these spaces, and formulas for computing the exponential map, and parallel transport. These unify the discussions for Euclidean space, hyperbolic spaces, spherical spaces, Grassmannian manifolds, and SPD already available in the representation learning literature.

Key features of (non-compact) RSS are that they offer an effective combination of Euclidean and hyperbolic geometry, without necessarily separating it in different factors. In fact, RSS are Riemannian manifolds of variable curvature which have a superior structure than products of constant curvature spaces. Furthermore, they have many subspaces isometric to Euclidean, hyperbolic spaces and products thereof. This eliminates the need to specify a fixed geometry in advance, making them an excellent target for learning embeddings of complex networks without a priori knowledge of their internal structure. Finally, the rich symmetry group of RSS is important because it makes them algorithmically tractable: it allows us to get explicit formulas, which are crucial for concrete implementations.

Given the aforementioned reasons, we propose the systematic use of Symmetric spaces in representation learning. In the following subsections, we introduce two aspects of the general theory of RSS to representation learning: Finsler distances and vector-valued distances. These give us, respectively, a concrete method to obtain better graph representations, and a new tool to analyze graph embeddings. Then, we describe our general framework for symmetric spaces.

8.1.1 Vector-valued Distance

In this section we review important notions about the vector-valued distance (VVD), already introduced in §2.5.2, and highlight the advantages it poses for the case of learning graph embeddings.

In Euclidean space, in the sphere or in hyperbolic space, the only invariant of two points is their distance. A pair of points can be mapped to any other pair of points if and only if their *distance* is the same. Instead, in a general RSS the invariant between two points is a *distance vector* in $\mathbb{R}^n$, where n is the rank of the RSS. This is, two pairs of points can be separated by the same distance, but have different *distance vectors*.

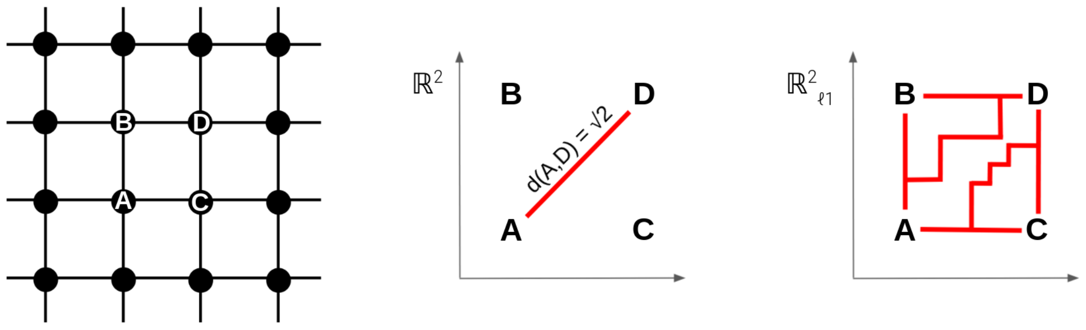


Figure 8.2: Left: Grid graph to be embedded. Center: Embeddings of the grid graph in the Euclidean plane $\mathbb{R}^2$ with Riemannian distance. Some distances get distorted by a factor of $\sqrt{2}$. Right: Same embeddings in $\mathbb{R}^2$ endowed with the ℓ^1 (or taxicab) metric. There are several shortest paths between A and D, and all of them accurately represent the same geodesic distance than in the graph.

The dimension of the space in which the vector-valued distance takes values defines the *rank* of the RSS. Geometrically, this represents the largest Euclidean subspace which can be isometrically embedded (hence, hyperbolic and spherical spaces are of $\mathrm{rank}\,-1$). The symmetries of an RSS fixing such a *maximal flat* form a finite group — the Weyl group of the RSS.

As we said, the VVD contains much more information than just the distance. Out of the VVD between two points, one can immediately read the regularity of the unique geodesics joining these two points. This is not possible knowing just the distance as a scalar. Moreover, the VVD contains the full information of the Riemannian distance and of all invariant Finsler distances (discussed in the next section).

From an application standpoint, the VVD offers several advantages. A single model can be run with different metrics, according to the chosen norm. We can easily recover the Riemannian metric and extend the approach to many other variants. Furthermore, given a representation of a graph in Riemannian symmetric space of rank n, we get finer invariants for the relative position between nodes of the graph, which can be useful to analyze the structure of the graph or perform tasks. We leverage this information to visualize the learned high-dimensional representations in §8.4.5.

In Appendix B we summarize the process for computing the VVD in general symmetric spaces.

8.1.2 Finsler Distances

Riemannian metrics are not well adapted to represent graphs. For example, though a two dimensional grid intuitively looks like a plane, any embedding of it in the Euclidean plane $\mathbb{R}^2$ necessarily distorts some distances by a factor of at least $\sqrt{2}$. This is due to the fact that while in the Euclidean plane length minimizing paths (geodesics) are unique, in graphs there are generally several shortest paths (see Figure 8.2). Instead, it is possible to find an abstract isometric embedding of the grid in $\mathbb{R}^2$ if the latter is endowed with the ℓ^1 (or taxicab) metric. In that case, the geodesic distances in the space accurately represent the distances in the graph. This is a first example of a Finsler distance (which have been introduced in §2.5.3). Another Finsler distance on $\mathbb{R}^n$ that plays a role in our work is the ℓ^∞ metric.

RSS do not only support a Riemannian metric, but a whole family of Finsler distances with the same symmetry group (group of isometries). For the reasons explained above, these Finsler metrics are more suitable to embed complex networks. Since Finsler metrics are in general not convex, they are less suitable for optimization problems. Due to this, we propose to combine the Riemannian and Finsler structure, by using a Riemannian optimization scheme, with loss functions based on the Finsler metric.

For additional notions on Finsler metrics for symmetric spaces, please refer to Appendix B.

8.2 The SYMPA Framework

The general theory of RSS not only unifies many spaces previously applied in representation learning, but also systematises their implementation. Using standard tools of this theory, in this section we provide a general framework to implement the mathematical methods required, and yield a neural model that learns graph embeddings in a given RSS. We outline the step involved in our framework:

1. Choosing a Riemmanian symmetric space.
2. Choosing a model of the symmetric space.
3. Implement an algorithm to compute distances
4. Implement an algorithm to compute gradients

We cover all these steps in detail in the following sections.

8.2.1 Choosing a Symmetric Space

We may utilize the classical theory of symmetric spaces to inform our choice of RSS. Every symmetric space M can be decomposed into an (almost) product $M = M_1 \times \cdots \times M_k$ of irreducible symmetric spaces. Apart from twelve exceptional examples, there are eleven infinite families of irreducible symmetric spaces to choose from, which we illustrate in Table 8.1 (see Helgason (1978) for more details). Each family of irreducible symmetric space has a distinct family of symmetry groups, which in turn determines

Type	Non-compact	Compact	$\mathrm{rk}_{\mathbb{R}}$	dim
AI	$\mathrm{SL}(n,\mathbb{R})/\mathrm{SO}(n,\mathbb{R})$	$\mathrm{SU}(n)/\mathrm{SO}(n)$	$n-1$	$\frac{(n-1)(n+2)}{2}$
A	$\mathrm{SL}(n,\mathbb{C})/\mathrm{SU}(2)$	$(\mathrm{SU}(n)\times\mathrm{SU}(n))/\mathrm{SU}(n)$	$n-1$	$(n+1)(n-1)$
BDI	$\mathrm{SO}(p,q)/\mathrm{SO}(p)\times\mathrm{SO}(q)$	$\mathrm{SO}(p+q)/\mathrm{SO}(p)\times\mathrm{SO}(q)$	$\min\{p,q\}$	pq
AIII	$\mathrm{SU}(p,q)/\mathrm{SU}(p)\times\mathrm{SU}(q)$	$\mathrm{SU}(p+q)/\mathrm{SU}(p)\times\mathrm{SU}(q)$	$\min\{p,q\}$	$2pq$
CI	$\mathrm{Sp}(2n,\mathbb{R})/\mathrm{U}(n)$	$\mathrm{Sp}(2n)/\mathrm{U}(n)$	n	$2n(n+1)$
DIII	$\mathrm{SO}^*(2n)/\mathrm{U}(n)$	$\mathrm{SO}(2n)/\mathrm{U}(n)$	$\lfloor\frac{n}{2}\rfloor$	$n(n-1)$
CII	$\mathrm{Sp}(p,q)/\mathrm{Sp}(p)\times\mathrm{Sp}(q)$	$\mathrm{Sp}(p+q)/\mathrm{Sp}(p)\times\mathrm{Sp}(q)$	$\min\{p,q\}$	$4pq$
AII	$\mathrm{SL}(n,\mathbb{H})/\mathrm{Sp}(n)$	$\mathrm{SU}(2n)/\mathrm{Sp}(n)$	$n-1$	$(n-1)(2n+1)$
D	$\mathrm{SO}(2n,\mathbb{C})/\mathrm{SO}(2n)$	$(\mathrm{SO}(2n)\times\mathrm{SO}(2n))/\mathrm{SO}(2n)$	n	$n(2n-1)$
B	$\mathrm{SO}(2n+1,\mathbb{C})/\mathrm{SO}(2n+1)$	$(\mathrm{SO}(2n+1)\times\mathrm{SO}(2n+1))/\mathrm{SO}(2n+1)$	n	$n(2n+1)$
C	$\mathrm{Sp}(n,\mathbb{C})/\mathrm{Sp}(n)$	$(\mathrm{Sp}(n)\times\mathrm{Sp}(n))/\mathrm{Sp}(n)$	n	

Table 8.1: The classical symmetric spaces. Row CI represents the Siegel spaces and their compact duals.

many mathematical properties of interest (for instance, the symmetry group determines the shape of the Weyl chambers, which determines the admissible Finsler metrics). Given a geometric property of interest, the theory of RSS allows one to determine which (if any) symmetric spaces enjoy it. For example, in §8.3 we choose Siegel spaces (row CI in Table 8.1) also because they admit Finsler metrics induced by the ℓ^1 metric on flats, which agrees with the intrinsic metric on grid-like graphs.

8.2.2 Choosing a Model of the Symmetric Space

Having selected an RSS, we must also select a model: a space M representing its points equipped with an action of its symmetry group G. Such a choice is of practical, rather than theoretical concern: the points of M should be easy to work with, and the symmetries of G straightforward to compute and apply. Each RSS may have many already-understood models in the literature to select from.

Implementing a product of symmetric spaces requires implementing each factor simultaneously. Given models $M_1,\ldots,M_k$ with symmetry groups $G_1,\ldots G_k$, the product $M = M_1\times\cdots\times M_k$ has as its points $m=(m_1,\ldots,m_k)$ the $k-$tuples with $m_i\in M_i$, with the group $G=G_1\times\cdots\times G_k$ acting component-wise. This general implementation of products directly generalizes products of constant curvature spaces.

8.2.3 Computing Distances

Given a choice of RSS, the fundamental quantity to compute is a distance function on M, typically used in the loss function. In contrast to general Riemannian manifolds, the rich symmetry of RSS allows this computation to be factored into a sequence of geometric steps. See Toolkit 1 for a schematic implementation using data from the standard theory of RSS (choice of maximal flat, Weyl chamber, and Finsler norm) and Algorithm 1 for a concrete implementation in the Siegel spaces.

Toolkit 1 Computing Distances on Riemannian Symmetric Spaces

```
1: Input from Model: Choice of basepoint $m$, maximal flat $F$, identification $\phi\colon F \to \mathbb{R}^n$,
   choice of Weyl Chamber $C \subset \mathbb{R}^n$, and Finsler norm $\|\cdot\|_F$ on $\mathbb{R}^n$.
2: Given $p, q \in M$:
3: Compute $g \in G$ such that $g(p) = m$ and $g(q) \in F$.
4: Compute $v' = \phi(g(q)) \in \mathbb{R}^n$, and $h \in G$ the Weyl group element such that $h(v') = v \in C$.
5: The Vector-valued Distance (VVD) is $\mathrm{vDist}(p, q) = v$.
6: The Riemannian Distance (RD) is $d^R(p, q) = \sqrt{\sum_i v_i^2}$.
7: The Finsler Distance (FD) is $d^F(p, q) = \|v\|_F$.
8: For a product $\prod M_i$, the VVD is the vector $(\mathrm{vDist}(p_i, q_i))$ of VVDs for each $M_i$. The RD,
   FD satisfy the pythagorean theorem: $d^X(p, q)^2 = \sum_i d^{X_i}(p_i, q_i)^2$, for $X \in \{R, F\}$.
```

8.2.4 Computing Gradients

To perform gradient-based optimization, the Riemannian gradient of these distance functions is required. Depending on the Riemannian optimization methods used, additional local geometry including parallel transport and the exponential map may be useful (Bonnabel, 2011; Bécigneul & Ganea, 2019). See Toolkit 2 for the relationships of these components to elements of the classical theory of RSS.

With respect to the use of Finsler metrics, we note that the combination of Finsler distances with a Riemannian optimization scheme is justified since the two metrics have the same isometry group. Thus, we can perfectly employ standard Riemannian optimization with a loss function based on Finsler distances. See Appendix B for a review of the general theory relevant to this schema.

8.3 SYMPA on Siegel Spaces

In this section we provide a concrete implementation of the general aspects of our SYMPA framework outlined above in the Siegel spaces.

Toolkit 2 Computing Local Geometry on Riemannian Symmetric Spaces

```
1: Input From Model: Geodesic reflections $\sigma_p \in G$, the metric tensor $\langle\cdot,\cdot\rangle$, basepoint $m \in M$,
   orthogonal decomposition $\mathfrak{stab}(m) \oplus \mathfrak{p} = \mathfrak{g}$, and identification $\phi\colon T_mM \to \mathfrak{p}$.
2: Given $f\colon M \to \mathbb{R}$, a geodesic $\gamma$, or $v \in T_mM$ respectively:
3: The Riemannian Gradient of $f$ is computed from the metric tensor by solving
   $\langle \mathrm{grad}_R(f), - \rangle = df(-)$
4: Parallel Transport along $\gamma$ is achieved by the differentials $(d\tau_t)_{\gamma(t_0)}$ of transvections $\tau_t =$
   $\sigma_{\gamma(t/2)}\sigma_{\gamma(t_0)}$ along $\gamma$.
5: The Riemannian Exponential $\exp_m^R(v) = g(m)$ is the matrix exponential $g = \exp(\phi(v)) \in$
   $G$ applied to $m$.
6: For a product $\prod M_i$ the Riemannian gradient, Parallel Transport, and Exponential map are
   computed component-wise.
```

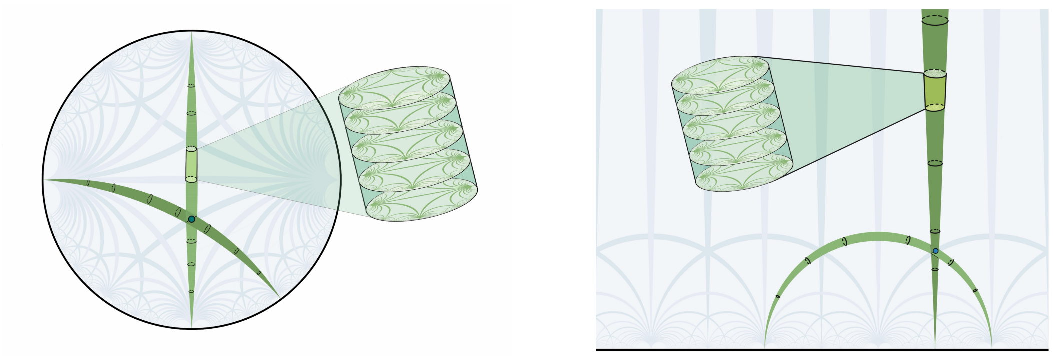

(a) Bounded Domain Model $\mathcal{B}_n$ (b) Siegel Upper Half Space $\mathcal{S}_n$

Figure 8.3: a) Every point of the disk is a complex symmetric n-dimensional matrix. b) A hyperbolic plane over SPD. $\mathcal{S}_2$ is a 6 dimensional manifold, the green lines represent totally geodesic submanifolds isometric to SPD that intersect in exactly one point. In dimension 2, SPD is isometric to the product of a hyperbolic plane and the line

8.3.1 Siegel Space

In this section we cover the step 1 of the framework, which consist of choosing a symmetric space. We work with Siegel spaces HypSPD$_n$ (Siegel, 1943) (row CI in Table 8.1), a versatile family of non-compact RSS, which has not yet been explored in geometric deep learning. The simplicity and the versatility of the Siegel space make it particularly suited for representation learning. Furthermore, Siegel spaces admit Finsler metrics induced by the ℓ^1 metric on flats, which agrees with the intrinsic metric on grid-like graphs.

8.3.2 Models of Siegel Spaces

Regarding the choice of a model (step 2), HypSPD$_n$ admits two distinct concrete and tractable matrix models generalizing the Poincaré disk and the upper half plane model of the hyperbolic space. Both their points and symmetries may be encoded by $n \times n$ matrices. In particular, they are open subsets of the space $\mathrm{Sym}(n, \mathbb{C})$ of symmetric $n \times n$-matrices over $\mathbb{C}$. HypSPD$_n$ has $n(n+1)$ dimensions.

The bounded symmetric domain model for HypSPD$_n$ generalizes the Poincaré disk. It is given by:[1]

$$\mathcal{B}_n := \{Z \in \mathrm{Sym}(n, \mathbb{C}) | \ \mathrm{Id} - Z^* Z \gg 0\}; \tag{8.1}$$

The Siegel upper half space model for HypSPD$_n$ generalizes the upper half plane model of the hyperbolic plane by:

$$\mathcal{S}_n := \{Z = X + iY \in \mathrm{Sym}(n, \mathbb{C}) | \ Y \gg 0\}. \tag{8.2}$$

[1]For a real symmetric matrix $Y \in \mathrm{Sym}(n, \mathbb{R})$ we write $Y \gg 0$ to indicate that Y is positive definite.

An explicit isomorphism from $\mathcal{B}_n$ to $\mathcal{S}_n$ is given by the Cayley transform (Cayley, 1846), a matrix analogue of the familiar map from the Poincare disk to upper half space model of the hyperbolic plane:

$$Z \mapsto i(Z + \mathrm{Id})(Z - \mathrm{Id})^{-1}.$$

The Siegel space HypSPD$_n$ contains SPD$_n$ as a totally geodesic submanifold, and in fact, it can be considered as a hyperbolic plane over SPD. The role that real lines play in the hyperbolic plane, in HypSPD$_n$ is played by SPD$_n$. This is illustrated in Figure 8.3b.

The Siegel space HypSPD$_n$ contains n-dimensional Euclidean subspaces, products of n-copies of hyperbolic planes, SPD$_n$ as well as products of Euclidean and hyperbolic spaces as totally geodesic subspaces (see Figure 8.3). It thus has a richer pattern of submanifolds than, for example, SPD. In particular, HypSPD$_n$ contains more products of hyperbolic planes than SPD$_n$: in HypSPD$_n$ we need 6 real dimension to contain $\mathbb{H}^2 \times \mathbb{H}^2$ and 12 real dimension to contain $(\mathbb{H}^2)^3$, whereas in SPD$_n$ we would need 9 (resp. 20) dimensions for this.

Implementation: A complex number $z \in \mathbb{C}$ can be written as $z = x+iy$ where $x, y \in \mathbb{R}$ and $i^2 = -1$. Analogously a complex symmetric matrix $Z \in \mathrm{Sym}(n, \mathbb{C})$ can be written as $Z = X + iY$, where $X = \Re(Z), Y = \Im(Z) \in \mathrm{Sym}(n, \mathbb{R})$ are symmetric matrices with real entries. We denote by $Z^* = X - iY$ the complex conjugate matrix.

Scalability: Like all RSS, HypSPD$_n$ has a *dual* – an RSS with similar mathematical properties but reversed curvature – generalizing the duality of hyperbolic spaces $\mathbb{H}^2$ and spherical spaces $\mathbb{S}^2$. We focus on HypSPD$_n$ over its dual for scalability reasons. The dual is a nonnegatively curved RSS of finite diameter, and thus does not admit isometric embeddings of arbitrarily large graphs. HypSPD$_n$, being nonpositively curved and infinite diameter, does not suffer from this restriction. See Appendix C for details on its implementation and experiments with the dual.

8.3.3 Computing Distances on Siegel Spaces

The Siegel space supports a Finsler metric F_1 that induces the ℓ^1 metric on the Euclidean subspaces. As already remarked, the ℓ^1 metric is particularly suitable for representing product graphs, or graphs that contain product subgraphs. Among all possible Finsler metrics supported by HypSPD$_n$, we focus on F_1 and F_∞ (the latter induces the ℓ^∞ metric on the flat).

To compute distances we employ the vector-valued distance. In the Siegel space, the Weyl group acts by permutations and reflections of the coordinates, allowing us to canonically represent each vector-valued distance as an n-tuple of non-increasing positive

Algorithm 1 Computing Distances on HypSPD$_n$

```
1: Given two points $Z_1, Z_2 \in \mathcal{S}_n$:
2: Define $Z_3 = \sqrt{\Im(Z_1)}^{-1}(Z_2 - \Re(Z_1))\sqrt{\Im(Z_1)}^{-1} \in \mathcal{S}_n$
3: Define $W = (Z_3 - i\mathrm{Id})(Z_3 + i\mathrm{Id})^{-1} \in \mathcal{B}_n$
4: Use the Takagi factorization to write $W = \overline{K}DK^*$ for $D$ real diagonal, and $K$ unitary.
5: Define $v_i = \log \frac{1+d_i}{1-d_i}$ for $d_i$ the diagonal entries of $D$.
6: Order the $v_i$ so that $v_1 \geq v_2 \geq \cdots \geq 0$. The **vector-valued distance** is $\mathrm{vDist}(Z_1, Z_2) = (v_1, v_2, \ldots, v_n)$.
7: The **Riemannian distance** is $d^R(Z_1, Z_2) := \sqrt{\sum_{i=1}^n v_i^2}$.
8: The **Finsler distance inducing the $\ell^1$-metric** is $d^{F1}(Z_1, Z_2) := \sum_{i=1}^n v_i$.
9: The **Finsler distance inducing the $\ell^\infty$-metric** is $d^{F\infty}(Z_1, Z_2) := \max\{v_i\} = v_1$.
```

numbers. Such a uniform choice of standard representative for all vector-valued distances is a fundamental domain for this group action, known as a *Weyl chamber* for the RSS. The scalar distance is given by applying either Riemannian or Finsler distance functions to the vector-valued distance. These computations are described in Algorithm 1, which is a concrete implementation of Toolkit 1.

Specifically, step 2 moves one point to the basepoint, step 4 moves the other into our chosen flat, step 5 identifies this with $\mathbb{R}^n$ and step 6 returns the vector-valued distance, from which all distances are computed.

We employ the Takagi factorization (Takagi, 1924) to obtain eigenvalues and eigenvectors of complex symmetric matrices in a tractable manner with automatic differentiation tools. See Appendix C for a detail explanation of this procedure.

Complexity of Distance Algorithm: Calculating distance between two points Z_1, Z_2 in either $\mathcal{S}_n$ or $\mathcal{B}_n$ spaces implies computing multiplications, inversions and diagonalizations of $n \times n$ matrices. We find that the cost of the distance computation with respect to the matrix dimensions is $\mathcal{O}(n^3)$. We prove this in Appendix C.

8.3.4 Riemannian Optimization on Siegel Spaces

With the proposed matrix models of the Siegel space, we optimize objectives based on the Riemannian or Finsler distance functions in the embeddings space. To overcome the lack of convexity of Finsler metrics, we combine the Riemannian and the Finsler structure, by using a Riemannian optimization scheme (Bonnabel, 2011) with a loss function based on the Finsler metric. In Algorithm 2 we provide a way to compute the Riemannian gradient from the Euclidean gradient obtained via automatic differentiation. This is a direct implementation of Toolkit 2 Item 3.

Given the Euclidean gradient of f obtained via automatic differentiation $\mathrm{grad}_E(f(Z))$, the Riemannian gradient for the Siegel upperhalf space at a point $Z \in \mathcal{S}_n$, where $Z =$

Algorithm 2 Computing the Riemannian Gradient on HypSPD$_n$

```
1: Given $f : \mathcal{S}_n \to \mathbb{R}$ and $Z = X + iY \in \mathcal{S}_n$:
2: Compute the Euclidean gradient $\text{grad}_E(f)$ at $Z$ of $f$ obtained via automatic differentiation
   (see Appendix C).
3: **The Riemannian gradient** is $\text{grad}_\text{R}(f) = Y \cdot \text{grad}_\text{E}(f) \cdot Y$.
```

$X + iY$ is given by:

$$\text{grad}(f(Z)) = Y \cdot \text{grad}_\text{E}(f(Z)) \cdot Y$$

In the case of the Bounded domain model, for $U \in \mathcal{B}_n$, we have:

$$\text{grad}(f(U)) = A \cdot \text{grad}_\text{E}(f(U)) \cdot A$$

where $A = \text{Id} - \overline{U}U$

To constrain the embeddings to remain within the Siegel space, we utilize a projection from the ambient space to our model. More precisely, given ϵ and a point $Z \in \text{Sym}(n, \mathbb{C})$, we compute a point $Z_\epsilon^\mathcal{S}$ (resp. $Z_\epsilon^\mathcal{B}$) close to the original point lying in the ϵ-interior of the model. For $\mathcal{S}_n$, starting from $Z = X + iY$ we orthogonally diagonalize $Y = K^tDK$, and then modify $D = \text{diag}(d_i)$ by setting each diagonal entry to $\max\{d_i, \epsilon\}$. An analogous projection is defined on the bounded domain $\mathcal{B}_n$, see Appendix C.

8.4 Graph Reconstruction

In this section we evaluate the representation capabilities of the proposed approach for the task of graph reconstruction.[2]

8.4.1 Experimental Setup

Implementation: All models and experiments were implemented in PyTorch (Paszke et al., 2019) with distributed data parallelism, for high performance on clusters of CPUs/GPUs. Given a complex matrix $Z \in \mathbb{C}^{n\times n}$, we model real and imaginary components $Z = X + iY$ with $X, Y \in \mathbb{R}^{n\times n}$ separate matrices with real entries. We followed standard complex math to implement basic arithmetic matrix operations. For complex matrix inversion we implemented the procedure detailed in Falkenberg (2007).

Hardware: All experiments were run on Intel Cascade Lake CPUs, with microprocessors Intel Xeon Gold 6230 (20 Cores, 40 Threads, 2.1 GHz, 28MB Cache, 125W TDP). Although the code supports GPUs, we did not utilize them due to higher availability of CPU's.

[2]Code available at: https://github.com/fedelopez77/sympa

Training: We embed graph nodes in a transductive setting. As input and evaluation data we take the shortest distance in the graph between every pair of connected nodes. Unlike previous work (Gu et al., 2019; Cruceru et al., 2020) we do not apply any scaling, neither in the input graph distances nor in the distances calculated on the space. We experiment with the loss proposed in Gu et al. (2019). Given graph distances $\{d_{\mathcal{G}}(X_i, X_j)\}_{ij}$ between all pairs of connected nodes, the loss is defined as:

$$\mathcal{L}(x) = \sum_{1 \leq i \leq j \leq n} \left| \left(\frac{d_{\mathcal{P}}(x_i, x_j)}{d_{\mathcal{G}}(X_i, X_j)} \right)^2 - 1 \right|$$

where $d_{\mathcal{P}}(x_i, x_j)$ is the distance between the corresponding node representations in the embeddings space. This formulation of the loss function minimizes the relation between the distance in the space, compared to the distance in the graph, and captures the average distortion.

We initialize the matrix embeddings in the Siegel upper half space by adding small symmetric perturbations to the matrix basepoint $i\mathrm{Id}$. For the Bounded model, we additionally map the points with the Cayley transform (see Appendix C).

For all models and datasets we run the same grid search. We train for 3000 epochs, reducing the learning rate by a factor of 5 if the model does not improve the performance after 50 epochs, and early stopping based on the average distortion if the model does not improve after 150 epochs. We use the burn-in strategy (Nickel & Kiela, 2017; Cruceru et al., 2020) training with a 10 times smaller learning rate for the first 10 epochs. We experiment with learning rates from $\{0.05, 0.01, 0.005, 0.001\}$, batch sizes from $\{512, 1024, 2048\}$ and max gradient norm from $\{10, 50, 250\}$.

Optimization: As stated before, the models under consideration are Riemannian manifolds, therefore they can be optimized via stochastic Riemannian optimization methods such as RSGD (Bonnabel, 2011) (we adapt the Geoopt implementation (Kochurov et al., 2020)). Given a function $f(\theta)$ defined over the set of embeddings (parameters) θ and let ∇_R denote the Riemannian gradient of $f(\theta)$, the parameter update according to RSGD is of the form:

$$\theta_{t+1} = \mathcal{R}_{\theta_t}(-\eta_t \nabla_R f(\theta_t))$$

where $\mathcal{R}_{\theta_t}$ denotes the retraction onto space at θ and η_t denotes the learning rate at time t. Hence, to apply this type of optimization we require the Riemannian gradient and a suitable retraction. Following Nickel & Kiela (2017) we experiment with a simple retraction:

$$\mathcal{R}_{\theta_t}(v) = \theta + v$$

Baselines: We compare our approach to constant-curvature baselines, such as Euclidean ($\mathbb{E}$) and hyperbolic ($\mathbb{H}$) spaces (we compare to the Poincaré model (Nickel & Kiela, 2017) since the Bounded Domain model is a generalization of it), Cartesian products thereof ($\mathbb{E} \times \mathbb{H}$ and $\mathbb{H} \times \mathbb{H}$) (Gu et al., 2019), and symmetric positive definite matrices (SPD) (Cruceru et al., 2020) in low and high dimensions. Preliminary experiments on the *dual* of HypSPD$_n$ and on spherical spaces showed poor performance thus we do not compare to them (see Appendix C). To establish a fair comparison, each model has the same number of free parameters. This is, the spaces $\mathcal{S}_n$ and $\mathcal{B}_n$ have $n(n+1)$ parameters, thus we compare to baselines of the same dimensionality.[3] All implementations are taken from Geoopt (Kochurov et al., 2020).

Evaluation Metrics: To measure the quality of the learned embeddings we follow the same fidelity metrics applied in previous work (Sala et al., 2018; Gu et al., 2019), which are distortion and precision. The distortion of a pair of connected nodes a, b in the graph G, where $f(a), f(b)$ are their respective embeddings in the space $\mathcal{P}$ is given by:

$$\text{distortion}(a, b) = \frac{|d_{\mathcal{P}}(f(a), f(b)) - d_G(a, b)|}{d_G(a, b)}$$

The average distortion D_{avg} is the average over all pairs of points. Distortion is a global metric that considers the explicit value of all distances.

The other metric that we consider is the mean average precision (mAP). It is a ranking-based measure for local neighborhoods that does not track explicit distances. Let $G = (V, E)$ be a graph and node $a \in V$ have neighborhood $\mathcal{N}_a = \{b_1, ..., b_{\deg(a)}\}$, where $\deg(a)$ is the degree of a. In the embedding f, define R_{a,b_i} to be the smallest ball around $f(a)$ that contains b_i (that is, R_{a,b_i} is the smallest set of nearest points required to retrieve the i-th neighbor of a in f). Then:

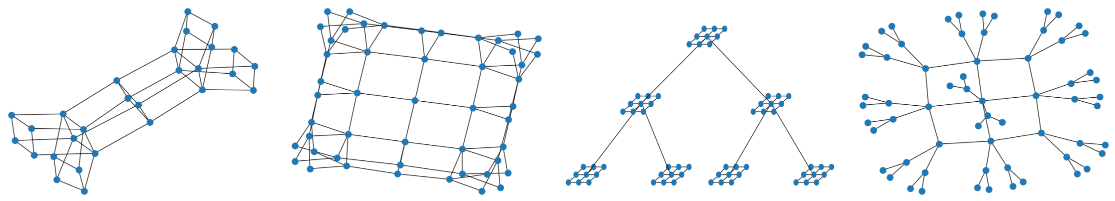
$$\text{mAP}(f) = \frac{1}{|V|} \sum_{a \in V} \frac{1}{\text{deg(a)}} \sum_{i=1}^{|\mathcal{N}_a|} \frac{|\mathcal{N}_a \cap R_{a,b_i}|}{|R_{a,b_i}|}$$

In all cases we report the average of 5 runs.

8.4.2 Synthetic Graphs

As a first step, we investigate the representation capabilities of different geometric spaces on synthetic graphs. Previous work has focused on graphs with pure geometric features, such as grids, trees, or their Cartesian products (Gu et al., 2019; Cruceru et al., 2020), which mix the grid- and tree-like features globally. We expand our analysis to rooted

[3]We also consider comparable dimensionalities for SPD$_n$, which has $n(n+1)/2$ parameters.



(a) TREE × GRID (b) TREE × TREE (c) TREE ⋄ GRIDS (d) GRID ⋄ TREES

Figure 8.4: a) Cartesian product of tree and 2D grid. b) Cartesian product of tree and tree. c) Rooted product of tree and 2D grids. d) Rooted product of 2D grid and trees.

products of trees and grids. These graphs mix features at different levels and scales. Thus, they reflect to a greater extent the complexity of intertwining and varying structure in different regions, making them a better approximation of real-world datasets. We consider the rooted product TREE ⋄ GRIDS of a tree and 2D grids, and GRID ⋄ TREES, of a 2D grid and trees.

We employ NetworkX (Hagberg et al., 2008) to generate the synthetic datasets, and their Cartesian and rooted products. The statistics of the synthetic datasets are presented in Table 8.2, and a diagram of the graphs can be seen in Figure 8.4.

By triples we mean the 3-tuple $(\mathbf{u}, \mathbf{v}, d(\mathbf{u}, \mathbf{v}))$, where **u, v** represent connected nodes in the graph, and $d(\mathbf{u,v})$ is the shortest distance between them.

Results

We report the results for synthetic graphs in Table 8.3. We find that the Siegel space with Finsler metrics significantly outperform constant curvature baselines in all graphs, except for the tree, where they have competitive results with the hyperbolic models. We observe that Siegel spaces with the Riemannian metric perform on par with the matching geometric spaces or with the best-fitting product of spaces across graphs of pure geometry (grids and Cartesian products of graphs). However, the F_1 metric outperforms the Riemannian and F_∞ metrics in all graphs, for both models. This is particularly noticeable for the 4D GRID, where the distortion achieved by F_1 models is almost null, matching the intuition of less

Graph	**Nodes**	**Edges**	**Triples**	**Grid Layout**	**Tree Valency**	**Tree Height**
4D GRID	625	2000	195,000	$(5)^4$		
TREE	364	363	66,066		3	5
TREE × GRID	496	1,224	122,760	4×4	2	3
TREE × TREE	225	420	25,200		2	3
TREE ⋄ GRIDS	775	1,270	299,925	5×5	2	4
GRID ⋄ TREES	775	790	299,925	5×5	2	4

Table 8.2: Synthetic graph statistics.

	4D GRID		TREE		TREE × GRID		TREE × TREE		TREE ◇ GRIDS		GRID ◇ TREES	
$(\lvert V\rvert, \lvert E\rvert)$	$(625, 2000)$		$(364, 363)$		$(496, 1224)$		$(225, 420)$		$(775, 1270)$		$(775, 790)$	
	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
$\mathbb{E}^{20}$	11.24±0.00	**100.00**	3.92±0.04	42.30	9.81±0.00	83.32	9.78±0.00	96.03	3.86±0.02	34.21	4.28±0.04	27.50
$\mathbb{H}^{20}$	25.23±0.05	63.74	**0.54±0.02**	**100.00**	17.21±0.21	83.16	20.59±0.11	75.67	14.56±0.27	44.14	14.62±0.13	30.28
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	11.24±0.00	**100.00**	1.19±0.04	**100.00**	9.20±0.01	**100.00**	9.30±0.04	98.03	2.15±0.05	58.23	2.03±0.01	**97.88**
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	18.74±0.01	78.47	0.65±0.02	**100.00**	13.02±0.91	88.01	8.61±0.03	97.63	1.08±0.06	77.20	2.80±0.65	84.88
SPD_6	11.24±0.00	**100.00**	1.79±0.02	55.92	9.23±0.01	99.73	8.83±0.01	98.49	1.56±0.02	62.31	1.83±0.00	72.17
$\mathcal{S}_4^R$	11.27±0.01	**100.00**	1.35±0.02	78.53	9.13±0.01	99.92	8.68±0.02	98.03	1.45±0.09	72.49	1.54±0.08	76.66
$\mathcal{S}_4^{F_\infty}$	5.92±0.06	99.61	1.23±0.28	99.56	4.81±0.55	99.28	3.31±0.06	99.95	10.88±0.19	63.52	10.48±0.21	72.53
$\mathcal{S}_4^{F_1}$	**0.01±0.00**	**100.00**	0.76±0.02	91.57	**0.81±0.08**	**100.00**	**1.08±0.16**	**100.00**	**1.03±0.00**	**78.71**	**0.84±0.06**	80.52
$\mathcal{B}_4^R$	11.28±0.01	**100.00**	1.27±0.05	74.77	9.24±0.13	99.22	8.74±0.09	98.12	2.88±0.32	72.55	2.76±0.11	96.29
$\mathcal{B}_4^{F_\infty}$	7.32±0.16	97.92	1.51±0.13	99.73	8.70±0.87	96.40	4.26±0.26	99.70	6.55±1.77	73.80	7.15±0.85	90.51
$\mathcal{B}_4^{F_1}$	0.39±0.02	**100.00**	0.77±0.02	94.64	0.90±0.08	**100.00**	1.28±0.16	**100.00**	1.09±0.03	76.55	0.99±0.01	81.82

Table 8.3: Results for synthetic datasets. Lower D_{avg} is better. Higher mAP is better. Metrics are given as percentage.

distorted grid representations through the taxicab metric.

Even when the structure of the data conforms to the geometry of baselines, the Siegel spaces with the Finsler-Riemannian approach are able to outperform them by automatically adapting to very dissimilar patterns without any a priori estimates of the curvature or other features of the graph. This showcases the flexibility of our models, due to its enhanced geometry and higher expressivity.

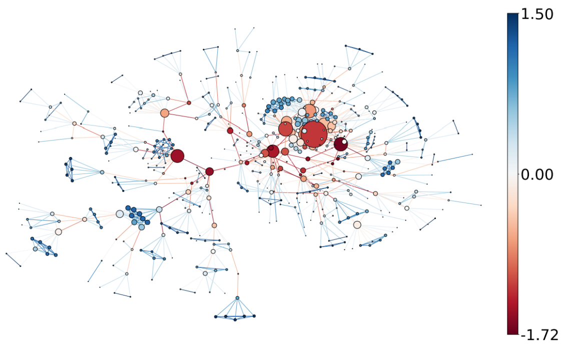
For graphs with mixed geometric features (rooted products), Cartesian products of spaces cannot arrange these compound geometries into separate Euclidean and hyperbolic subspaces. RSS, on the other hand, offer a less distorted representation of these tangled patterns by exploiting their richer geometry which mixes hyperbolic and Euclidean features. Moreover, they reach a competitive performance on the local neighborhood reconstruction, as the mean precision shows.

8.4.3 Real-world Graphs

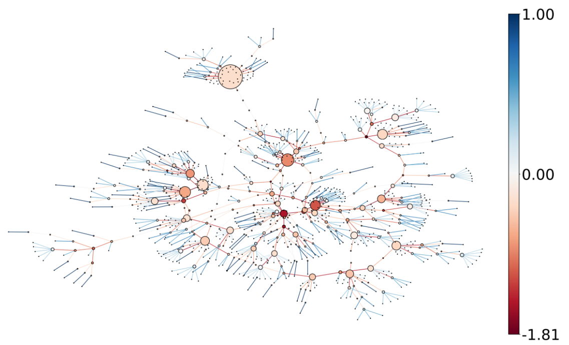
We compare the models on two road networks, namely USCA312 of distances between North American cities and EUROROAD between European cities, BIO-DISEASOME, a network of human disorders and diseases with reference to their genetic origins (Goh et al., 2007), a graph of computer science Ph.D. advisor-advisee relationships (Nooy et al., 2011), and a dense social network from Facebook (McAuley & Leskovec, 2012). These graphs

Graph	**Nodes**	**Edges**	**Triples**	**δ-mean**	**δ-max**	**Sectional Curvature**
USCA312	312	48,516	48,516			
BIO-DISEASOME	516	1,188	132,870	0.20	2.50	-0.29±0.48
CSPHD	1,025	1043	524,800	0.51	6.50	-0.77±0.38
EUROROAD	1,039	1305	539,241	0.73	7.00	-0.18±0.42
FACEBOOK	4,039	88234	8,154,741	0.10	1.50	

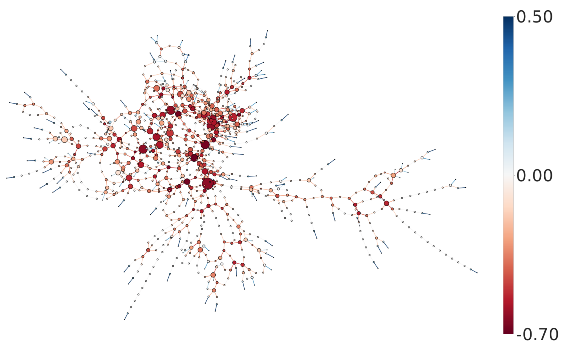
Table 8.4: Real-world graph statistics.



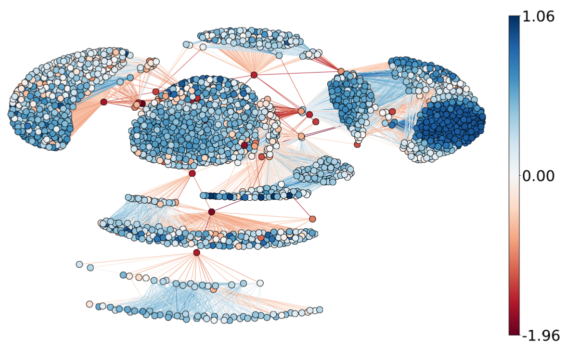
(a) BIO-DISEASOME



(b) CSPHD



(c) EUROROAD



(d) FACEBOOK

Figure 8.5: Ollivier-Ricci curvature plots. Figures for BIO-DISEASOME, CSPHD and FACEBOOK are taken from Cruceru et al. (2020).

have been analyzed in previous work as well (Gu et al., 2019; Cruceru et al., 2020). The datasets were downloaded from the Network Repository (Rossi & Ahmed, 2015). Stats are presented in Table 8.4. We do not compute all values for USCA312 since it is a complete weighted graph.

Graph Analysis

We report the datasets' curvature and hyperbolicity in Table 8.4. Regarding the distribution of the sectional curvature presented in Figure 8.6 we can observe that the CSPHD dataset has a very negative curvature given its tree-like structure. On the other hand, we can see on this plot and in Figure 8.5 that the FACEBOOK dataset is very dense, with cluster of very high connectivity, which induce a "flat" curvature. A similar pattern can be seen in some particular regions of the BIO-DISEASOME and EUROROAD datasets. This analysis highlights the diversity of topologies and characteristics of the graphs under study.

Results

We report the results in Table 8.5. On the USCA312 dataset, which is the only weighted graph under consideration, the Siegel spaces perform on par with the compared target

	USCA312	BIO-DISEASOME		CSPHD		EUROROAD		FACEBOOK	
$(\|V\|, \|E\|)$	$(312, 48516)$	$(516, 1188)$		$(1025, 1043)$		$(1039, 1305)$		$(4039, 88234)$	
	D_{avg}	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
$\mathbb{E}^{20}$	**0.18±0.01**	3.83±0.01	76.31	4.04±0.01	47.37	4.50±0.00	87.70	3.16±0.01	32.21
$\mathbb{H}^{20}$	2.39±0.02	6.83±0.08	91.26	22.42±0.23	60.24	43.56±0.44	54.25	3.72±0.00	44.85
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	**0.18±0.00**	2.52±0.02	91.99	3.06±0.02	73.25	4.24±0.02	89.93	2.80±0.01	34.26
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	0.47±0.18	2.57±0.05	**95.00**	7.02±1.07	**79.22**	23.30±1.62	75.07	2.51±0.00	36.39
SPD_6	0.21±0.02	2.54±0.00	82.66	2.92±0.11	57.88	19.54±0.99	92.38	2.92±0.05	33.73
$\mathcal{S}_4^R$	0.28±0.03	2.40±0.02	87.01	4.30±0.18	59.95	29.21±0.91	84.92	3.07±0.04	30.98
$\mathcal{S}_4^{F_\infty}$	0.57±0.08	2.78±0.49	93.95	27.27±1.00	59.45	46.82±1.02	72.03	**1.90±0.11**	**45.58**
$\mathcal{S}_4^{F_1}$	**0.18±0.02**	1.55±0.04	90.42	**1.50±0.03**	64.11	**3.79±0.07**	**94.63**	2.37±0.07	35.23
$\mathcal{B}_4^R$	0.24±0.07	2.69±0.10	89.11	28.65±3.39	62.66	53.45±2.65	48.75	3.58±0.10	30.35
$\mathcal{B}_4^{F_\infty}$	0.21±0.04	4.58±0.63	90.36	26.32±6.16	54.94	52.69±2.28	48.75	2.18±0.18	39.15
$\mathcal{B}_4^{F_1}$	**0.18±0.07**	**1.54±0.02**	90.41	2.96±0.91	67.58	21.98±0.62	91.63	5.05±0.03	39.87

Table 8.5: Results for real-world datasets. Lower D_{avg} is better. Higher mAP is better. Metrics are given as percentage.

manifolds. For all other datasets, the model with Finsler metrics outperforms all baselines. In line with the results for synthetic datasets, the F_1 metric exhibits an outstanding performance across several datasets.

Overall, these results show the strong reconstruction capabilities of RSS for real-world data as well. Our graph analysis indicates that vertices in these real-world dataset form networks with a more intricate and heterogeneous geometries, which the Siegel space is able to unfold to a better extent.

8.4.4 High-dimensional Spaces

In Table 8.6 we compare the approach in high-dimensional spaces (rank 17 which is equal to 306 free parameters), also including spherical spaces $\mathbb{S}$. The results show that our models operate well with larger matrices, where we see further improvement in our distortion and mean average precision over the low dimensional spaces of rank 4. We observe that even

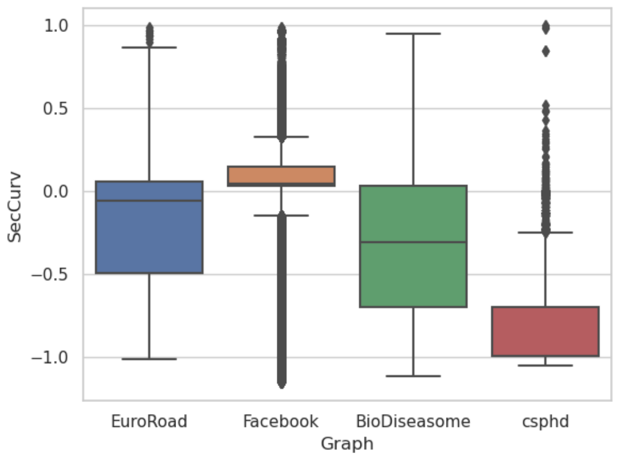


Figure 8.6: Sectional curvature for real-world graphs.

	TREE × GRID		GRID ◇ TREES		BIO-DISEASOME	
	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
$\mathcal{S}_4^R$	9.13	99.92	1.54	76.66	2.40	87.01
$\mathcal{S}_4^{F_\infty}$	4.81	99.28	10.48	72.53	2.78	93.95
$\mathcal{S}_4^{F_1}$	<u>0.81</u>	**100.00**	<u>0.84</u>	80.52	1.55	90.42
$\mathbb{E}^{306}$	9.80	85.14	2.81	67.69	3.52	88.45
$\mathbb{H}^{306}$	17.31	82.97	15.92	27.14	7.04	91.46
$\mathbb{S}^{306}$	73.78	35.36	81.67	58.26	70.91	84.61
$\mathbb{E}^{153} \times \mathbb{H}^{153}$	9.14	**100.00**	1.52	<u>97.85</u>	2.36	95.65
$\mathbb{S}^{153} \times \mathbb{S}^{153}$	60.71	6.93	70.00	5.64	55.51	19.51
$\mathcal{S}_{17}^R$	9.19	99.89	1.31	75.45	2.13	93.14
$\mathcal{S}_{17}^{F_\infty}$	4.82	97.45	11.45	94.09	<u>1.50</u>	<u>98.27</u>
$\mathcal{S}_{17}^{F_1}$	**0.03**	**100.00**	**0.27**	**99.23**	**0.73**	**99.09**

Table 8.6: Results for different datasets in high-dimensional spaces. Best result is **bold**, second best <u>underlined</u>.

though we notably increase the dimensions of the baselines to 306, the Siegel models of rank 4 (equivalent to 20 dimensions) significantly outperform them. These results match the expectation that the richer variable curvature geometry of RSS better adapts to graphs with intricate geometric structures.

8.4.5 New Tools to Analyze the Embedding Space

One reason to embed graphs into Riemannian manifolds is to use geometric properties of the manifold to analyze the structure of the graph. Embeddings into hyperbolic spaces, for example, have been used to infer and visualize hierarchical structure in data sets (Nickel & Kiela, 2018). Visualizations in RSS are difficult due to their high dimensionality. As a solution we use the vector-valued distance function in the RSS to develop new tools to visualize and to analyze structural properties of the graphs.

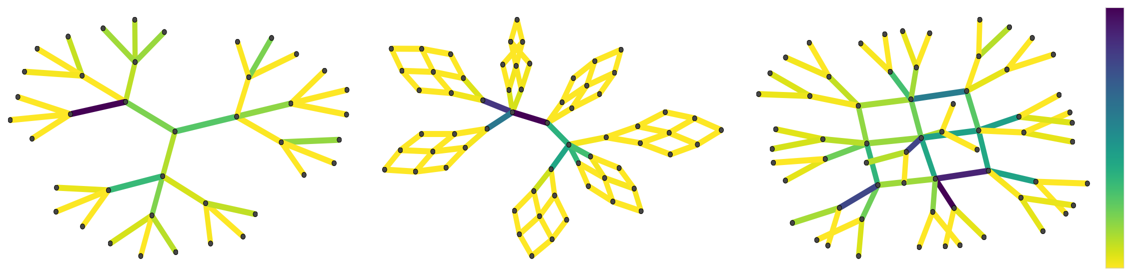


Figure 8.7: Edge coloring of $\mathcal{S}_2^{F1}$ for a tree (left), and a rooted product of TREE ◇ GRIDS (center), and of GRID ◇ TREES.

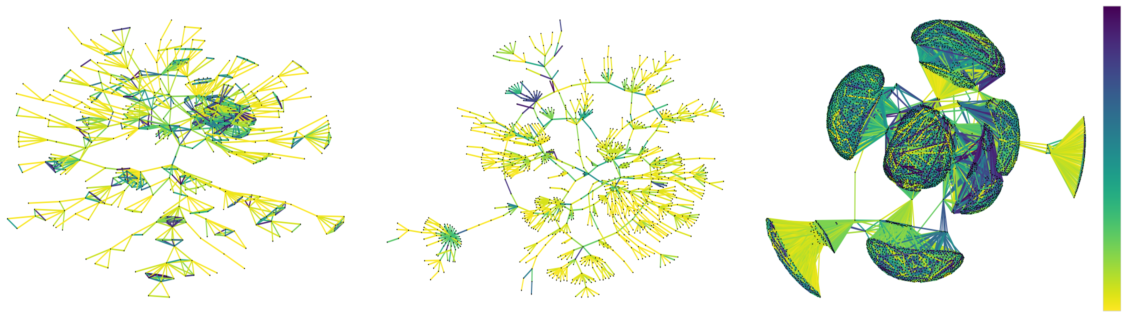


Figure 8.8: Edge coloring of $\mathcal{S}_2^{F1}$ for BIO-DISEASOME (left) and CSPHD (center) and FACEBOOK (right). Edge colors indicate the angle of the vector-valued distance for each edge, on a linear scale from 0 (yellow) to $\pi/4$ (blue).

Continuous Edge Coloring

We focus on HypSPD$_2$, the Siegel space of rank $k = 2$, where the vector-valued distance is just a vector in a cone in $\mathbb{R}^2$. We take edges connecting pairs of nodes (Z_i, Z_j) and assign the angle of the vector $\mathrm{vDist}(Z_i, Z_j) = (v_1, v_2)$ (see Algorithm 1, step 6) to each edge in the graph. This angle assignment provides a continuous edge coloring that can be leveraged to find structure in graphs.

Initially, we analyze S_2^{F1}. We see in Figure 8.7 that the edge coloring makes the large-scale structure of the tree (blue/green edges) and the leaves (yellow edges) visible. This is even more striking for the rooted products. In TREE $\diamond$ GRIDS the edge coloring distinguishes the hyperbolic parts of the graph (blue edges) and the Euclidean parts (yellow edges). For the GRID $\diamond$ TREES, the Euclidean parts are labelled by blue/green edges and the hyperbolic parts by yellow edges. Thus, even though we trained the embeddings based only on the distances, it automatically adapts to other features of the graph.

In the edge visualizations for real-world datasets (Figure 8.8), the edges in the denser connected parts of the graph have a higher angle, as it can be seen for the BIO-DISEASOME and FACEBOOK data sets. For CSPHD, the tree structure is emphasized by the low angles.

Furthermore, we compare the three analyzed metric spaces and we plot the edge coloring of $\mathcal{S}_2^R, \mathcal{S}_2^{F_\infty}$, and $\mathcal{S}_2^{F1}$ for the TREE $\diamond$ GRIDS and the GRID $\diamond$ TREES datasets in Figures 8.9 and 8.10 respectively. We can observe that in the Riemannian metric plots (left-hand side) there is no clear pattern that separates flat and hierarchical components in the graphs. The F_∞ and $F1$ metrics are the best at capturing the structural aspect of the datasets. They recognize very similar patterns, though they assign opposite angles to the vector-valued distance vectors, and this can be noticed from the fact that the colors assigned are in opposite sides of the spectrum (yellow means angles close to zero, blue means angles close to 45°).

This analysis suggests that the continuous values that we assign to edges are a powerful tool to automatically discover dissimilar patterns in graphs. This can be further used in efficient clustering of the graph.

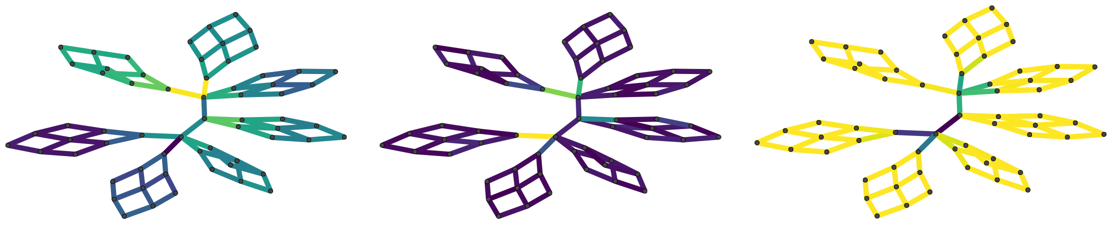


Figure 8.9: Edge coloring of $\mathcal{S}_2^R$ (left), $\mathcal{S}_2^{F\infty}$ (center), and $\mathcal{S}_2^{F1}$ (right) for a TREE ◇ GRIDS.

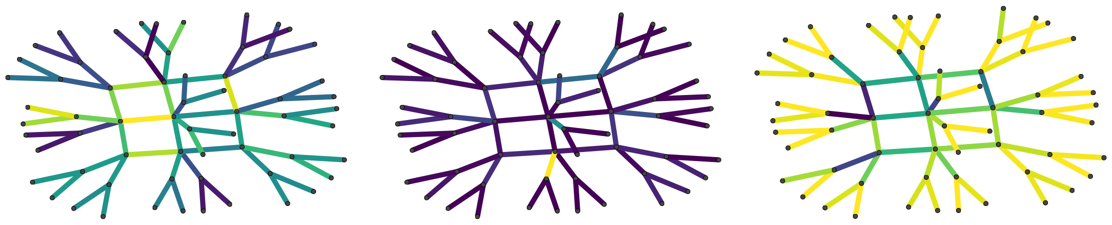


Figure 8.10: Edge coloring of $\mathcal{S}_2^R$ (left), $\mathcal{S}_2^{F\infty}$ (center), and $\mathcal{S}_2^{F1}$ (right) for a GRID ◇ TREES.

Continuous Node Coloring

To construct a continuous node coloring we choose one vertex of our graph as the root R. For every other node Z_i we take the vector-valued distance from R to Z_i, given by $\text{vDist}(R, Z_i) = (v_1, v_2)$, and assign the ratio v_2/v_1 as a value for the node Z_i. We again represent the corresponding real number by a color shading. It can be thought as the accumulated angle over a path from the root R to the node Z_i.

In Figure 8.11 we plot both, the edge and node coloring of the three metric spaces for the 2D grid graph. We see that all edges have the same angle. Furthermore, in the case of the Finsler distances we can also observe more clearly that the symmetries of the graph are respected in the embedding given the node coloring. This shows that the Finsler embeddings are much better in representing structural features of the graphs than the Riemannian embeddings.

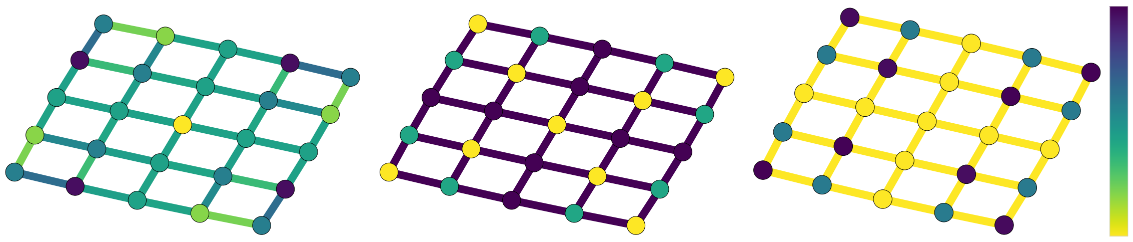


Figure 8.11: Analysis of $\mathcal{S}_2^R$ (left), $\mathcal{S}_2^{F\infty}$ (center), and $\mathcal{S}_2^{F1}$ (right) for a 5×5 grid. Node colors indicate the angle of the vector-valued distance by taking the path from the central node. Edge colors indicate the angle for each edge.

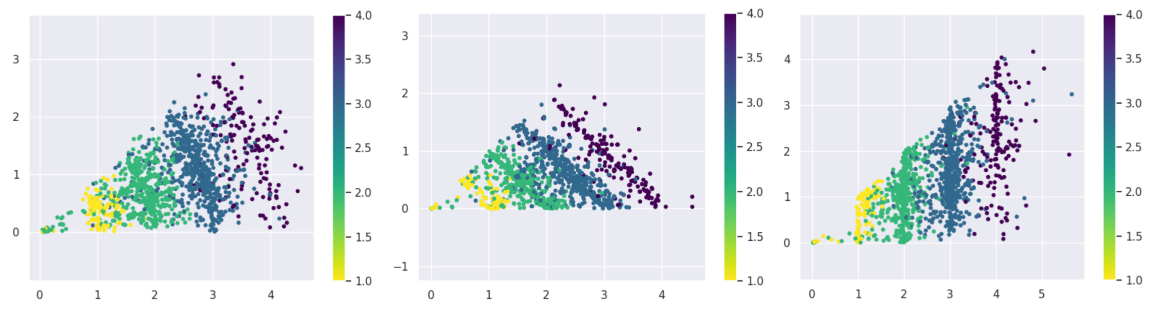


Figure 8.12: Plot of (v_1, v_2) of $\mathcal{S}_2^R$ (left), $\mathcal{S}_2^{F1}$ (center), and $\mathcal{S}_2^{F\infty}$ (right) for vertex pairs sampled from BIO-DISEASOME. Color indicates ground-truth distance.

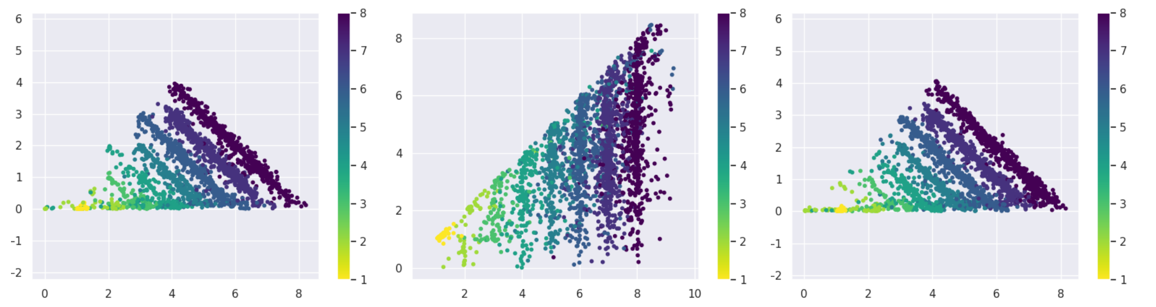


Figure 8.13: Plot of (v_1, v_2) of $\mathcal{S}_2^R$ (left), $\mathcal{S}_2^{F1}$ (center), and $\mathcal{S}_2^{F\infty}$ (right) for vertex pairs sampled from TREE. Color indicates ground-truth distance.

Vectorial Distance Plots

In this case, we sample pairs of connected vertices of the graph (Z_i, Z_j) and directly plot the result of $\mathrm{vDist}(Z_i, Z_j) = (v_1, v_2)$. In Figure 8.12 and 8.13 we show the plots of (v_1, v_2) for the embeddings of different dataset embedded into the Upper Half models with respect to Riemannian, F_1 and F_∞ metrics. In the F_1 case, the addition of both d-values sums up to the distance, whereas for the F_∞, the largest v (v_1) corresponds to the distance.

8.5 Recommender Systems

Our method can be applied in different downstream tasks that involve embedding graphs, such as recommender systems. The recommendation problem can be interpreted as a particular case of a link prediction task over a bipartite graph of users and items (Li et al., 2014). These systems mine user-item interactions and recommend items to users according to the distance/similarity between their respective embeddings (Hsieh et al., 2017).

8.5.1 Experimental Setup

Training: Given a set of observed user-item interactions $\mathcal{T} = \{(u, v)\}$, we follow a metric learning approach (Vinh Tran et al., 2020) and learn embeddings by optimizing the following hinge loss function:

$$\mathcal{L} = \sum_{(u,v)\in\mathcal{T}} \sum_{(u,w)\notin\mathcal{T}} [m + d_{\mathbb{K}}(\mathbf{u}, \mathbf{v})^2 - d_{\mathbb{K}}(\mathbf{u}, \mathbf{w})^2]_+ \tag{8.3}$$

where $\mathbb{K}$ is the target space, w is an item the user has not interacted with, $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{K}$, $m > 0$ is the hinge margin and $[z]_+ = max(0, z)$. To generate recommendations, for each user u we rank the items $\mathbf{v}_i$ according to their distance to $\mathbf{u}$. Since it is very costly to rank all the available items, we randomly select 100 samples which the user has not interacted with, and rank the ground truth amongst these samples (He et al., 2017).

Evaluation Metrics: We adopt normalized discounted cumulative gain (nDG) and hit ratio (HR), both at 10, as ranking evaluation metrics for recommendations.

Data: We evaluate the different models over two MovieLens datasets (ML-1M and ML-100K) (Harper & Konstan, 2015), LAST.FM, a dataset of artist listening records (Cantador et al., 2011), and MEETUP, crawled from Meetup.com (Pham et al., 2015). Statistics for the datasets are presented in Table 8.7. To generate evaluation splits, the penultimate and last item the user has interacted with are withheld as dev and test set respectively.

8.5.2 Results

We report the performance for all analyzed models in Table 8.8. While in the Movies datasets, the Riemannian model marginally outperforms the baselines, in the other two cases the F_1 model achieves the highest performance by a larger difference. These systems learn to model users' preferences, and embeds users and items in the space, in a way that is exploited for the task of generating recommendations. In this manner we demonstrate how downstream tasks can profit from the enhanced graph representation capacity of our models, and we highlight the flexibility of the method, in this case applied in combination

Dataset	**Users**	**Items**	**Interactions**	**Density (%)**
ML-1M	6,040	3,706	1,000,209	4.47
ML-100K	943	1,682	100,000	6.30
LAST.FM	1,892	17,632	92,834	0.28
MEETUP	46,895	16,612	277,863	0.04

Table 8.7: Recommender system dataset statistics.

	ML-1M		ML-100K		LAST.FM		MEETUP	
	HR@10	nDG	HR@10	nDG	HR@10	nDG	HR@10	nDG
$\mathbb{E}^{20}$	46.9±0.6	22.7	54.6±1.0	28.7	55.4±0.3	24.6	69.8±0.4	46.4
$\mathbb{H}^{20}$	46.0±0.5	23.0	53.4±1.0	28.2	54.8±0.5	24.9	71.8±0.5	48.5
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	52.0±0.7	27.4	53.1±1.3	27.9	45.5±0.9	18.9	70.7±0.2	47.5
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	46.7±0.6	23.0	54.8±0.9	29.1	55.0±0.9	24.6	71.7±0.1	48.8
$\mathcal{SPD}_6$	45.8±1.0	22.1	53.3±1.4	28.0	55.4±0.2	25.3	70.1±0.6	46.5
$\mathcal{S}_4^R$	**53.8±0.3**	**27.7**	**55.7±0.9**	28.6	53.1±0.5	24.8	65.8±1.2	43.4
$\mathcal{S}_4^{F_\infty}$	45.9±0.9	22.7	52.5±0.3	27.5	53.8±1.7	32.5	69.0±0.5	46.4
$\mathcal{S}_4^{F_1}$	52.9±0.6	27.2	55.6±1.3	**29.4**	**61.1±1.2**	**38.0**	**74.9±0.1**	**52.8**

Table 8.8: Results for recommender system datasets.

with a collaborative metric learning approach (Hsieh et al., 2017).

8.6 Node Classification

Our proposed graph embeddings can be used in conjunction with standard machine learning pipelines, such as downstream classification. To showcase this, we evaluate our model for node classification on three hierarchical clustering datasets.

8.6.1 Experimental Setup

Following the procedure of Chami et al. (2020b), we embed three hierarchical clustering datasets and then use the learned embeddings as input features for a logistic regression classifier.

Data: All datasets were downloaded from the UCI Machine Learning Repository (Dua & Graff, 2017).[4] Statistics about the datasets used are presented in Table 8.9.

Deriving a graph: For all datasets we use the cosine distance on the datapoints' features to compute a complete input distance graph. We employ the available features and normalize them so that each attribute has mean zero and standard deviation one. Once we have a graph, we embed it in the exact same way than in the graph reconstruction task.

[4] https://archive.ics.uci.edu/ml/datasets.php

Dataset	Nodes	Classes	Triples
IRIS	150	3	11,175
ZOO	101	7	5,050
GLASS	214	6	22,790

Table 8.9: Machine learning datasets used for node classification.

Dataset	IRIS	ZOO	GLASS
$\mathbb{E}^{20}$	83.3±1.1	88.7±1.8	67.2±2.5
$\mathbb{H}^{20}$	84.0±0.6	87.3±1.5	62.8±2.0
$\mathbb{E}^{10} \times \mathbb{H}^{10}$	85.6±1.1	88.0±1.4	64.8±4.3
$\mathbb{H}^{10} \times \mathbb{H}^{10}$	87.8±1.4	87.3±1.5	63.4±3.4
SPD_6	88.0±1.6	88.7±2.2	66.9±2.0
$\mathcal{S}_4^R$	88.0±0.5	88.7±2.2	66.6±2.4
$\mathcal{S}_4^{F_\infty}$	89.1±0.5	88.7±2.5	65.2±3.0
$\mathcal{S}_4^{F_1}$	**89.3±1.1**	**90.7±1.5**	**67.5±3.9**
$\mathcal{B}_4^R$	86.0±1.9	88.7±1.4	65.5±3.1
$\mathcal{B}_4^{F_\infty}$	84.4±0.0	87.3±1.9	65.6±1.7
$\mathcal{B}_4^{F_1}$	85.6±1.4	89.3±2.8	64.2±1.7

Table 8.10: Accuracy for node classification based on its embedding.

Matrix Mapping: Since the node embeddings lie in different metric spaces, we apply the corresponding logarithmic map to obtain a "flat" (Euclidean) representation before classifying. For the Siegel upper half-space model of dimension n, we apply the following mapping. From each complex matrix embedding $Z = X + iY$ we stack the result of the following operations in matrix form as:

$$M = \begin{pmatrix} Y + XY^{-1}X & XY^{-1} \\ Y^{-1}X & Y^{-1} \end{pmatrix}$$

where $M \in \mathbb{R}^{2n \times 2n}$. This mapping is the natural realisation of HypSPD$_n$ as a totally geodesic submanifold of SPD_{2n}. Since $M \in \text{SPD}_{2n}$, finally we apply the LogEig map as proposed by Huang & Gool (2017), which yields a representation in a flat space. This operations results in new matrix of the form:

$$\text{LogEig}(M) = \begin{pmatrix} U & V \\ V & -U \end{pmatrix}$$

where $U, V \in Sym(n)$. The final step is to take the upper triangular from U and V, and concatenate them as a vector of $n(n+1)$ dimensions. This procedure is implemented for the Upper half-space. In the case of the Bounded domain model, we first map the points to the upper half-space with the Cayley transform.

8.6.2 Results

Results are presented in Table 8.10. In all cases we see that the embeddings learned by our models capture the structural properties of the dataset, so that a simple classifier can separate the nodes into different clusters. $\mathcal{S}_4^{F_1}$ offers the best performance in the three datasets. This suggests that embeddings in Siegel spaces learn meaningful representations that can be exploited into downstream tasks. Moreover, we showcase how to map these embeddings to "flat" vectors. In this way, they can be integrated with classical Euclidean network layers.

8.7 Conclusions

Riemannian manifold learning has regained attention due to appealing geometric properties that allow methods to represent non-Euclidean data arising in several domains (Rubin-Delanchy, 2020). We propose the systematic use of symmetric spaces, which comprises embeddings in hyperbolic spaces (Chamberlain et al., 2017; Ganea et al., 2018a; Nickel & Kiela, 2018), spherical spaces (Meng et al., 2019; Defferrard et al., 2020), combinations thereof (Bachmann et al., 2020; Grattarola et al., 2020; Law & Stam, 2020), Cartesian products of spaces (Gu et al., 2019; Tifrea et al., 2019), Grassmannian manifolds (Huang et al., 2018) and the space of symmetric positive definite matrices (SPD) (Huang & Gool, 2017; Cruceru et al., 2020), among others.

We develop SYMPA, a general framework that allows practitioners to choose a Riemannian symmetric space and implement the mathematical tools required to learn graph embeddings. Moreover, we introduce the use of Finsler metrics integrated with a Riemannian optimization scheme, which provide a significantly less distorted representation over several data sets. As a new tool to discover structure in the graph, we leverage the vector-valued distance function on a RSS.

To demonstrate a concrete implementation, we apply our general framework on Siegel spaces, a rich class of RSS that had not been explored in geometric deep learning, and we develop tractable and mathematically sound algorithms to learn embeddings in these spaces through gradient-descent methods. We showcase the effectiveness of the proposed approach on conventional as well as new datasets for the graph reconstruction task, and in two downstream tasks: recommender systems and node classification. Our method ties or outperforms constant-curvature baselines without requiring any previous assumption on geometric features of the graphs. This shows the flexibility and enhanced representation capacity of Siegel spaces, as well as the versatility of our approach. As main limitation we consider the computational complexity of working with spaces of matrices. Due to this, the cost of many operations tends to be polynomial instead of linear.

In the following chapter we provide another implementation of the SYMPA framework on the space of symmetric positive definite matrices. Furthermore, we bridge the gap between Euclidean and SPD geometry by developing gyrocalculus in SPD. We showcase how the SYMPA framework can be seamlessly integrated with gyrocalculus, yielding closed-form expressions of arithmetic operations.

Chapter 9

Representing Multi-Relational Graphs on SPD Manifolds

"The real voyage of discovery consists not in seeking new lands but seeing with new eyes."
– Marcel Proust

Up until this point, we have worked with graphs where edges are associated with a numerical weight. In this chapter we shift the focus to multi-relational graphs, a different type of graph that accounts for edges as labeled relations with diverse semantic meaning. They are formally defined as:

Definition 9.1 (Multi-relational graphs)**.** They are given by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of nodes or vertices, $\mathcal{E}$ is the set of edges and $\mathcal{R}$ is the set of admissible relations in the graph. In this case, the edge notation is extended to include a relation $r \in \mathcal{R}$, such that for $u, v \in \mathcal{V}$, the edge is noted as $(u, r, v) \in \mathcal{E}$.

A popular application of multi-relational graphs is to employ them as *knowledge graphs*. Knowledge graphs (KGs) are data structures for representing heterogeneous facts in knowledge bases that use a graph-structured data model. In knowledge graphs, nodes represent entities and typed-edges represent relationships among entities. Facts are stored in the shape of *(head, relation, tail)* triples, which can be queried and used in downstream applications. See Figure 9.1 for an example.

The usual approach to work with KGs is to learn representations of entities and relations as embeddings, for some choice of space, such that the KG structure is preserved. In initial chapters we explored hyperbolic spaces, given their capabilities to accurately represent hierarchical graphs. However, knowledge graphs exhibit an intricate and varying structure as a result of the logical properties of the relationships they encode (Miller, 1992; Suchanek et al., 2007; Lehmann et al., 2015). An item can be connected to different entities by

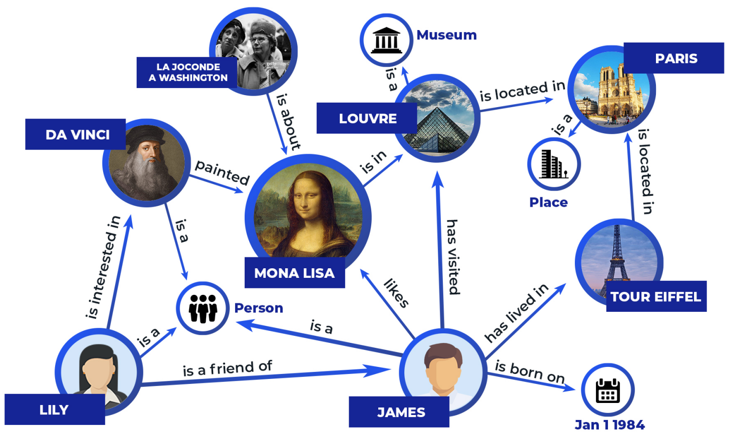


Figure 9.1: Knowledge graphs store heterogeneous information about diverse entities, such as people, objects, location, events, or abstract concepts. Source: Seth (2019).

symmetric, anti-symmetric, or hierarchical relations. To capture these non-trivial patterns more expressive spaces and operators become necessary.

In this chapter we explore the manifold of symmetric positive definite (SPD) matrices to learn graph representations. This manifold exhibits a high expressivity due to its richer geometry encompassing Euclidean as well as hyperbolic spaces, which it both contains. Hence, embeddings in this space can accommodate hierarchical structures in the hyperbolic subspaces, while at the same time represent subgraphs with Euclidean features. This makes SPD matrices a much more effective manifold than using only hyperbolic or Euclidean spaces, since we can combine both geometries yielding versatile graph embeddings.

The SPD space has been extensively studied in previous work (see §9.1) due to its many appealing properties. Since the SPD space is a Riemannian symmetric space, we reinterpret it under the lens of the SYMPA framework (proposed in the previous chapter). Through our framework, we develop a neural model capable of learning graph representations in the SPD manifold. Furthermore, we employ the vector-valued distance function, and revisit two of its main advantages. First, its versatility to implement universal models, where we can compute the Riemannian, or any Finsler distance. And second, its capacity to analyze the structural properties of the learned representations. We employ the vector-valued distance to develop a tool to visualize high-dimensional SPD embeddings, providing better explainability on what the models learn.

In order to operate with the learned representations in the space, we bridge the gap between Euclidean and SPD geometry by developing gyrocalculus in SPD. The flexibility of the SYMPA framework allows us to seamlessly integrate the algebraic formalism of gyrocalculus, yielding closed-form expressions of arithmetic operations, such as addition, scalar multiplication and matrix scaling. This provides means to translate previously implemented ideas in different metric spaces to their analog notions in SPD. Our methods faithfully respect the geometric structure of the Riemannian manifold and thus benefit from its enhanced representation capabilities.

9.1 Related Work on SPD

Symmetric Positive Definite matrices have been applied in many tasks in computer vision such as pedestrian detection (Tuzel et al., 2008; Tosato et al., 2010), action (Harandi et al., 2014; Li & Lu, 2018; Nguyen et al., 2019) or face recognition (Huang et al., 2014, 2015), object (Ionescu et al., 2015; Yin et al., 2016) and image set classification (Wang et al., 2018), visual tracking (Wu et al., 2015), and medical imaging analysis (Pennec et al., 2006; Arsigny et al., 2006b) among others. They have been used to capture statistical notions (Gaussian distributions (Said et al., 2017), covariance (Tuzel et al., 2006)), while respecting the Riemannian geometry of the underlying SPD manifold, which offers a convenient trade-off between structural richness and computational tractability (Cruceru et al., 2020). Previous work has applied approximation methods that locally flatten the manifold by projecting it to its tangent space (Carreira et al., 2012; Vemulapalli & Jacobs, 2015), or by embedding the manifold into higher dimensional Hilbert spaces (Ha Quang et al., 2014; Yin et al., 2016).

These methods face problems such as distortion of the geometrical structure of the manifold and other known concerns with regard to high-dimensional spaces (Dong et al., 2017). To overcome these issues, several distances on SPD manifolds have been proposed, such as the Affine Invariant metric (Pennec et al., 2006), the Stein metric (Sra, 2012), the Bures–Wasserstein metric (Bhatia et al., 2019) or the Log-Euclidean metric (Arsigny et al., 2006a,b), with their respective geometric properties. However, the representational power of SPD is not fully exploited in many cases (Pennec et al., 2006; Arsigny et al., 2006a). At the same time, it is hard to translate operations into their non-Euclidean domain given the lack of closed-form expressions. There has been a growing need to generalize basic operations, such as addition, rotation, reflection or scalar multiplication, to their Riemannian geometric counterparts to leverage this structure.

In the context of Deep Learning, previous work has proposed alternatives to the basic neural building blocks respecting the geometry of the space. For example, transformation layers (Dong et al., 2017; Gao et al., 2019; Huang & Gool, 2017), alternate convolutional layers based on SPDs (Zhang et al., 2018a) and Riemannian means (Chakraborty et al., 2020), or appended after the convolution (Brooks et al., 2019b), recurrent models (Chakraborty et al., 2018), projections onto Euclidean spaces (Li et al., 2018; Mao et al., 2019) and batch normalization (Brooks et al., 2019a). Our work follows this line, providing explicit formulas for translating Euclidean arithmetic notions into SPDs.

By applying our general SYMPA framework on the SPD manifold, we exploit the vector-valued distance function, and treat Riemannian and Finsler metrics on SPD in a unified approach. Finsler metrics have previously been applied in compressed sensing (Donoho & Tsaig, 2008), information geometry (Shen, 2006), for clustering categorical distributions (Nielsen & Sun, 2019), and in robotics (Ratliff et al., 2020). With regard

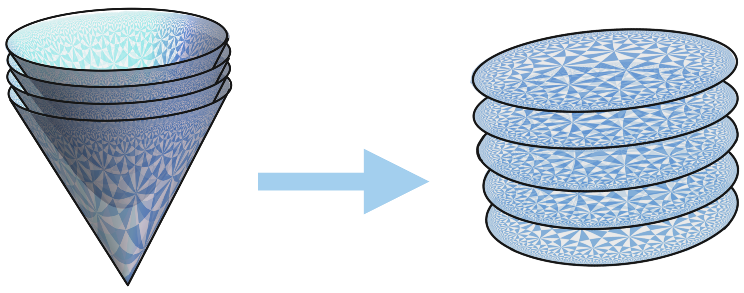


Figure 9.2: SPD_2 is foliated by hyperboloids, each of which is a copy of the hyperbolic plane.

to optimization, matrix backpropagation techniques have been explored (Ionescu et al., 2015), with some of them accounting for different Riemannian geometries (Huang & Gool, 2017; Brooks et al., 2019a). Nonetheless, we opt for tangent space optimization (Chami et al., 2019) by exploiting the explicit formulations of the exponential and logarithmic map, which enables us to use off-the-shelf optimizers.

9.2 SYMPA on the Space SPD_n

In this section we apply the SYMPA framework on the space of symmetric definite positive matrices. We follow the same steps outlined in the previous chapter to implement algorithms to compute distances and gradients.

9.2.1 Space and Model for SPD_n

The first steps of the SYMPA framework involve choosing a Riemannian symmetric space and a model of it. In this section we briefly recall the main properties of the space of symmetric positive definite matrices, already introduced in Chapter 2, and the representation model adopted.

The space SPD_n is a Riemannian manifold of non-positive curvature of $n(n+1)/2$ dimensions. Points in SPD_n are modelled as positive definite real symmetric $n \times n$ matrices, with the identity matrix I being a natural basepoint. The tangent space to any point of SPD_n can be identified with the vector space S_n of all real symmetric $n \times n$ matrices. SPD_n contains n-dimensional Euclidean subspaces, $(n-1)$-dimensional hyperbolic subspaces as well as products of $\lfloor \frac{n}{2} \rfloor$ hyperbolic planes.

In Figure 9.2 we visualize the smallest nontrivial example. SPD_2 identifies with the inside of a cone in $\mathbb{R}^3$, cut out by requiring both eigenvalues of the matrix $\left(\begin{smallmatrix} x & y \\ y & z \end{smallmatrix}\right)$ to be positive. It carries the product geometry of the hyperbolic plane times a line.

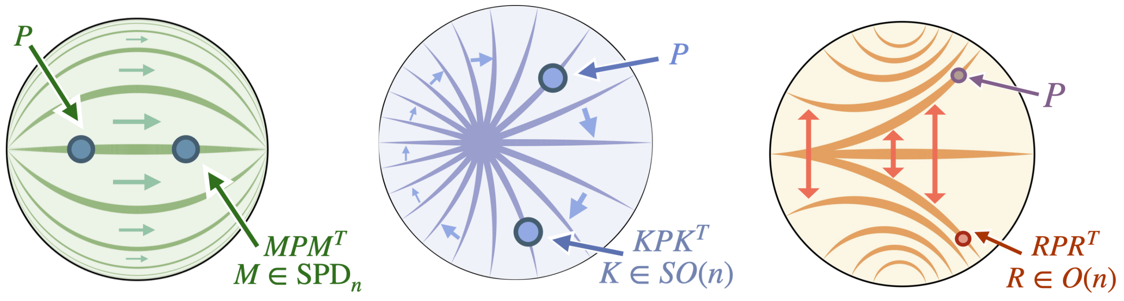


Figure 9.3: Some isometries of SPD_n have analogous Euclidean counterparts. Translation (left), rotation (center) and reflection (right).

Exponential and logarithmic maps: The exponential map, $\exp\colon S_n \to \mathrm{SPD}_n$, gives a connection between the Euclidean geometry of the tangent space S_n and the curved geometry of SPD_n. Its inverse is the logarithmic map, $\log\colon \mathrm{SPD}_n \to S_n$. Since we apply them based at $I \in \mathrm{SPD}_n$, this pair of functions forms a diffeomorphism that allows one to freely move between 'tangent space coordinates' or the original 'manifold coordinates'. We prove this in Appendix D.

Symmetries in SPD_n

The prototypical symmetries of SPD_n are parameterized by elements of $GL(n;\mathbb{R})$: any invertible matrix M defines the symmetry $P \mapsto MPM^T$ acting on all points $P \in \mathrm{SPD}_n$. Thus many geometric transformations SPD_n can be completed using standard optimized matrix algorithms as opposed to custom-built procedures. See Appendix D for a brief review of these symmetries.

Among these, we may find SPD_n-generalizations of familiar symmetries of Euclidean geometry. When also M is an element of SPD_n, the symmetry $P \mapsto MPM^T$ is a generalization of an Euclidean *translation*, fixing no points of SPD_n . When M is an orthogonal matrix, the symmetry $P \mapsto MPM^T$ is conjugation by M, and thus fixes the basepoint $I = MIM^T = MM^{-1} = I$. We think of elements fixing the basepoint as being SPD_n-*rotations* or SPD_n-*reflections*, when the matrix M is a familiar rotation or reflection (see Figure 9.3).

The Euclidean symmetry of *reflecting in a point* also has a natural generalization to SPD_n. Euclidean reflection in the origin is given by $p \mapsto -p$; and its SPD_n-analog, reflection in the basepoint I, is matrix inversion $P \mapsto P^{-1}$. The general SPD_n-reflection in a point $Q \in \mathrm{SPD}_n$ is a conjugate of this by an SPD_n translation, given by $P \mapsto QP^{-1}Q$.

9.2.2 Computing Distances in SPD_n

As we noticed in §9.1, several distances on SPD manifolds have been proposed, such as the Affine Invariant metric (Pennec et al., 2006), the Stein metric (Sra, 2012), the Bures–Wasserstein metric (Bhatia et al., 2019) or the Log-Euclidean metric (Arsigny et al., 2006a,b), with their respective formulas and geometric properties. Instead, the SYMPA framework provides us with the vector-valued distance function to compute a unique vector from where many distance metrics can be derived. In this section we implement the vector-valued distance function in SPD_n, and highlight some useful applications. In Appendix D, we provide a review of VVDs in SPD_n, and detail how the VVD generalizes the previous SPD metrics.

Vector-valued Distance Function

To briefly recall, in SPD the relative position between two points is determined by a vector, which we refer to as the vector-valued distance (VVD). Only if the VVD between two points $A, B \in \mathrm{SPD}_n$ is the vector $v \in \mathbb{R}^n$, and the VVD between $C, D \in \mathrm{SPD}_n$ is also v, then there exists an isometry mapping A to C and B to D.

To assign this vector in SPD_n we introduce the vector-valued distance function $d_{vv}\colon \mathrm{SPD}_n \times \mathrm{SPD}_n \to \mathbb{R}^n$, which assigns to two points a vector, instead of a scalar. For two points $P, Q \in \mathrm{SPD}_n$, the VVD is defined as:

$$d_{vv}(P, Q) = \log(\lambda_1(P^{-1}Q), \ldots, \lambda_n(P^{-1}Q)) \tag{9.1}$$

where $\lambda_1(P^{-1}Q) \geq \ldots \geq \lambda_n(P^{-1}Q)$ are the eigenvalues of $P^{-1}Q$ sorted in descending order.

As we said, the VVD contains much more information than just the distance. For example, given a representation of a graph in SPD_n we get finer invariants for the relative position between nodes of the graph. We leverage this information to visualize the learned high-dimensional representations in §9.5.7.

Riemannian and Finsler Metrics on SPD_n

Any norm on $\mathbb{R}^n$ derived from the VVD that is invariant under permutation of the entries induces a metric on SPD_n. As a consequence of this, SPD_n do not only support a Riemannian metric, but also Finsler metrics, a whole family of distances with the same symmetry group (group of isometries). As described in §8.1.2, these metrics are of special importance since distance minimizing geodesics are not necessarily unique in Finsler geometry. Two different paths can have the same minimal length. This is particularly valuable when embedding graphs in SPD_n, since in graphs there are generally several shortest paths. We obtain the Finsler metrics F_1 or F_∞ by taking the respective ℓ_1 or ℓ_∞

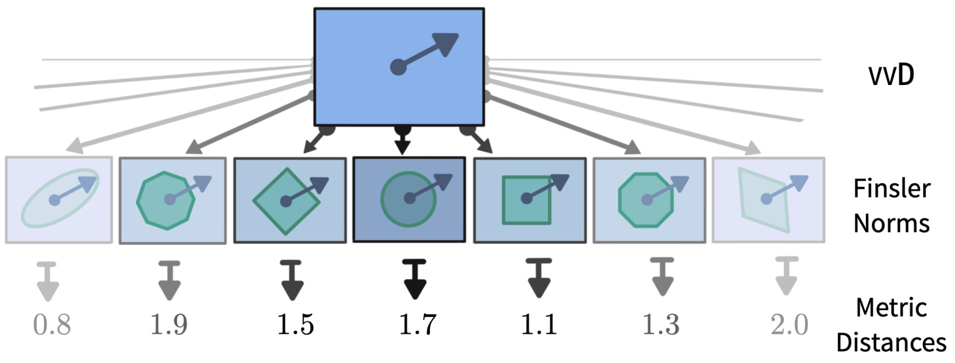


Figure 9.4: The vector-valued distance allows to reconstruct the Riemannian, or any Finsler distance.

norms of the VVD in $\mathbb{R}^n$ (see Figure 9.4). The Riemannian metric is obtained by using the standard l_2 norm on the VVD vector. This is: $d^R(P, Q) = ||d_{vv}(P, Q)||_2$. For a review of the theory of Finsler metrics in SPD_n see Appendix D.

9.2.3 Computing Gradients on SPD_n

To perform Riemannian gradient-based optimization (Bonnabel, 2011; Bécigneul & Ganea, 2019), the Riemannian gradient is required. Given $f : \mathrm{SPD}_n \to \mathbb{R}$ and $P \in \mathrm{SPD}_n$, the formula to compute the Riemannian gradient is:

$$\mathrm{grad}_{\mathrm{R}}(f(P)) = P \cdot \mathrm{grad}_{\mathrm{E}}(f(P)) \cdot P$$

where $\mathrm{grad}_E(f(P))$ is the Euclidean gradient at P of f obtained via automatic differentiation (Cruceru et al., 2020). However, in our experiments we adopt an alternative method based on tangent space optimization, described in §9.4.3.

9.3 Gyrocalculus on SPD_n

We presented a general introduction to gyrocalculus on gyrovector spaces in Chapter 2. Furthermore, we employed them to implement hyperbolic components in Chapter 7. In this section we develop the algebraic formalism of gyrocalculus adapted to the geometry of SPD_n spaces. We note that the SYMPA framework does not introduce any additional requisite for the gyrocalculus development, which can be seamlessly integrated into the model.

To build an analog of many Euclidean operators in SPD_n, we require also a translation of operations internal to Euclidean geometry, chief among these being the vector space operations of addition and scalar multiplication. We describe a gyrovector space structure

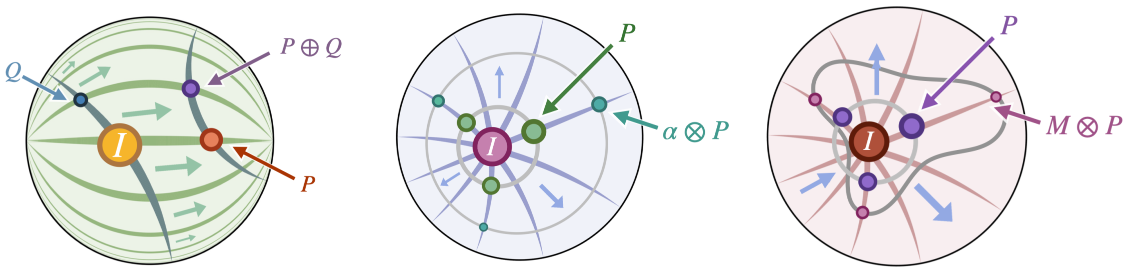


Figure 9.5: Gyro-addition (left), gyro-scalar multiplication (center) and matrix scaling (right).

on SPD_n, which provides geometrically meaningful extensions of these vector space operations. These operations provide a template for translation, where one may attempt to replace $+, -, \times$ in formulas familiar from Euclidean spaces with the analogous operations $\oplus, \ominus, \otimes$ on SPD_n.[1] While straightforward, such translation requires some care, as gyro-addition is neither commutative nor associative. See Appendix A for a review of the underlying general theory of gyrogroups and additional guidelines for accurate formula translation.

9.3.1 Addition and Subtraction

The gyrovector space structure of hyperbolic geometry exploited by Ganea et al. (2018b), Bachmann et al. (2020), and Shimizu et al. (2021) arises from physics, where $\oplus$ is the velocity addition operator in special relativity. This was given a purely geometric interepretation by Vermeer (2005) which directly generalizes to a candidate operation for $\oplus$ on SPD_n. Given a fixed choice I of basepoint and two points $P, Q \in \mathrm{SPD}_n$, we define the gyroaddition of P and Q to be the point $P \oplus Q \in \mathrm{SPD}_n$ which is the image of Q under the isometry which translates I to P along the geodesic connecting them (see Figure 9.5).

Fixing $P \in \mathrm{SPD}_n$, we may compute the value of $P \oplus Q$ for arbitrary Q as the result of applying the SPD_n-translation moving the basepoint to P, evaluated on Q. We see also that the additive inverse of a point with respect to this operation must then be given by its geodesic reflection in I:

$$P \oplus Q = \sqrt{P} Q \sqrt{P} \qquad \ominus P = P^{-1} \tag{9.2}$$

As this operation encodes a symmetry of SPD_n, it is possible to recast certain geometric statements purely in the gyrovector formalism. In particular, the vector-valued distance

[1]Throughout this chapter, the symbols $\oplus$ and $\otimes$ refer to operations developed on SPD_n and not on hyperbolic spaces, as in Chapter 7. We repeat the notation since the geometric meaning of the operation under the lens of the gyrocalculus formalism is the same, even though the space where they are defined differs.

$d_{vv}(P, Q)$ may be computed as the logarithm of the eigenvalues of $\ominus P \oplus Q$ (see Appendix D).

9.3.2 Scalar Multiplication and Matrix Scaling

For a fixed basepoint I, we define the scalar multiplication of a point $P \in \mathrm{SPD}_n$ by a scalar $\alpha \in \mathbb{R}_+$ to be the point which lies at distance $\alpha d(I, P)$ from I in the direction of P, where $d(\cdot, \cdot)$ is the metric distance on SPD_n. That is, we think of the operation $\alpha \otimes \cdot$ as geometrically analogous to standard scalar multiplication on $\mathbb{R}^n$: upon multiplication by α, each point of SPD_n is moved α times farther away from the basepoint I. Geometrically, this is a transfer of the vector-space scalar multiplication on the tangent space to SPD_n:

$$\alpha \otimes P = P^{\alpha} = \exp(\alpha \log(P)), \tag{9.3}$$

where $\exp, \log$ are the matrix exponential and logarithm. We further generalize the notion of scalar multiplication to allow for different relative expansion rates in different directions. For a fixed basepoint I and a point $P \in \mathrm{SPD}_n$, we can replace the scalar α from Equation 9.3 with an arbitrary real symmetric matrix $A \in S_n$. We define this *matrix scaling* by:

$$A \otimes P = \exp(A \odot \log(P)) \tag{9.4}$$

where $A \odot X$ denotes the Hadamard product. We denote the matrix scaling with $\otimes$, extending the previous usage: for any $\alpha \in \mathbb{R}$, we have $[\alpha] \otimes P = \alpha \otimes P$ where $[\alpha]$ is the matrix with every entry α.

9.4 Implementation

In this section we detail how we learn representations in SPD_n, and implement different linear mappings so that they conform to the premises of each operator. The proposed mappings, along with the gyrovector operations introduced in the previous section can be seen as feature transformations and employed as building blocks for SPD neural models.

9.4.1 Embeddings in SPD_n and S_n

We are interested in learning embeddings in SPD_n. To do so we exploit the connection between SPD_n and its tangent space S_n through the exponential and logarithmic maps. To learn an embedding $P \in \mathrm{SPD}_n$, we first model it as a symmetric matrix $U \in S_n$. We impose symmetry on U by learning a triangular matrix $X \in \mathbb{R}^{n \times n}$ with $n(n+1)/2$ parameters, such that $U = X + X^T$. To obtain the matrix $P \in \mathrm{SPD}_n$, we employ the exponential map: $P = \exp(U)$. Modeling embeddings on the tangent space offers advantages for optimization, explained in §9.4.3. For the matrix scaling $A \otimes P$, we impose symmetry on the factor matrix $A \in S_n$ in the same way that we learn the symmetric matrix U.

9.4.2 Isometries: Rotations and Reflections

Rotations in n dimensions are described as collections of pairwise orthogonal 2-dimensional rotations in planes (with a leftover 1-dimensional "axis of rotation" in odd dimensions). We utilize this observation to efficiently build elements of $O(n)$ out of two-dimensional rotations in coordinate planes. More precisely, for any $\theta \in [0, 2\pi)$ and choice of sign $\{+, -\}$ we let $R^{\pm}(\theta)$ denote the 2-dimensional rotation $(+)$ or reflection $(-)$ as $R^{\pm}(\theta) = \left(\begin{smallmatrix} \cos\theta & \mp\sin\theta \\ \sin\theta & \pm\cos\theta \end{smallmatrix}\right)$. Then for any pair $i < j$ in $1 \dots n$, we denote by $R^{\pm}_{ij}(\theta)$ the transformation which applies $R^{\pm}(\theta)$ to the $x_i x_j$-plane of $\mathbb{R}^n$, and leaves all other coordinates fixed. For example, in $O(5)$ the element $R^{+}_{24}(\theta)$ denotes the transformation where we replace the entries (ii, ij, ji, jj) of I_n with the corresponding values of $R^{+}(\theta)$:

$$R^{+}_{24}(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

More general, rotations and reflections are built by taking products of these basic transformations. Given a $^{n(n-1)}/_2$-dimensional vector of angles $\vec{\theta} = (\theta_{12}, \dots, \theta_{ij}, \dots,)$ and a choice of sign, we define the rotation and reflection corresponding to $\vec{\theta}$ by:

$$\mathrm{Rot}(\vec{\theta}) = \prod_{i<j} R^{+}_{ij}(\theta_{ij}) \qquad \mathrm{Ref}(\vec{\theta}) = \prod_{i<j} R^{-}_{ij}(\theta_{ij}) \tag{9.5}$$

where $\mathrm{Rot}(\vec{\theta}), \mathrm{Ref}(\vec{\theta}) \in \mathbb{R}^{n \times n}$ are the isometry matrices, and the vector of angles $\vec{\theta}$ can be regarded as a learnable parameter of the model. Finally, we denote the application of the transformation M to the point $P \in \mathrm{SPD}_n$ by:

$$M \circledcirc P = MPM^T \tag{9.6}$$

9.4.3 Optimization

For the proposed rotations and reflections, the learnable weights are vectors of angles $\vec{\theta} \in \mathbb{R}^{\frac{n(n-1)}{2}}$, which do not pose an optimization challenge. On the other hand, embeddings in SPD have to be optimized respecting the geometry of the manifold, but as explained in §9.4.1, we model them on the space of symmetric matrices S_n, and then we apply the exponential map. In this manner, we are able to perform tangent space optimization

(Chami et al., 2019) using standard Euclidean techniques, and circumvent the need for Riemannian optimization (Bonnabel, 2011; Bécigneul & Ganea, 2019), which we found to be less numerically stable. Due to the geometry of SPD_n (see Appendix D), this is an exact procedure, which does not incur losses in representational power.

9.4.4 Complexity

The most frequently utilized operation when learning graph embeddings is the distance calculation, thus we analyze its complexity. Calculating the distance between two points in SPD_n implies computing multiplications, inversions and diagonalizations of $n \times n$ matrices. We find that the cost of the distance computation with respect to the matrix dimensions is $\mathcal{O}(n^3)$. In Appendix D we detail the complexity of different operations.

We consider the computational complexity of working with spaces of matrices to be the main drawback, since the cost of many operations is polynomial instead of linear. However, when working on SPD_n a matrix of rank n implies $n(n+1)/2$ dimensions, thus a large n value is usually not required.

9.5 Knowledge Graph Completion

In this section we employ the transformations developed on SPD to build neural models for knowledge graph completion, that we apply in three different setups. Task-specific models in different geometries have been developed in the three cases, hence we consider them adequate benchmarks for representation learning.

9.5.1 Problem Formulation

Knowledge graphs represent heterogeneous knowledge in the shape of *(head, relation, tail)*. Given an incomplete KG, the task is to predict which unknown links are valid.

More formally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ be a knowledge graph where $\mathcal{V}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{E} \subset \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is the set of triples stored in the graph. The usual approach is to learn a scoring function $\phi : \mathcal{V} \times \mathcal{R} \times \mathcal{V} \to \mathbb{R}$ that measures the likelihood of a triple to be true, with the goal of scoring all missing triples correctly. To do so, we propose to learn representations of entities as embeddings in SPD_n, and relation-specific transformation in the manifold, such that the KG structure is preserved.

9.5.2 Data

For experiments and analysis we employ two standard benchmarks for knowledge base completion, namely WN18RR (Bordes et al., 2013; Dettmers et al., 2018) and FB15k-237

	WN18RR	**FB15k-237**
Entities	40,943	14,541
Relations	11	237
Triples:		
- Train	86,835	272,115
- Dev	3,034	17,535
- Test	3,134	20,466
δ-mean	0.415	0.151
δ-max	3	1.5
Sectional Curvature	-0.58±0.44	0.16±0.37

Table 9.1: Statistics of the knowledge graph datasets.

(Bordes et al., 2013; Toutanova & Chen, 2015). WN18RR is a subset of WordNet (Miller, 1992) containing 11 lexical relationships between $40,943$ word senses. FB15k-237 is a subset of Freebase (Bollacker et al., 2008), a collaborative knowledge base of general world knowledge, with $14,541$ entities and 237 relationships.

Graph Analysis

To analyze the knowledge graphs we rely on the sectional curvature and the δ-hyperbolicity. We do not plot the Ollivier-Ricci curvature due to the large size of the graphs.

In Table 9.1 we can see relevant values to the graph analysis, together with other statistics about the datasets. We observe that according to the δ-hyperbolicity, FB15k-237 seems to be more hyperbolic-like than WN18RR. However, we plot the distribution of the sectional curvature over both knowledge graphs in Figure 9.6. We can see that for WN18RR, the curvature distribution is much more negative, suggesting that the graph contains more regions that fit better on hyperbolic subspaces. On the other hand, the sectional curvature of FB15k-237 is closer to zero, which indicates that the connectivity

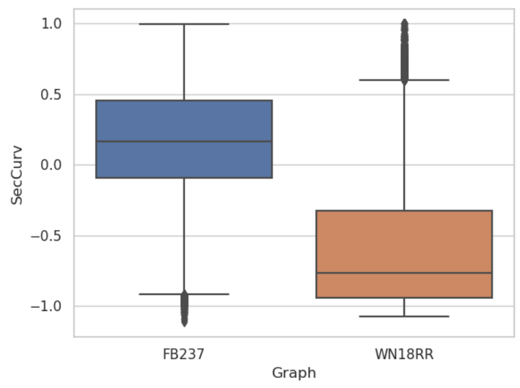


Figure 9.6: Distribution of sectional curvature for the two analyzed datasets.

of the graph exhibits flat areas that can be accommodated in Euclidean subspaces. Given the diverse features that these graphs exibit, we hypothesize that a model operating on the SPD manifold, which combines both, Euclidean and hyperbolic subspaces, will improve the performance.

9.5.3 Related Work for Knowledge Graph Completion

In Chapter 5 we covered methods graph embeddings where edges are associated with a weight. In this section we review related work for knowledge graph embeddings that account for edges as labeled relations with diverse semantic meaning.

Most of the KG embedding methods learn vectors $\mathbf{h}, \mathbf{t} \in \mathbb{U}^{n_\mathcal{V}}$ for $h, t \in \mathcal{V}$, and $\mathbf{r} \in \mathbb{U}^{n_\mathcal{R}}$ for $r \in \mathcal{R}$, for some choice of space $\mathbb{U}$, typically $\mathbb{R}$. Recent approaches propose to embed the graph into non-Euclidean geometries such as hyperbolic spaces (Balazevic et al., 2019; Kolyvakis et al., 2020; Chami et al., 2020c), to model embeddings over the complex numbers $\mathbb{C}$ (Trouillon et al., 2016; Lacroix et al., 2018; Sun et al., 2019), or to apply quaternion algebra (Zhang et al., 2019). We describe a subset of these methods that combine different operators and achieve state-of-the-art performance on KG completion tasks.

TransE: Introduced by Bordes et al. (2013), it models entities as low-dimensional vectors and represent each relation as a single vector that linearly interacts with the entity vectors. The scoring function is defined as:

$$\phi(h, r, t) = -d_\mathbb{R}(\mathbf{h} + \mathbf{r}, \mathbf{t}), \qquad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^n$$

RotC: Sun et al. (2019) map entities and relations to the complex vector space $\mathbb{C}^n$ and defines each relation as a rotation in the complex plane from the source entity to the target entity. Given a triple (h, r, t), it is expected that $\mathbf{t} \approx \mathbf{h} \circ \mathbf{r}$, where $\circ$ denotes the Hadamard (element-wise) product. Rotations are chosen since they can simultaneously model and infer inversion, composition, symmetric or anti-symmetric patterns.

MuRP: By establishing a comparison with word analogies through hyperbolic distances (Tifrea et al., 2019), Balazevic et al. (2019) propose a scoring function based on relation-specific Möbius multiplication on the head entity, and Möbius addition (Ganea et al., 2018b) on the tail entity:

$$\phi(h, r, t) = -d_\mathbb{H}(M_r \otimes_\mathbb{H} \mathbf{h}, \mathbf{t} \oplus_\mathbb{H} \mathbf{r})^2 + b_h + b_t, \qquad \mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{H}^n$$

where $b_h, b_t \in \mathbb{R}$ are scalar biases for the head and tail entities respectively, and $d_\mathbb{H}$ is the hyperbolic distance. We distinguish the fact that the multiplication and addition are performed on hyperbolic space with the sub-index $\mathbb{H}$. MuRE is the alternative model that replaces the operations for their Euclidean equivalent.

RotH and RefH: Chami et al. (2020c) extend MuRP with rotations and reflections in hyperbolic space by learning relationship-specific isometries through Givens transformations.[2] The result of these operations is combined with an attention mechanism in the tangent space (Chami et al., 2019). They also present RotE and RefE with the respective rotations and reflections performed on Euclidean space.

9.5.4 Proposed Models

Based on the developed operations and neural components, we propose two models that extend successful approaches for knowledge base completion

Scaling model: We follow the base hyperbolic model MuRP and adapt it into SPD_n by means of the *matrix scaling*. Its scoring function has shown success in the task given that it combines multiplicative and additive components, which are fundamental to model different properties of KG relations (Allen et al., 2021). We translate it into SPD_n as:

$$\phi(h, r, t) = -d((\mathbf{M}_r \otimes \mathbf{H}) \oplus \mathbf{R}, \mathbf{T})^2 + b_h + b_t \tag{9.7}$$

where $\mathbf{H}, \mathbf{T} \in \mathrm{SPD}_n$ are embeddings and $b_h, b_t \in \mathbb{R}$ are scalar biases for the head and tail entities respectively. $\mathbf{R} \in \mathrm{SPD}_n$ and $\mathbf{M}_r$ are matrices that depend on the relation. For $d(\cdot, \cdot)$, we experiment with the Riemannian and the Finsler One metric distances.

Isometric model: A possible alternative is to embed the relation-specific transformations as elements of the $O(n)$ group (*i.e.*, rotations and reflections). This technique has proven effective in different metric spaces (Yang et al., 2020; Chami et al., 2020c). In this case, $\mathbf{M}_r$ is a rotation or reflection matrix as in Equation 9.5, and the scoring function is defined as:

$$\phi(h, r, t) = -d((\mathbf{M}_r \circledcirc \mathbf{H}) \oplus \mathbf{R}, \mathbf{T})^2 + b_h + b_t \tag{9.8}$$

9.5.5 Experimental Setup

All models and experiments were implemented in PyTorch (Paszke et al., 2019) with distributed data parallelism, for high performance on clusters of CPUs/GPUs.

[2]https://en.wikipedia.org/wiki/Givens_rotation

Training: We follow the standard data augmentation protocol by adding inverse relations to the datasets (Lacroix et al., 2018). We optimize the cross-entropy loss with uniform negative sampling defined as:

$$\mathcal{L} = \sum_{(h,r,t)\in\mathcal{T}} \log(1 + \exp(Y_t \phi(h,r,t))) \tag{9.9}$$

where $\mathcal{T}$ is the set of training triples, and $Y_t = -1$ if t is a factual triple or $Y_t = 1$ if t is a negative sample. We employ the AdamW optimizer (Loshchilov & Hutter, 2019). To select optimal hyper-parameters, we conduct a grid search using the validation set. We experiment with matrices of dimension $n \times n$ where $n \in \{14, 20, 24\}$ (this is the equivalent of $\{105, 210, 300\}$ degrees of freedom respectively) , learning rates from $\{1\mathrm{e}{-4}, 5\mathrm{e}{-5}, 1\mathrm{e}{-5}\}$ and weight decays of $\{1\mathrm{e}{-2}, 1\mathrm{e}{-3}\}$. In all cases we train for 5000 epochs, with batch size of 4096 and 10 negative samples.

Evaluation Metrics: At test time, we rank the correct tail or head entity against all possible entities using the scoring function, and use inverse relations for head prediction (Lacroix et al., 2018). Following previous work, we compute two ranking-based metrics: mean reciprocal rank (MRR), which measures the mean of inverse ranks assigned to correct entities, and hits at K (H@K, $K \in \{1, 3, 10\}$), which measures the proportion of correct triples among the top K predicted triples. We follow the standard evaluation protocol of filtering out all true triples in the KG during evaluation, since predicting a high rank for these triples should not be penalized (Bordes et al., 2013).

Baselines: We compare our models with their respective equivalents in different metric spaces, which are also state-of-the-art models for the task. For the scaling model, these are MuRE and MuRP (Balazevic et al., 2019), which perform the scaling operation in Euclidean and hyperbolic space respectively. For the isometric models, we compare to RotC (Sun et al., 2019), RotE and RotH, (Chami et al., 2020c) (rotations in Complex, Euclidean and hyperbolic space respectively), and RefE and RefH (Chami et al., 2020c) (reflections in Euclidean and hyperbolic space). Baseline results are taken from the original papers.

We do not compare our implementation with previous work on SPD representation learning methods due to the fact that experiments with knowledge graph embedding models require arithmetic operations in the space, such as addition, which are not defined in previous work. Thus, a vis-a-vis comparison of an equivalent model is not possible. Moreover, the definition of the transformation layers employed in Dong et al. (2017), Gao et al. (2019), or Huang & Gool (2017) requires optimizing over compact Stiefel manifolds, plus the derivation of a particular Riemannian matrix backpropagation rule, which is a

		WN18RR				FB15k-237			
Operation	Model	MRR	HR@1	HR@3	HR@10	MRR	HR@1	HR@3	HR@10
Scaling	MuRE	47.5	43.6	48.7	55.4	33.6	24.5	37.0	52.1
	MuRP	48.1	**44.0**	49.5	56.6	33.5	24.3	36.7	51.8
	$\mathrm{SPD}^{R}_{\mathrm{Sca}}$	48.1	43.1	50.1	57.6	**34.5**	**25.1**	**38.0**	**53.5**
	$\mathrm{SPD}^{F_1}_{\mathrm{Sca}}$	**48.4**	42.6	**51.0**	**59.0**	32.9	23.6	36.3	51.5
Rotations	RotC	47.6	42.8	49.2	57.1	33.8	24.1	37.5	53.3
	RotE	49.4	44.6	51.2	58.5	**34.6**	**25.1**	**38.1**	**53.8**
	RotH	**49.6**	**44.9**	**51.4**	**58.6**	34.4	24.6	38.0	53.5
	$\mathrm{SPD}^{R}_{\mathrm{Rot}}$	46.2	39.7	49.6	57.8	32.9	23.6	36.3	51.6
	$\mathrm{SPD}^{F_1}_{\mathrm{Rot}}$	40.9	30.5	48.2	57.3	32.1	22.9	35.4	50.5
Reflections	RefE	47.3	43.0	48.5	56.1	**35.1**	**25.6**	**39.0**	**54.1**
	RefH	46.1	40.4	48.5	56.8	34.6	25.2	38.3	53.6
	$\mathrm{SPD}^{R}_{\mathrm{Ref}}$	48.3	44.0	49.7	56.7	32.5	23.4	35.6	51.0
	$\mathrm{SPD}^{F_1}_{\mathrm{Ref}}$	**48.7**	**44.3**	**50.1**	**57.4**	31.6	22.5	34.6	50.0

Table 9.2: Results for Knowledge graph completion.

highly non-trivial and impractical approach, whereas our implementation employs off-the-shelf optimizers.

9.5.6 Results

We report the performance for all analyzed models, segregated by operation, in Table 9.2. On both dataset, the scaling model $\mathrm{SPD}_{\mathrm{Sca}}$ outperforms its direct competitors MuRE and MuRP, and this is specially notable in HR@10 for WN18RR: 59.0 for $\mathrm{SPD}^{F_1}_{\mathrm{Sca}}$vs 55.4 and 56.6 respectively. SPD reflections are very effective on WN18RR as well. They outperform their Euclidean and hyperbolic counterparts RefE and RefH, in particular when equipped with the Finsler metric. Rotations on the SPD manifold, on the other hand, seem to be less effective. However, Euclidean and hyperbolic rotations require 500 dimensions whereas the $\mathrm{SPD}_{\mathrm{Rot}}$ models are trained on matrices of rank 14 (equivalent to 105 dims).

Regarding the choice of a distance metric, the Finsler One metric is better suited with respect to HR@3 and HR@10 when using scalings and reflections on WN18RR. For the FB15k-237 dataset, SPD models operating with the Riemannian metric outperform their Finsler counterparts. This suggests that the Riemannian metric is capable of disentangling the large number of relationships in this dataset to a better extent.

For the FB15k-237 dataset, we observe a prevalence of spaces that can model Euclidean features. This is, the Euclidean rotations are reflections are the best, whereas for scaling, the SPD manifold, which contains Euclidean subspaces, outperforms its competitors. We consider this result in line with the sectional curvature reported for this dataset in §9.5.2. Moreover, the WN18RR dataset exhibits a negative sectional curvature, and we can see that the hyperbolic or SPD models are the best in all cases for that KG. One more time,

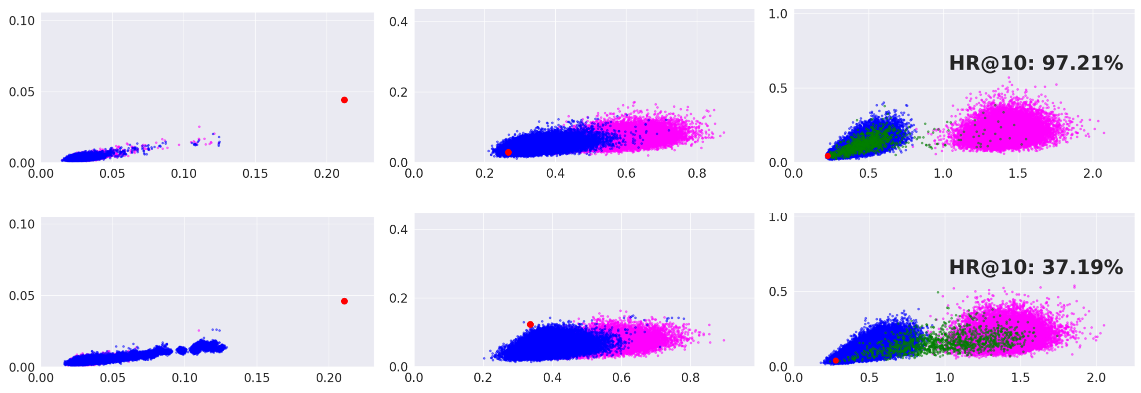


Figure 9.7: Train, negative and validation triples for relationships *'derivationally related form'* (top) and *'hypernym'* (bottom) for 5 (left), 50 (center) and 3000 (right) epochs for the $\mathrm{SPD}^{F_1}_{\mathrm{Sca}}$ model. The red dot corresponds to the relation addition $\mathbf{R}$.

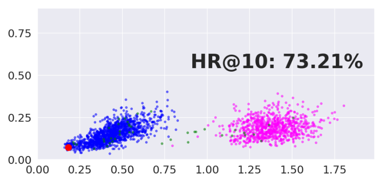
aligning the embedding space to the topological characteristics of the data results in an effective geometric inductive bias that enhances the performance.

In these experiments we have evaluated models applying equivalent operations and scoring functions in different geometries for the task of embedding a knowledge graph, thus they can be thought as a vis-a-vis comparison of the metric spaces. We observe that SPD models tie or outperform baselines in most instances. This showcases the improved representation capacity of the SPD manifold for multi-relational graphs, particularly when compared to Euclidean and hyperbolic spaces. Moreover, it demonstrates the effectiveness of the proposed metrics and operations in this manifold.

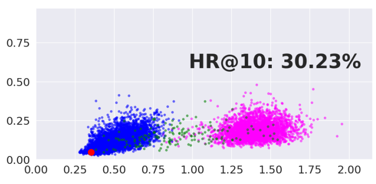
9.5.7 Visualizations through the VVD

One reason to embed data into Riemannian manifolds, such as SPD, is to use geometric properties of the manifold to analyze the structure of the data. Visualizations in SPD are difficult due to their high dimensionality. As a solution we use the vector-valued distance function to develop a new tool to visualize and analyze structural properties of the learned representations.

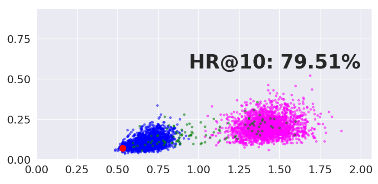
We adopt the vector $(n-1, n-3, \cdots, -n+3, -n+1)$, as the barycenter of the space in $\mathbb{R}^n$ where the VVD is contained. Then, we plot the norm of the VVD vector and its angle with respect to this barycenter. In Figure 9.7, we compute and plot the VVD corresponding to $d(\mathbf{M}_r \otimes \mathbf{H}, \mathbf{T})$ and $\mathbf{R}$ as defined in Equation 9.7 for KG models trained on WN18RR. In early stages of the training, all points fall near the origin (left side of the plots). As training evolves, the model learns to separate true (h, r, t) triples from corrupted ones (center part). When the training converges, the model is able to clearly disentangle and cluster positive and negative samples. We observe how the position of the validation triples (green points, not seen during training) directly correlates with the performance of each relation. Plots for more relations can be seen in Figure 9.8.



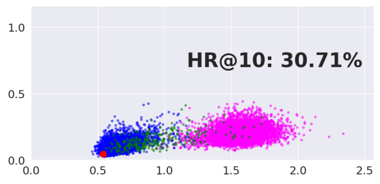
(a) *'Also see'*



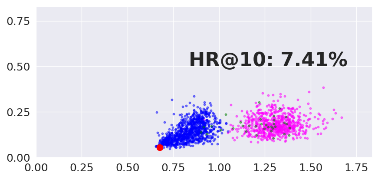
(b) *'Has part'*



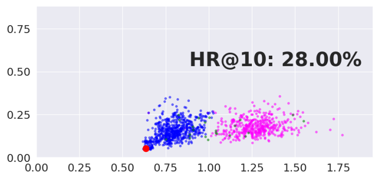
(c) *'Instance hypernym'*



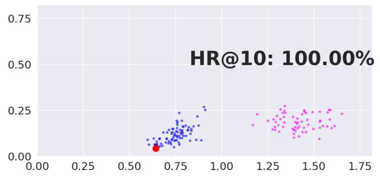
(d) *'Member meronym'*



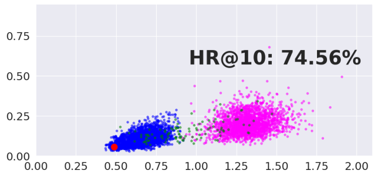
(e) *'Member of domain region'*



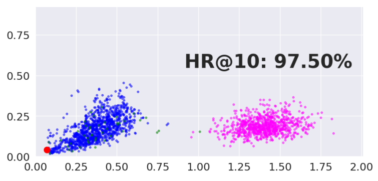
(f) *'Member of domain usage'*



(g) *'Similar to'*



(h) *'Synset domain topic of'*



(i) *'Verb group'*

Figure 9.8: Train, negative and validation triples for WN18RR relationships of the $\mathrm{SPD}^{F_1}_{\mathrm{Sca}}$ model after convergence. The red dot corresponds to the relation addition $\mathbf{R}$.

9.6 Knowledge Graph-based Recommender Systems

Recommender systems (RS) estimate users' preferences for items to provide personalized recommendations and a better user experience. The underlying assumption is that users may be interested in items selected by people who share similar interactions with them. Mathematically, a recommendation method can be posed as a matrix completion problem (Candès & Recht, 2012), where columns and rows represent users and items, respectively, and matrix values represent a score determining whether a user would like an item or not. From a geometric perspective, the recommendation problem can be modelled as a link prediction task over a graph of users and items (Li et al., 2014).

KG embedding methods have been widely adopted into the recommendation problem as an effective tool to model side information, which helps to alleviate data sparsity and enhances the performance (Zhang et al., 2016; Guo et al., 2020). For instance, one reason for recommending a movie to a particular user is that the user has already watched many movies from the same genre or director (Ma et al., 2019). Given multiple relations between users, items, and heterogeneous entities, the goal is to predict the user's next item purchase or preference.

We adopt the approach of modelling the recommendations as a link prediction task. In addition, we aim to incorporate side information between users, items and other entities. Hence we apply our KG embedding method from the previous section as is, to embed this

Dataset	**Users**	**Items**	**Other Entities**	**Train Relations**		**Dev/Test**
				User-item	**Others**	
Software	1,826	802	689	8,242	6,078	1,821
Luxury Beauty	3,819	1,581	2	20,796	26,044	3,639
Prime Pantry	14,180	4,970	1,100	102,848	99,118	14,133
MindReader	961	2,128	11,775	11,279	99,486	953

Table 9.3: Statistics for KG Recommender datasets.

multi-relational graph. We evaluate the capabilities of the approach by only measuring the performance over user-item interactions.

9.6.1 Data

To investigate the recommendation problem endowed with added relationships, we employ the Amazon dataset (McAuley & Leskovec, 2013; Ni et al., 2019) with users' purchases of products, and the MindReader dataset (Brams et al., 2020) of movie recommendations. To generate evaluation splits, the penultimate and last item the user has interacted with are withheld as dev and test sets respectively. We only consider users with 3 or more interactions. Statistics of the datasets with the added relationships can be seen in Table 9.3.

Amazon Dataset: We adopt the 5-core split for the branches "Software", "Luxury & Beauty" and "Prime Pantry", which form a diverse dataset in size and domain. We add relationships used in previous work (Zhang et al., 2018b; Ai et al., 2018). These are:

- *also_bought*: users who bought item A also bought item B.
- *also_view*: users who bought item A also viewed item B.
- *category*: the item belongs to one or more categories.
- *brand*: the item belongs to one brand.

MindReader Dataset: We consider a user-item interaction when a user gave an explicit positive rating to the movie. The relationships added are:

- *directed_by*: the movie was directed by this person.
- *produced_by*: the movie was produced by this person/company.
- *from_decade*: the movie was released in this decade.
- *followed_by*: the movie was followed by this other movie.

	SOFTWARE		LUXURY		PANTRY		MINDREADER	
Model	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
TRANSE	28.5±0.1	47.2±0.5	35.6±0.1	52.3±0.1	16.6±0.0	35.3±0.1	19.1±0.4	37.6±0.1
ROTC	28.5±0.3	45.4±1.4	33.0±0.1	49.8±0.2	14.5±0.0	31.3±0.2	25.3±0.3	50.3±0.6
MURE	29.4±0.4	47.1±0.4	35.6±0.7	54.0±0.3	19.4±0.1	39.5±0.2	25.2±0.3	49.9±0.6
MURP	29.6±0.3	47.9±0.3	**37.5±0.1**	**55.2±0.3**	19.4±0.1	39.8±0.2	25.3±0.3	49.3±0.2
$\text{SPD}^{R}_{\text{Sca}}$	29.4±0.4	48.1±0.8	**37.5±0.2**	55.1±0.2	19.5±0.0	39.6±0.3	25.4±0.1	49.8±0.3
$\text{SPD}^{F_1}_{\text{Sca}}$	28.8±0.1	46.9±0.5	37.3±0.3	54.1±0.9	19.0±0.1	38.8±0.2	**25.7±0.5**	49.5±0.1
$\text{SPD}^{R}_{\text{Rot}}$	**30.3±0.2**	48.6±0.9	37.2±0.1	54.8±0.4	**20.0±0.1**	**40.3±0.1**	25.3±0.0	**50.5±0.3**
$\text{SPD}^{F_1}_{\text{Rot}}$	30.1±0.1	**49.1±0.3**	36.9±0.1	54.5±0.6	19.2±0.0	39.3±0.1	**25.7±0.0**	49.5±0.2
$\text{SPD}^{R}_{\text{Ref}}$	29.6±0.2	48.0±0.5	37.3±0.2	55.0±0.2	19.3±0.0	39.7±0.3	25.3±0.0	49.1±0.1
$\text{SPD}^{F_1}_{\text{Ref}}$	29.3±0.1	47.5±0.6	36.8±0.0	54.8±0.1	18.6±0.2	38.3±0.3	24.8±0.2	47.9±1.8

Table 9.4: Results for Knowledge graph-based recommender systems.

- *has_genre*: the movie belongs to this genre.
- *has_subject*: the movie has this subject.
- *starring*: the movie was starred by this person.

9.6.2 Experimental Setup

Training: In this setup we also augment the data by adding inverse relations and optimize the loss from Equation 9.9. We set the size of the matrices to 10×10 dimensions (equivalent to 55 free parameters). We conduct a grid search to select optimal batch size and learning rate, using the validation set. We report the average of 3 runs.

Evaluation Metrics: Since it is very costly to rank all the available items, we follow the standard procedure of evaluating against 100 randomly selected samples the user has not interacted with (He et al., 2017). To evaluate the recommendation performance we focus on the *buys / likes* relation. For each user u we rank the items i_j according to the scoring function $\phi(u, buys, i_j)$. We adopt MRR and H@10, as ranking metrics for recommendations.

Baselines: We compare to TransE (Bordes et al., 2013), RotC (Sun et al., 2019), MuRE and MuRP (Balazevic et al., 2019) trained with 55 dimensions.

9.6.3 Results

In Table 9.4 we observe that the SPD models tie or outperform the baselines in both MRR and HR@10 across all analyzed datasets. Rotations in both, Riemannian and Finsler metrics, are more effective in this task, achieving the best performance in 3 out of 4 cases, followed by the scaling models. Overall, this shows the capabilities of the systems to effectively represent user-item interactions enriched with relations between items and their attributes, thus learning to better model users' preferences. Furthermore, it displays the versatility of the approach to diverse data domains.

9.7 Conclusions

In this chapter we model multi-relational graphs on the space of SPD matrices. We choose this manifold given its high expressivity and rich geometry, which encompasses Euclidean as well as hyperbolic spaces. To learn graph representations on this Riemannian symmetric space, we apply the SYMPA framework. The general view of the framework allows us to introduce the vector-valued distance function in this space, which we exploit to implement universal models (including Finsler distances and generalizing previous metrics on SPD), and to provide a geometric interpretation on what the models learn.

Moreover, we bridge the gap between Euclidean and SPD geometry under the lens of the gyrovector theory, providing means to transfer standard arithmetic operations from the Euclidean setting to their analog notions in SPD. These tools enable practitioners to exploit the full representation power of SPD, and profit from the enhanced expressivity of this manifold.

We propose and evaluate SPD models on two tasks and six datasets, which showcases the versatility of the approach and ease of integration with downstream tasks. The results reflect the superior expressivity of SPD when compared to Euclidean or hyperbolic baselines. Nevertheless, this work is not without limitations. We consider the computational complexity of working with spaces of matrices to be the main drawback, since the cost of many operations is polynomial instead of linear.

Part IV

Conclusions

Chapter 10

Conclusions and Future Work

"We ultimately pursue, not conclusions, but beginnings."
– Sam Tanenhaus

10.1 Conclusions

In this thesis, we have advocated for non-Euclidean Riemannian manifolds as target embedding spaces for learning neural graph representations. Our premise is that by aligning the target space with the topology of the graph-structured data, we introduce a geometric inductive bias that guides models to achieve higher performance with a reduced parameter footprint. Furthermore, we showcase the integration and utility of the learned embeddings in different downstream tasks.

Initially, we developed different techniques to derive hierarchical information from large label inventories. We incorporated it into an NLP model through hyperbolic graph embeddings, leading to an enhanced performance due to the hierarchical structure of the data.

Next, we introduced a model that jointly learns task-specific graph embeddings from a label inventory and performs multi-class multi-label classification in hyperbolic space. This model achieved a classification performance comparable to state-of-the-art systems on very fine-grained labels with a remarkable reduction of the parameter size.

Third, we proposed SYMPA, a general framework to embed graphs on symmetric spaces. Our framework enables practitioners to choose a Riemannian symmetric space and implement the mathematical tools required to learn graph embeddings. We demonstrated a concrete implementation of the framework on Siegel spaces, and showcased their versatility on different tasks with graphs of very diverse structures.

Finally, we devised the means to translate Euclidean and hyperbolic multi-relational graph embedding models into the space of symmetric positive definite (SPD) matrices. To do so we developed gyrocalculus in this geometry, and integrated it with the SYMPA framework.

Overall, our aim has been to utilize alternative representation methods with well-established mathematical foundations, that can be integrated into neural pipelines from diverse domains. We hope that this work eases the adoption of non-Euclidean components into Deep Learning models, yielding lightweight and efficient systems with strong geometric priors.

10.2 Future Research Directions

Michael: *We have to think what is going to be fashionable in three years.*
Federico: *So, do we need to predict the future?*
Michael: *No, we don't predict the future. We create it.*
– On deciding a PhD topic. April, 2018.

Our work on Geometric Deep Learning opens several avenues for future research.

- **Development of deep neural network architectures adapted to the geometry of Riemannian symmetric spaces:** To continue the work done in this thesis, and in line with the previous work described in §9.1, it would be particularly useful to focus on matrix manifolds such as Siegel or SPD spaces. In this manner, more expressive feature transformations can be implemented. Furthermore, adapting classification objectives to these geometries would allow models to be trained on a richer set of tasks, similar to the multi-logistic regression proposed in Ganea et al. (2018b).

- **Supervised approaches for graphs based on Graph Neural Networks (GNN) operating on non-Euclidean Manifolds:** In this regard, there is already a strand of research adapting GNNs to hyperbolic geometry (Chami et al., 2019; Liu et al., 2019; Bachmann et al., 2020). This could be further extended to other particular cases, such as Siegel and SPD spaces. Or in a more general case, a framework for adapting *message-passing* GNNs (Gilmer et al., 2017) to a Riemannian symmetric space of choice.

- **Curvature learning on Siegel spaces:** Since the Siegel space embeds as a subspace of its compact dual, there is an explicit geometric transition, generalizing the connection between spherical and hyperbolic geometry (see Appendix C). A possible research direction is to use the geometric transition between symmetric spaces to extend the curvature learning approach exploited by Gu et al. (2019) and Chami et al. (2019).

- **Mapping Euclidean data to different geometries:** There are massive amounts of information already encoded under Euclidean assumptions in pre-trained models, such as BERT and many other language models (Reif et al., 2019). Re-training them on different vector spaces or geometries might be unfeasible. A valuable addition would be to develop methods to map a set of points from Euclidean space onto a Riemannian manifold of choice while preserving the structure. An approach in this line, focused on retaining topological features, has been proposed by Moor et al. (2020).

- **Visualization tools based on the vector-valued distance function:** As already stated, the VVD contains much more information than just the distance and it can be leveraged to visualize what models learn. This can be exploited to infer different properties of the structure of the graph, complementing the tools already developed in §8.4.5, and the toolkit of curvature measures employed in this work (§3.2). Moreover, interactive visualizations which provide further understanding on the impact of the chosen distance function (Riemannian or any of the Finsler ones) would also be interesting to pursue.

Part V

Appendices

Appendix A

Gyrocalculus

A primary difficulty of building analogs of Euclidean quantities in curved spaces is the lack of a vector space structure, making the translation of operations like vector addition or scalar multiplication difficult to immediately interpret. The need for these is already well-noted stumbling block in hyperbolic geometry, as any algorithm using the Euclidean addition of points cannot be implemented directly (for example considering the Poincare disk model, the sum of two points in the disk need not lie in the disk: and even when it does, the result is rarely geometrically meaningful). To combat this, means of interfacing with hyperbolic geometry using "vector-space-like" operations was developed by Ungar (2008a), which provides an analog of addition $\oplus\colon \mathbb{H}^n \times \mathbb{H}^n \to \mathbb{H}^n$ and of scalar multiplication $\otimes\colon \mathbb{R} \times \mathbb{H}^n \to \mathbb{H}^n$ called ‘gyro-addition’ and ’gyro-scalar multiplication’ respectively. We give a brief introduction to this general theory below, see Ungar’s treatment from the lens of differential geometry (Ungar, 2005) for further information.

Gyrogroups

Gyrogroups are a generalization of groups which encode algebraically some of the geometric properties of symmetric spaces. More precisely, a gyrogroup structure on a set G is given by a binary operation $\oplus$, which is assumed to have an identity element $0 \in G$ and left inverses $\ominus g$ for each $g \in G$. Keeping with the conventions familiar from arithmetic, we write $a \ominus b$ to mean $a \oplus (\ominus b)$. The crucial difference from group theory is that $\oplus$ is *not* required to be associative. Instead, the additional structure of a *gyration operator* $\mathrm{gyr}\colon G \times G \to \mathrm{Aut}(G)$ captures the nonassociativity of $\oplus$ by

$$a \oplus (b \oplus c) = (a \oplus b) \oplus \mathrm{gyr}(a, b)c$$

.

For $(G, \oplus, \mathrm{gyr})$ to form a gyrogroup, an additional axiom is imposed on this gyration, namely that it satisfy the *left loop identity*, $\operatorname{gyr}(a,b) = \operatorname{gyr}(a \oplus b, b)$.

Gyrogroups generalize groups in the sense that every group G is a gyrogroup with its usual binary operation as $\oplus$, and trivial gyration. As with standard groups, it is helpful to have at one's disposal a collection of elementary deductions from these axioms, which may significantly simplify further calculations.

Proposition A.1. The identity of a gyrogroup is unique, every left inverse is also a right inverse, and every element has a unique (left, and hence also right) inverse.

See Ungar (2005) §3 for a proof of this proposition, which uses only the axioms of a gyrogroup. It can be shown that when a gyrogroup structure exists on a set G, it is determined by the operation $\oplus$ alone, in the sense that for any a, b, c we have

$$\operatorname{gyr}(a,b)c = (\ominus (a \oplus b)) \oplus (a \oplus (b \oplus c)) \tag{A.1}$$

We record also useful properties of the gyration operator following from this, which simplify calculation.

Proposition A.2. The following gyrations are trivial: the gyration of any element with zero $\operatorname{gyr}(0,a) = \operatorname{gyr}(0,a) = \operatorname{gyr}(\ominus a, a) = \mathrm{id}_G$, or with its inverse $\operatorname{gyr}(\ominus a, a) = \operatorname{gyr}(a, \ominus a) = \mathrm{id}_G$. A useful consequence of these is the *nested gyration identity*:

$$\operatorname{gyr}(a, \ominus \operatorname{gyr}(a,b)b) \operatorname{gyr}(a,b) = \mathrm{id}_G$$

These are also proven in Ungar (2005) §3 , and follow directly from the axioms of a gyrogroup.

Because of the additional complexity of $\oplus$ compared to the binary operation of a standard group, it is often useful in applications to introduce a second binary operation, the *gyrogroup co-operation* $\boxplus$ and its inverse $\boxminus$, defined by

$$a \boxplus b = a \oplus \operatorname{gyr}(a, \ominus b)b \qquad a \boxminus b = a \boxplus \ominus a$$

This operation provides a useful shorthand for solving equations in gyrogroups, which we discuss further down in this Appendix.

More on Gyrovector Spaces

Though the operation $\oplus$ is not commutative in the usual sense, a gyrogroup G is called *gyrocommutative* if it commutes *up to gyrations*: ie for every $a, b \in G$, $a \oplus b = \operatorname{gyr}(a,b)(b \oplus a)$.

It is within this restricted class of gyro-commutative gyrogroups that a satisfactory analog of familiar vector space operations can be constructed Ungar (2018).

A gyrovector space is a gyro-commutative gyrogrorup $(G, \oplus)$ together with a scalar multiplication $\otimes \colon \mathbb{R} \times G \to G$ such that 1 acts as the identity, and its interaction with standard multiplication, gyro-addition and gyration are constrained by

$$\begin{aligned} r_1 \otimes (r_2 \otimes a) &= r_1 r_2 \otimes a \\ (r_1 + r_2) \oplus a &= (r_1 \otimes a) \oplus (r_2 \otimes a) \\ r \otimes \mathrm{gyr}(a,b)c &= \mathrm{gyr}(a,b)(r \otimes c) \\ \mathrm{gyr}(r_1 \otimes a, r_2 \otimes a) &= I \end{aligned} \tag{A.2}$$

Typically a gyrovector space is also assumed to be constructed within an ambient real inner product space, and there are additional compatibility relations between the operations of $(G, \oplus, \otimes)$ and the ambient vector space addition $(+)$ and norm $\|v\| = \sqrt{v \cdot v}$.

Gyro-vector spaces generalize vector spaces much as gyro-groups generalized groups: every vector space is a gyro-vector space with trivial gyration. As such, the formalism of gyro-vector spaces provides a convenient generalization where one may attempt to replace $+, -, \times$ in formulas familiar from Euclidean spaces with $\oplus, \ominus, \otimes$; being careful to recall that gyro-addition is neither commutative nor associative, and gyro-multiplication rarely distributes over $\oplus$.

Solving Equations in Gyrogroups

As an example of the difficulties posed by this, if one requires the solution to the Euclidean equation $a + x = b$, it is equally correct to write $x = b - a$ or $x = -a + b$. But the translations $x = b \ominus a$ and $x = \ominus a \oplus b$ into a gyrogroup G need not be equal, and generically only the latter solves the gyrovector equation $a \oplus x = b$.

To make this more systematic, note that working inwards respecting the order of operations, we are able to solve any equation in a gyrogroup if we compute a *left cancellation law*, *right cancellation law* and *invert scalar multiplication*.

Proposition A.3 (Left-Cancellation)**.** Let a, b be elements of a gyrogroup G. Then the relation $a \oplus x = b$ is satisfied by the unique value $x = (\ominus a) \oplus b$.

Proof. Substituting the claimed expression for x, we verify by direct computation from the axioms of a gyrogroup, and the basic properties of Propositions A.1.

$$\begin{aligned} a \oplus x &= a \oplus ((\ominus a) \oplus b) \\ &= (a \oplus \ominus a) \oplus \operatorname{gyr}(a, \ominus a)b \\ &= 0 \oplus \operatorname{gyr}(a, \ominus a)b \\ &= id_G(b) \\ &= b \end{aligned}$$

□

Proposition A.4 (Right-Cancellation)**.** Let a, b be elements of a gyrogroup G. Then the relation $x \oplus a = b$ is satisfied by the unique value $x = b \boxminus a = a \ominus \operatorname{gyr}(a, \ominus b)b$, where $\boxminus$ is the additive inverse of the gyrogroup co-operation from Section A.

Proof. To begin, we put the proposed solution $b \boxminus a$ in a more usable form:

$$\begin{aligned} b \boxminus a &= b \boxplus \ominus a \\ &= b \oplus \operatorname{gyr}(b, \ominus \ominus a) \ominus a \\ &= b \ominus \operatorname{gyr}(b, a)a \end{aligned}$$

We now verify the claim by subsituting the given value of x, and using the properties described in Propositions A.1 and A.2, (in particular, in the third step we expand a using nested gyration)

$$\begin{aligned} x \oplus a &= (b \boxminus a) \oplus a \\ &= (b \ominus \operatorname{gyr}(b, a)a) \oplus \operatorname{id}_G(a) \\ &= (b \ominus \operatorname{gyr}(b, a)a) \oplus (\operatorname{gyr}(b, \ominus \operatorname{gyr}(b, a)a) \operatorname{gyr}(b, a)a) \\ &= b \oplus (\ominus \operatorname{gyr}(b, a)a \oplus \operatorname{gyr}(b, a)a) \\ &= b \oplus 0 \\ &= b \end{aligned}$$

□

Proposition A.5 (Inverting Scalar Multiplication)**.** Let $r \in \mathbb{R}$ be any scalar, and a an element of a gyrogroup G. Then the relation $r \otimes x = a$ is satisfied by the unique element $x = \left(\frac{1}{r}\right) \otimes a$.

Proof. Substituting x immediately yeidls the result given the axioms of gyro-scalar multiplication:

$$\begin{aligned} r\otimes &= r \otimes \left(\frac{1}{r} \otimes a\right) \\ &= \left(r \times \frac{1}{r}\right) \otimes a \\ &= 1 \otimes a \\ &= a \end{aligned}$$

□

These three cancellation laws allow one work correctly with the gyro-translations of Euclidean vector space statements. Take for example the vector space expression $a + rx + b = c$ for vectors a, b, c, x and scalar r. One possible gyro-vector space translation of this is $(a \oplus (r \otimes x)) \oplus b = c$ — and given this translation, we may work fully within the gyrovector space to solve for x as follows:

$$\begin{aligned} (a \oplus (r \otimes x)) \oplus b &= c \\ a \oplus (r \otimes x) &= c \boxminus b \\ r \otimes x &= (\ominus a) \oplus (c \boxminus b) \\ x &= \frac{1}{r} \otimes ((\ominus a) \oplus (c \boxminus b)) \end{aligned}$$

Appendix B

Symmetric Spaces

Vector-Valued Distance for Symmetric Spaces

The familiar geometric invariant of pairs of points is simply the distance between them. For rank n symmetric spaces, this one dimensional invariant is superseded by an n-dimensional invariant: the *vector-valued distance*.

Abstractly, one computes this invariant as follows: for a symmetric space M with $\mathrm{Isom}_0(M) = G$, choose a distinguished basepoint $m \in M$, and let $K < G$ be the subgroup of symmetries fixing m. Additionally choose a distinguished maximal flat $F \subset M$ containing m, and an identification of this flat with $\mathbb{R}^n$. Given any pair of points $p, q \in M$, one may find an isometry $g \in G$ moving p to m, and q to some other point $g(q) = v \in F$ in the distinguished flat. Under the identification of F with $\mathbb{R}^n$, the difference vector $v - m$ is a vector-valued invariant of the original two points, and determines the vector-valued distance. In practice we may arrange so that m is identified with $\mathbf{0}$, so this difference is simply v.

In rank 1, the flat F identifies with $\mathbb{R}^1$, and this difference vector $v - m$ with a number. This number encodes all geometric information about the pair (p, q) invariant under the symmetries of M. Indeed, the distance from p to q is simply its absolute value!

In rank n, "taking the absolute value" has an n-dimensional generalization, via a finite a finite group of symmetries of called the Weyl group. This group $W < K$ acts on the flat F, and abstractly, *the vector-valued distance* $\mathrm{vDist(p,q)}$ *from* p *to* q *is this difference vector up to the action of the Weyl group*. This vector-valued distance $\mathrm{vDist(p,q)}$ is the complete invariant for pairs of points in M - it contains all geometric information about the pair which is invariant under all symmetries. In particular, given the vector-valued distance $\mathrm{vDist(p,q)}$, the (Riemannian) distance from p to q is trivial to compute - it is given by the length of $\mathrm{vDist(p,q)}$ in $\mathbb{R}^n$.

The identification of F with $\mathbb{R}^n$ makes this more explicit. Here the Weyl group acts as a group of linear transformations, which divide $\mathbb{R}^n$ into a collection of conical fundamental

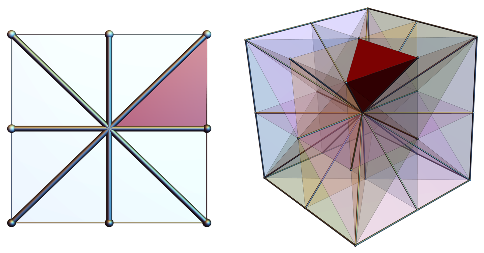


Figure B.1: A choice of Weyl chamber the Siegel spaces of rank n is given by $C = \{(v_i) \in \mathbb{R}^n \mid v_1 \geq v_2 \geq \cdots \geq v_n \geq 0\}$. In rank 1, this is the nonnegative reals. Illustrated here are ranks $n = 2, 3$.

domains for the action, known as Weyl chambers. Choosing a fixed Weyl chamber C, we may use these symmetries to move our originally found difference vector $v - m$ into C. The vector-valued distance is this resulting vector $\mathrm{vDist(p,q) \in C}$.

For example, in rank n Siegel space, the Weyl group acts on $\mathbb{R}^2$ by the reflection symmetries of a cube, and a choice of Weyl chamber amounts to a choice of linear ordering of the vector components with respect to zero. One choice is shown in Figure B.1. In rank 2, this chamber is used to display the vector-valued distances associated to edges and nodes of an embedded graph in §8.4.5. Note that once a Weyl chamber is picked it may be possible to find the vector-valued distance corresponding to a vector in $\mathbb{R}^n$ without explicit use of the Weyl group: for the Siegel spaces this is by sorting the vector components in nondecreasing order.

Computing Distance: The process for computing the vector-valued distance is summarized below. It is explicitly carried out for the Siegel spaces and their compact duals in Appendix C.

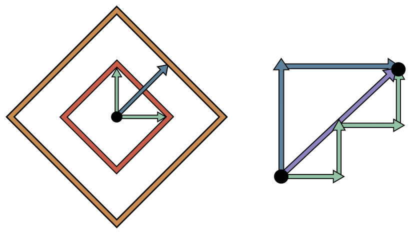
Let M, G, K, F, m be as in the previous section. Choose an identification $\phi\colon F \to \mathbb{R}^n$ which sends the basepoint m to $\mathbf{0}$, and a Weyl chamber $C \subset \mathbb{R}^n$ for the Weyl group W. For any pair of points $p, q \in M$;

1. **Move p to the basepoint:**
   Compute $g \in G$ such that $g(p) = m$.

2. **Move q into the flat:**
   Compute $k \in K$ such that $k(g(q)) \in F$. Now both $g(p) = m$ and $k(g(q))$ lie in the distinguished flat F.

3. **Identify the flat with $\mathbb{R}^n$:**
   Compute $u = \phi(k(g(q))) \in \mathbb{R}^n$. The points $\mathbf{0}$ and u represent p, q after being moved into the flat, respectively.

4. **Return the Vector-Valued Distance:**


(a) Left: Vectors of length 1 and 2 with respect to the ℓ^1 norm on $\mathbb{R}^2$. Right: three geodesics of length 4 in the ℓ^1 metric (to same scale as left image).

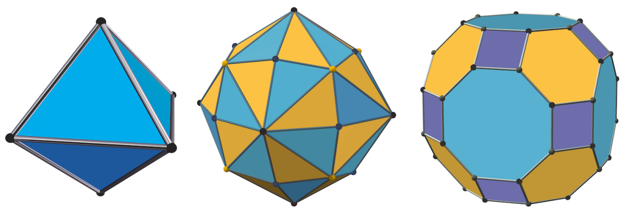

(b) The unit spheres of several Finsler metrics on $\mathbb{R}^3$ invariant under the Weyl group of the rank 3 Siegel space. The octahedron induces the ℓ^1 metric.

Figure B.2: Diagrams of Finsler Metrics.

Compute $v \in C$ such that $v = Au$ for some element $A \in W$. This is the vector-valued distance $\mathrm{vDist}(\mathrm{p}, \mathrm{q})$

The **Riemannian distance** is computed directly from the vector-valued distance as its Euclidean norm, $\mathrm{dist}(p, q) = \|\mathrm{vDist}(p, q)\|$.

Finsler Metrics for Symmetric Spaces

In this section we provide additional notions to the introduction on Finsler metrics from §2.5.3.

Finsler Metrics on $\mathbb{R}^n$: Any norm on $\mathbb{R}^n$ defines a Finsler metric. As norms on a vector space are uniquely determined by their unit spheres, the data of a Finsler metric is given by a convex polytope S containing $\mathbf{0}$. An important example in this work is the ℓ^1 Finsler metric on $\mathbb{R}^n$, given by the norm $\|(x_i)\|_{\ell^1} = \sum_i |x_i|$. Its unit sphere is the boundary of the dual to the n-dimensional cube (in $\mathbb{R}^2$, this is again a square, but oriented at $45°$ with respect to the coordinate axes).

Given such an P, the Finsler norm $\|v\|_F$ of a vector $v \in \mathbb{R}^n$ is the unique positive ℓ such that $\frac{1}{\ell}v \in \partial P$. Figure B.2a shows the spheres of radius 1 and 2 with respect to the ℓ^1 metric on the plane.

Finsler Metrics on Symmetric Spaces: To define a Finsler metric on a symmetric space M, it suffices to define it on a chosen maximal flat, and evaluate on arbitrary pairs of points with the help of the vector-valued distance. To induce a well defined Finsler metric M, a norm on this designated flat need only be invariant under the Weyl group W. Said geometrically, the unit sphere of the norm $\|\cdot\|_F$ needs to contain it as a subgroup of its symmetries. Given such a norm, the Finsler distance between two points is simply the

Finsler norm of their vector-valued distance

$$d_F(p,q) = \|\mathrm{vDist}(p,q)\|_F.$$

Consequentially once the vector-valued distance is known, any selection of Riemannian or Finsler distances may be computed at marginal additional cost.

Local Geometry for Riemannian Optimization

Different Riemannian optimization methods require various input from the local geometry. In this section we describe a computation of the Riemannian gradient, parallel transport and the exponential map for general irreducible symmetric spaces.

Riemannian Gradient

Given a function $f \colon M \to \mathbb{R}$, the *differential* of f is a 1-form which measures how infinitesimal changes in the input affects (infinitesimally) the output. More precisely at each point $p \in M$, df is a linear functional on T_pM sending a vector v to the directional derivative $df_p(v)$ of f in direction v.

In Euclidean space, this data is conveniently expressed as a vector: *the gradient* ∇f defined such that $(\nabla f(p)) \cdot v = df_p(v)$. This extends directly to any Riemannian manifold, where the dot product is replaced with the Riemannian metric. That is, the *Riemannian gradient* of a function $f \colon M \to \mathbb{R}$ is the vector field $grad_R(f)$ on M such that

$$g_p(\mathrm{grad}_R(f), v) = df_p(v)$$

for every $p \in M$, $v \in T_pM$. Given a particular model (and thus, a particular coordinate system and metric tensor) one may use this implicit definition to give a formula for grad_R. See Appendix C for an explicit example, deriving the Riemannian gradient for Siegel space from its metric tensor.

Parallel Transport

While the lack of curvature in Euclidean space allows all tangent spaces to be identified, in general symmetric spaces the result of transporting a vector from one tangent space to another is a nontrivial, path dependent operation. This *parallel transport* assigns to a path γ in M from p to q an isomorphism $P_\gamma \colon T_pM \to T_qM$ interpreted as taking a vector $v \in T_pM$ at p to $P_\gamma(v) \in T_qM$ by “moving without turning" along γ.

The computation of parallel transport along geodesics in a symmetric space is possible directly from the isometry group. To fix notation, for each $m \in M$ let $\sigma_m \in G$ be the geodesic reflection fixing m. Let γ be a geodesic in M through p at $t = 0$. As t varies, the isometries $\tau_t = \sigma_{\gamma(t/2)} \circ \sigma_p$, called *transvections*, form the 1-parameter subgroup of translations along γ. If $p, q \in M$ are two points at distance L apart along the the geodesic γ, the transvection τ_L takes p to q, and its derivative $(d\tau_L)_p = P_\gamma \colon T_pM \to T_qM$ performs the parallel transport for γ.

The Exponential Map & Lie Algebra

The exponential map for a Riemannian manifold M is the map $\exp\colon TM \to M$ such that if $v \in T_p(X)$, $\exp(v)$ is the point in M reached by traveling distance $\|v\|$ along the geodesic on M through p with initial direction parallel to v.

When M is a symmetric space with symmetry group G, this may be computed using the Lie group exponential $\exp\colon \mathfrak{g} \to G$ (the matrix exponential, when G is a matrix Lie group). Choose a point $p \in M$ and let σ_p be the geodesic reflection in p. Then σ_p defines an involution $G \to G$ by $g \mapsto \sigma_p \circ g \circ \sigma_p$ (where composition is as isometries of M), and the eigenspaces of the differential of this involtuion give a decomposition $\mathfrak{g} = \mathfrak{k} \oplus \mathfrak{p}$ into the $+1$ eigenspace $\mathfrak{k}$ and the -1 eigenspace $\mathfrak{p}$. Here $\mathfrak{k}$ is the Lie algebra of the stabilizer $K = \mathfrak{stab}(p) < G$, and so $\mathfrak{p}$ identifies with T_pM under the differential of the quotient $G \to G/K \cong M$.

Let $\phi\colon T_pM \to \mathfrak{p}$ be the inverse of this identification. Then for a vector $v \in T_pM$, we may find the point $q = \exp_p(v) \in M$ as follows:

1. Compute $\phi(v) = A \in \mathfrak{p}$. This is the tangent vector v, viewed as a matrix in the Lie algebra to G.
2. Compute $g = \exp(A)$, where $\exp$ is the matrix exponential.
3. Use the action of G on M by isometries to compute $q = g(p)$.

Appendix C

Siegel Spaces

This appendix gives the mathematic calculations required to implement two models of Siegel space (the bounded domain model and upper half space) as well as a model of its compact dual.

Linear Algebra Conventions

A few clarifications from linear algebra can be useful:

1. The inverse of a matrix X^{-1}, the product of two matrices XY, the square X^2 of a square matrix are understood with respect to the matrix operations. Unless all matrices are diagonal these are different than doing the same operation to each entry of the matrix.

2. If $Z = X + iY$ is a complex matrix,

   - Z^t denotes the transpose matrix, i.e. $(Z^t)_{ij} = Z_{ji}$,
   - $\overline{Z} = X - iY$ denotes the complex conjugate
   - X^* denotes its transpose conjugate, i.e. $X^* = \overline{X^t}$.

3. A complex square matrix Z is *Hermitian* if $Z^* = Z$. In this case its eigenvalues are real and positive. It is *unitary* if $Z^* = Z^{-1}$. In this case its eigenvalues are complex numbers of absolute value 1 (i.e. points in the unit circle).

4. If X is a real symmetric, or complex Hermitian matrix, $X \gg 0$ means that X is positive definite, equivalently all its eigenvalues are bigger than zero.

Takagi Factorization

Given a complex symmetric matrix A, the Takagi factorization (Takagi, 1924) is an algorithm that computes a real diagonal matrix D and a complex unitary matrix K such that

$$A = \overline{K} D K^*.$$

This will be useful to work with the bounded domain model. It is done in a few steps

1. Find Z_1 unitary, D real diagonal such that

$$A^* A = Z_1^* D^2 Z_1$$

2. Find Z_2 orthogonal, B complex diagonal such that

$$\overline{Z}_1 A Z_1^* = Z_2 B Z_2^t$$

   This is possible since the real and imaginary parts of $\overline{Z}_1 A Z_1^*$ are symmetric and commute, and are therefore diagonalizable in the same orthogonal basis.

3. Set Z_3 be the diagonal matrix with entries

$$(Z_3)_{ii} = \left(\sqrt{\frac{b_i}{|b_i|}}\right)^{-1}$$

   where $b_i = (B)_{ii}$

4. Set $K = Z_1^* Z_2 Z_3$, D as in Step 1. It then holds

$$A = \overline{K} D K^*.$$

Siegel Space and its Models

We consider two models for the symmetric space, the bounded domain

$$\mathcal{B}_n := \{Z \in \mathrm{Sym}(n, \mathbb{C}) |\ \mathrm{Id} - Z^* Z \gg 0\}$$

and the upper half space

$$\mathcal{S}_n := \{X + iY \in \mathrm{Sym}(n, \mathbb{C}) |\ Y \gg 0\}.$$

An explicit isomorphism between the two domains is given by the Cayley transform

$$\begin{array}{rccc} c: & \mathcal{B}_n & \to & \mathcal{S}_n \\ & Z & \mapsto & i(Z+\mathrm{Id})(Z-\mathrm{Id})^{-1} \end{array}$$

whose inverse $c^{-1} = t$ is given by

$$\begin{array}{rccc} t: & \mathcal{S}_n & \to & \mathcal{B}_n \\ & X & \mapsto & (X-i\mathrm{Id})(X+i\mathrm{Id})^{-1} \end{array}$$

When needed, a choice of **basepoint** for these models is $i\mathrm{Id} \in \mathcal{S}_n$ for upper half space and the zero matrix $\mathbf{0} \in \mathcal{B}_n$ for the bounded domain. A convenient choice of **maximal flats** containing these basepoints are the subspaces $\{iD \mid D = \mathrm{diag}(d_i),\ d_i > 0\} \subset \mathcal{S}_n$ and $\{D = \mathrm{diag}(d_i) \mid d_i \in (-1,1)\} \subset \mathcal{B}_n$.

The group of symmetries of the Siegel space $\mathcal{S}_n$ is $\mathrm{Sp}(2n,\mathbb{R})$, the subgroup of $\mathrm{SL}(2n,\mathbb{R})$ preserving a symplectic form: a non-degenerate antisymmetric bilinear form on $\mathbb{R}^{2n}$. In this text we will choose the symplectic form represented, with respect to the standard basis, by the matrix $\left(\begin{smallmatrix} 0 & \mathrm{Id}_n \\ -\mathrm{Id}_n & 0 \end{smallmatrix}\right)$ so that the symplectic group is given by the matrices that have the block expression

$$\left\{ \begin{pmatrix} A & B \\ C & D \end{pmatrix} \;\middle|\; \begin{array}{l} A^tD - C^tB = \mathrm{Id} \\ A^tC = C^tA \\ B^tD = D^tB \end{array} \right\}$$

where A, B, C, D are real $n \times n$ matrices.

The symplectic group $\mathrm{Sp}(2n,\mathbb{R})$ acts on $\mathcal{S}_n$ by non-commutative fractional linear transformations

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot Z = (AZ+B)(CZ+D)^{-1}.$$

The action of $\mathrm{Sp}(2n,\mathbb{R})$ on $\mathcal{B}_n$ can be obtained through the Cayley transform.

Computing the Vector-Valued Distance

The Riemannian metric, as well as any desired Finsler distance, are computable directly from the vector-valued distance as explained in Appendix B. Following those steps, we give an explicit implementation for the upper half space model below, and subsequently use the Cayley transform to extend this to the bounded domain model.

Given as input two points $Z_1, Z_2 \in \mathcal{S}_n$ we perform the following computations:

1) Move Z_1 to the basepoint: Compute the image of Z_2 under the transformation taking

Z_1 to iI, defining

$$Z_3 := \sqrt{\Im Z_1}^{-1}(Z_2 - \Re Z_1)\sqrt{\Im Z_1}^{-1} \in \mathcal{S}_n$$

2) Move Z_2 into the chosen flat: Define

$$W = t(Z_3) \in \mathcal{B},$$

and use the Takagi factorization to write

$$W = \overline{K} D K^*$$

for some real diagonal matrix D with eigenvalues between 0 and 1, and some unitary matrix K. *Note: to make computations easier, we are leveraging the geometry of both models here, so in fact $i(I+D)(I-D)^{-1}$ is the matrix lying in the standard flat containing iI.*

3) Identify the flat with $\mathbb{R}^n$: Define the vector $v = (v_i) \in \mathbb{R}^n$ with

$$v_i = \log \frac{1+d_i}{1-d_i},$$

for d_i the i^{th} diagonal entry of the matrix D from the last step.

4) Return the Vector-Valued Distance: Sort the absolute values of the entries of v to be in nonincreasing order, and set $\mathrm{vDist}(Z_1, Z_2)$ equal to the resulting list.

$$\mathrm{vDist} = (|v_{i_1}|, |v_{i_2}|, \ldots, |v_{i_n}|)$$

$$|v_{i_1}| \geq |v_{i_2}| \geq \cdots \geq |v_{i_n}|$$

Bounded domain: In this case, given $W_1, W_2 \in \mathcal{B}$ we consider the pair $Z_1, Z_2 \in \mathcal{S}_n$ obtained applying the Cayley transform $Z_i = t(W_i)$. Then we can apply the previous algorithm, indeed

$$\mathrm{vDist}(W_1, W_2) = \mathrm{vDist}(Z_1, Z_2).$$

Riemannian & Finsler Distances

The Riemannian distance between two points X, Y in the Siegel space (either the upper half space or bounded domain model) is induced by the Euclidean metric on its maximal flats. This is calculable directly from the vector-valued distance $\mathrm{vDist}(X,Y) = (v_1, v_2, \ldots, v_n)$ as

$$d^R(X,Y) = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

The Weyl group for the rank n Siegel space is the symmetry group of the n cube. Thus, any Finsler metric on $\mathbb{R}^n$ whose unit sphere has these symmetries has these symmetries induces a Finsler metric on Siegel space. The class of such finsler metrics includes many well-known examples such as the ℓ^p metrics

$$\|(v_1, \dots, v_n)\|_{\ell^p} = \left(\sum_i |v_i|^p \right)^{\frac{1}{p}},$$

which is one of the reasons the Siegel space is an attractive avenue for experimentation.

Of particular interest are the ℓ^1 and ℓ^∞ Finsler metrics. The distance functions induced on the Siegel space by them are given below

$$d^{F_1}(X, Y) = \sum_{i=1}^{n} v_i \qquad d^{F_\infty}(X, Y) = v_1.$$

Where X, Y are points in Siegel space (again, either in the upper half space or bounded domain models), and the v_i are the component of the vector-valued distance $\mathrm{vDist}(X, Y) = (v_1, v_2, \dots, v_n)$.

There are explicit bounds between the distances, for example

$$\frac{1}{\sqrt{n}} d^{F_1}(X, Y) \leq d^{R}(X, Y) \leq d^{F_1}(X, Y) \tag{C.1}$$

Furthermore, we have

$$\begin{aligned} d^{F_1}(X, Y) = \log\det(\sqrt{R(X, Y)} + \mathrm{Id}) - \\ \log\det(\mathrm{Id} - \sqrt{R(X, Y)}) \end{aligned} \tag{C.2}$$

which, in turn, allows to estimate the Riemannian distance using (C.1).

Distance Algorithm Complexity

In this section we discuss the computational theoretical complexity of the different operations involved in the development of this work. We employ Big O notation[1]. Since in all cases operations are not nested, but are applied sequentially, the costs can be added resulting in a polynomial expression. Thus, by applying the properties of the notation, we disregard lower-order terms of the polynomial.

[1] https://en.wikipedia.org/wiki/Big_O_notation

Real Matrix Operations: For $n \times n$ matrices with real entries, the associated complexity of each operation is as follows:[2]

- Addition and subtraction: $\mathcal{O}(n^2)$
- Multiplication: $\mathcal{O}(n^{2.4})$
- Inversion: $\mathcal{O}(n^{2.4})$
- Diagonalization: $\mathcal{O}(n^3)$

Complex Matrix Operations: A complex symmetric matrix $Z \in \operatorname{Sym}(n, \mathbb{C})$ can be written as $Z = X + iY$, where $X = \Re(Z), Y = \Im(Z) \in \operatorname{Sym}(n, \mathbb{R})$ are symmetric matrices with real entries. We implement the elemental operations for these matrices with the following associated costs:

- Multiplication: $\mathcal{O}(n^{2.4})$. It involves 4 real matrix multiplications, plus additions and subtractions.
- Square root: $\mathcal{O}(n^3)$. It involves a diagonalization and 2 matrix multiplications.[3]
- Inverse: $\mathcal{O}(n^{2.4})$. It involves real matrix inversions and multiplications (Falkenberg, 2007).

Takagi Factorization: This factorization involves complex and real multiplications ($\mathcal{O}(n^{2.4})$), and diagonalizations ($\mathcal{O}(n^3)$). It also involves the diagonalization of a $2n \times 2n$ matrix, which implies:

$$\mathcal{O}((2n)^3) = \mathcal{O}(8n^3) \simeq \mathcal{O}(n^3) \tag{C.3}$$

Therefore, the final boundary for its cost is $\mathcal{O}(n^3)$.

Cayley Transform: This operation along with its inverse are composed of matrix inversions and multiplications, thus the cost is $\mathcal{O}(n^{2.4})$.

Distance Algorithm: The full computation of the distance algorithm in the upperhalf space involves matrix square root, multiplications, inverses, and the application of the Cayley transform and the Takagi factorization. Since they are applied sequentially, without affecting the dimensionality of the matrices, we can take the highest value as the asymptotic cost of the algorithm, which is $\mathcal{O}(n^3)$.

[2]`https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations`

[3]`https://en.wikipedia.org/wiki/Square_root_of_a_matrix`

For the bounded domain, the matrices are mapped into the upperhalf space by an additional application of the inverse Cayley transform, and then the same distance algorithm is applied. Therefore, in this space the complexity also converges to $\mathcal{O}(n^3)$.

Riemannian Gradient

We consider on $\mathrm{Sym}(n, \mathbb{C})$ the Euclidean metric given by

$$\|V\|_E^2 = \mathrm{tr}(V\overline{V}),$$

here tr denotes the trace, and, as above, $V\overline{V}$ denotes the matrix product of the matrix V and its conjugate.

Siegel upperhalf space: The Riemannian metric at a point $Z \in \mathcal{S}_n$, where $Z = X + iY$ is given by Siegel (1943)

$$\|V\|_R^2 = \mathrm{tr}(Y^{-1}VY^{-1}\overline{V}).$$

As a result we deduce that

$$\mathrm{grad}(f(Z)) = Y \cdot \mathrm{grad_E}(f(Z)) \cdot Y$$

Bounded domain: In this case we have

$$\mathrm{grad}(f(Z)) = A \cdot \mathrm{grad_E}(f(Z)) \cdot A$$

where $A = \mathrm{Id} - \overline{Z}Z$

Embedding Initialization

Different embeddings methods initialize the points close to a fixed basepoint. In this manner, no a priori bias is introduced in the model, since all the embeddings start with similar values.

We use the **basepoints** specified previously: $i\mathrm{Id}$ for Siegel upper half space and $\mathbf{0}$ for the bounded domain model.

In order to produce a random point we generate a random symmetric matrix with small entries and add it to our basepoint. As soon as all entries of the perturbation are smaller than $1/n$ the resulting matrix necessarily belongs to the model. In our experiments, we generate random symmetric matrices with entries taken from a uniform distribution $\mathcal{U}(-0.001, 0.001)$.

Projecting Back to the Models

The goal of this section is to explain two algorithms that, given ϵ and a point $Z \in \mathrm{Sym}(n, \mathbb{C})$, return a point $Z_\epsilon^{\mathcal{S}}$ (resp. $Z_\epsilon^{\mathcal{B}}$), a point close to the original point lying in the ϵ-interior of the model. This is the equivalent of the projection applied in Nickel & Kiela (2017) to constrain the embeddings to remain within the Poincaré ball, but adapted to the structure of the model. Observe that the projections are not conjugated through the Cayley transform.

Siegel upperhalf space: In the case of the Siegel upperhalf space $\mathcal{S}_n$ given a point $Z = X + iY \in \mathrm{Sym}(n, \mathbb{C})$

1. Find a real n-dimensional diagonal matrix D and an orthogonal matrix K such that
$$Y = K^t D K$$
2. Compute the diagonal matrix D_ϵ with the property that
$$(D_\epsilon)_{ii} = \begin{cases} D_{ii} & \text{if } D_{ii} > \epsilon \\ \epsilon & \text{otherwise} \end{cases}$$
3. The projection is given by
$$Z_\epsilon^{\mathcal{S}} := X + iK^t D_\epsilon K$$

Bounded Domain: In the case of the bounded domain $\mathcal{B}$ given a point $Z = X + iY \in \mathrm{Sym}(n, \mathbb{C})$

1. Use the Takagi factorization to find a real n-dimensional diagonal matrix D and an unitary matrix K such that
$$Y = \overline{K} D K^*$$
2. Compute the diagonal matrix $D_\epsilon^{\mathcal{B}}$ with the property that
$$(D_\epsilon^{\mathcal{B}})_{ii} = \begin{cases} D_{ii} & \text{if } D_{ii} < 1 - \epsilon \\ 1 - \epsilon & \text{otherwise} \end{cases}$$
3. The projection is given by
$$Z_\epsilon^{\mathcal{B}} := \overline{K} D_\epsilon^{\mathcal{B}} K^*$$

Crossratio and Distance

Given two points X, Y in Siegel space, there is an alternative means of calculating the vector-valued distance (and thus any Riemannian or Finsler distance one wishes) via an invariant known as the cross ratio.

Siegel upperhalf space: Given two points $X, Y \in \mathcal{S}_n$ their crossratio is given by the complex $n \times n$-matrix

$$R_{\mathcal{S}}(X,Y) = (X-Y)(X-\overline{Y})^{-1}(\overline{X}-\overline{Y})(\overline{X}-Y)^{-1}.$$

It was established by Siegel Siegel (1943) that if $r_1, \dots, r_n$ denote the eigenvalues of R (which are necessarily real greater than or equal to 1) and we denote by v_i the numbers

$$v_i = \log \frac{1+\sqrt{r}_i}{1-\sqrt{r}_i}$$

then the v_i are the components of the vector-valued distandce $\mathrm{vDist}(X,Y)$. Thus, the Riemannian distance is

$$d^R(X,Y) = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

The Finsler distances d^{F_1} and d^{F_∞} are likewise given by the same formulas as previously.

In general it is computationally difficult to compute the eigenvalues, or the squareroot, of a general complex matrix. However, we can use the determinant $\det_R$ of the matrix $R(X,Y)$ to give a lower bound on the distance:

$$\log \frac{1+\sqrt{\det_R}}{1-\sqrt{\det_R}} \leq d^R(X,Y).$$

Bounded domain: The same study applies to pairs of points $X, Y \in \mathcal{B}$, but their crossratio should be replaced by the expression

$$\begin{aligned} R_{\mathcal{B}}(X,Y) = (X-Y)(X-\overline{Y^{-1}})^{-1} \\ (\overline{X^{-1}}-\overline{Y^{-1}})(\overline{X^{-1}}-Y)^{-1} \end{aligned} \tag{C.4}$$

The Compact Dual of the Siegel Space

The compact dual to the (non-positively) curved Siegel space is a compact non-negatively curved symmetric space; in rank 1 this is just the 2-sphere. Many computations in the compact dual are analogous to those for the Siegel spaces, and are presented below.

Model

Abstractly, the compact dual is the space of complex structures on quaternionic n-dimensional space compatible with a fixed inner product. It is convenient to work with a coordinate chart, or *affine path* covering all but a measure zero subset of this space. We denote this patch by $\mathcal{D}_n$, which consists of all $n \times n$ complex symmetric matrices:

$$\mathcal{D}_n = \mathrm{Sym}(n; \mathbb{C})$$

With this choice of model, tangent vectors to $\mathcal{D}_n$ are also represented by complex symmetric matrices. More precisely, for each $W \in \mathcal{D}_n$ we may identify the tangent space $T_W\mathcal{D}_n$ with $\mathrm{Sym}(n, \mathbb{C})$.

Basepoint: The basepoint of $\mathcal{D}_n$ is the zero matrix $\mathbf{0}$.

Maximal Flat: A useful choice of maximal flat is the subspace of real diagonal matrices.

Projection: The model $\mathcal{D}_n$ is a linear subspace of the space of $n \times n$ complex matrices. Orthogonal projection onto this subspace is given by symmetrization,

$$W \mapsto \frac{1}{2}(W + W^t).$$

Isometries: The symmetries of the compact dual are given by the compact symplectic group $\mathrm{Sp}(n)$. With respect to the model $\mathcal{D}_n$, we may realize this as the intersection of the complex symplectic group $\mathrm{Sp}(2n, \mathbb{C})$ and the unitary group $U(2n, \mathbb{C})$

$$\left\{ \begin{pmatrix} A & B \\ C & D \end{pmatrix} \;\middle|\; \begin{array}{l} A^tD - C^tB = \mathrm{Id} \\ A^tC = C^tA \\ B^tD = D^tB \\ A^*A + C^*C = \mathrm{Id} \\ B^*B + D^*D = \mathrm{Id} \\ A^*B + C^*D = 0 \end{array} \right\}$$

where A, B, C, D are *complex* $n \times n$ matrices. The first four conditions are analogs of those defining $\mathrm{Sp}(2n; \mathbb{R})$, and the final three come from the defining property that a unitary matrix M satisfies $M^*M = \mathrm{Id}$.

This group acts on $\mathcal{D}_n$ by non-commutative fractional linear transformations

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot W = (AW + B)(CW + D)^{-1}.$$

Riemannian Metric & Gradient: The Riemannian metric at a point $W \in \mathcal{D}_n$ is given by

$$\langle U, V \rangle_W = (\mathrm{Id} + \overline{W}W)^{-1}U(\mathrm{Id} + \overline{W}W)^{-1}\overline{V},$$

where U, V are tangent vectors at W.

The gradient of a function on the compact dual can be written in terms of its Euclidean gradient, via a formula very similar to that for the Bounded Domain model of the Siegel space. In this case we have

$$\mathrm{grad}(f(W)) = A \cdot \mathrm{grad}_{\mathrm{E}}(f(W)) \cdot A$$

where (the only difference from the bounded domain version being that the $-$ sign in the definition of A has been replaced with a $+$).

Vector-Valued Distance

We again give an explicit implementation of the abstract procedure described in Appendix B, to calculate the vector-valued distance associated to an arbitrary pair $W_1, W_2 \in \mathcal{D}_n$ as follows:

Move $W1$ to the basepoint:

1. Use the Takagi factorization to write

   $$W_1 = \overline{U} P U^*$$

   for a unitary matrix U and real diagonal matrix P.

2. From P, we build the diagonal matrix $A = (\mathrm{Id} + P^2)^{-1/2}$. That is, the diagonal entries of A are $a_i = \frac{1}{\sqrt{1+p_i^2}}$ for p_i the diagonal entries of P.

3. From A, U we build the following elements $M, R \in \mathrm{Sp}(n)$ of the compact symplectic group:

   $$M = \begin{pmatrix} A & -AP \\ AP & A \end{pmatrix} \qquad R = \begin{pmatrix} U^t & 0 \\ 0 & U^* \end{pmatrix}$$

We now use the transformation $M \cdot R$ to move the pair (W_1, W_2) to a pair $(\mathbf{0}, Z)$. Because W_1 ends at the basepoint by construction, we focus on W_2.

4. Compute $X = R.W_2$, that is $X = U^t W_2 U$.

5. Compute $Z = M.X$, that is $Z = (AX - AP)(APX - A)^{-1}$. Alternatively, this simplifies to the conjugation $Z = AYA^{-1}$ by A of the matrix $Y = (X - P)(PX - \mathrm{Id})^{-1}$

Move Z into the chosen flat: Use the Takagi factorization to write

$$Z = \overline{K} D K^*$$

for a unitary matrix K and real diagonal matrix D.

Identify the Flat with $\mathbb{R}^n$: Produce from D the n-vector

$$v = (\arctan(d_1), \dots \arctan(d_n))$$

Where $(d_1, \dots d_n)$ are the diagonal entries of D.

Return the Vector-Valued Distance: Order the the entries of v in nondecreasing order. This is the vector-valued distance.

$$\mathrm{vDist} = (v_{i_1}, v_{i_2}, \dots, v_{i_n})$$

$$v_{i_1} \geq v_{i_2} \geq \dots \geq v_{i_n} \geq 0$$

Riemannian and Finsler Distances:

The Riemannian distance between two points X, Y in the compact dual is calculable directly from the vector-valued distance $\mathrm{vDist}(X, Y) = (v_1, v_2, \dots, v_n)$ as

$$d^R(X, Y) = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

The Weyl group for the compact dual is the same as for Seigel space, the symmetries of the n-cube. Thus the same collection of Finsler metrics induce distance functions on the compact dual, and their formulas in terms of the vector-valued distance are unchanged.

$$d^{F_1}(X, Y) = \sum_{i=1}^{n} v_i \qquad d^{F_\infty}(X, Y) = v_1.$$

Interpolation between Siegel Space and its Compact Dual

The Siegel space and its compact dual are part of a 1-parameter family of spaces indexed by a real parameter $k \in \mathbb{R}$. When $n = 1$ the symmetric spaces are two-dimensional, and this k is interpreted as their (constant) curvature. That is, this family represents an interpolation between the hyperbolic plane ($k = -1$) and the sphere ($k = 1$) through Euclidean space ($k = 0$) as schematically represented in Figure 2.7. Below we describe the generalization of this to all n, by giving the model, symmetries, and distance functions in terms of the parameter $k \in \mathbb{R}$.

Model: Our models are most similar to the Bounded domain model of Siegel space, and so we use notation to match. For each $k \in \mathbb{R}$ we define the subset $\mathcal{B}_n^k$ of $\mathrm{Sym}(n, \mathbb{C})$ as follows:

$$\mathcal{B}_n^k = \begin{cases} \{W \mid \mathrm{Id} + kW^*W >> 0\} & k < 0 \\ \mathrm{Sym}(n, \mathbb{C}) & k \geq 0 \end{cases}$$

The **basepoint** for $\mathcal{B}_n^k$ is the zero matrix **0** for all k. **Projection back to the model** is analogous to what is done for the bounded domain when $k < 0$, and is just symmetrization for $k \geq 0$.

Isometries: Denote by G^k the isometry group of $\mathcal{B}_n^k$. A uniform description of G^k can be given in close analogy to the description of the symmetries of the compact dual. For each $k \in \mathbb{R}$, $G^k = \mathrm{Sp}(2n, \mathbb{C}) \cap \mathcal{U}^k$ where $\mathcal{U}^k$ is a generalization of the usual unitary group

$$\mathcal{U}^k = \{M \mid M^* \left(\begin{smallmatrix} k\mathrm{Id} & 0 \\ 0 & \mathrm{Id} \end{smallmatrix}\right) M = \left(\begin{smallmatrix} k\mathrm{Id} & 0 \\ 0 & \mathrm{Id} \end{smallmatrix}\right)\}$$

Riemannian Geometry: The Riemannian metric at a point $W \in \mathcal{B}_n^k$ is given by the formula

$$\langle U, V\rangle_W^k = \mathrm{tr}\left(A^{-1}UA^{-1}\overline{V}\right)$$

Where $A = \mathrm{Id} + k\overline{W}W$. As before, this allows us to compute the **Riemannian Gradient** in terms of the Euclidean gradient on $\mathcal{B}_n^k$:

$$\mathrm{grad}(f(W)) = A \cdot \mathrm{grad}_{\mathrm{E}}(f(W)) \cdot A$$

From the Riemannian metric we may explicitly compute the distance function from the basepoint **0** to a real diagonal matrix $D \in \mathcal{B}_n^k$:

$$\mathrm{dist}^k(\mathbf{0}, D) = \begin{cases} \sqrt{\sum_i \frac{\mathrm{arctanh}\left(d_i\sqrt{|k|}\right)^2}{\sqrt{|k|}}} & k < 0 \\ \sqrt{\sum_i d_i^2} & k = 0 \\ \sqrt{\sum_i \frac{\arctan\left(d_i\sqrt{k}\right)^2}{\sqrt{k}}} & k > 0 \end{cases}$$

Distance: The seven step procedure for calculating distance in the compact dual can be modified to give a procedure for the distance in $\mathcal{B}_n^k$. To calculate the Riemannian distance, Step 7 must be replaced with the distance formula above. The only other changes involve the construction of the matrix called M

- In step 2, the computation of P is unchanged but A is replaced with $A = (\mathrm{Id} + kP^2)^{-1/2}$.

- In step 3, the matrix M is replaced with

$$M = \begin{pmatrix} A & -AP \\ \mathrm{sgn}(k)AP & A \end{pmatrix}$$

	2D GRID		TREE	
$(\lvert V\rvert, \lvert E\rvert)$	$(36, 60)$		$(40, 39)$	
	D_{avg}	mAP	D_{avg}	mAP
$\mathcal{S}_3^R$	12.29	100.00	4.27	95.00
$\mathcal{S}_3^{F_\infty}$	0.21	100.00	**2.01**	100.00
$\mathcal{S}_3^{F_1}$	0.02	100.00	2.10	100.00
$\mathcal{B}_3^R$	12.26	100.00	4.14	94.17
$\mathcal{B}_3^{F_\infty}$	0.29	100.00	2.04	100.00
$\mathcal{B}_3^{F_1}$	**0.01**	100.00	2.06	99.17
$\mathcal{D}_3^R$	47.59	54.35	69.65	29.05
$\mathcal{D}_3^{F_\infty}$	63.85	18.94	75.33	15.18
$\mathcal{D}_3^{F_1}$	28.68	82.96	38.84	55.28
$\mathcal{S}_4^R$	12.27	100.00	4.20	98.33
$\mathcal{S}_4^{F_\infty}$	0.49	100.00	1.72	100.00
$\mathcal{S}_4^{F_1}$	**0.01**	100.00	1.58	100.00
$\mathcal{B}_4^R$	12.24	100.00	4.10	100.00
$\mathcal{B}_4^{F_\infty}$	0.17	100.00	**1.18**	100.00
$\mathcal{B}_4^{F_1}$	**0.01**	100.00	1.48	100.00
$\mathcal{D}_4^R$	41.82	78.20	65.95	31.76
$\mathcal{D}_4^{F_\infty}$	53.31	79.34	74.19	19.16
$\mathcal{D}_4^{F_1}$	13.38	100.00	23.64	71.94

Table C.1: Comparison of the compact dual model to the upper half space and bounded domain model on two synthetic datasets.

Where $\operatorname{sgn}(k) = 0$ if $k = 0$. Note the computation of $M.X$ in Step 5 also changes, as M has changed. Now $M.X = (AX - AP)(\operatorname{sgn}(k)APX - A)^{-1}$.

Experiments on the Compact Dual

We perform experiments on small synthetic datasets to compare the performance of the dual model to the upper half Siegel space and the Bounded domain model. Results are reported in Table C.1. We can observe the reduced representation capabilities of the compact dual model, even on small datasets.

Appendix D

Differential Geometry of SPD_n

Orthogonal Diagonalization

Every real symmetric matrix may be orthogonally diagonalized: For every point $P \in \mathrm{SPD}_n$ we may find a positive diagonal matrix D and an orthogonal matrix K such that $P = KDK^T$. This diagonalization has two practical consequences: it allows efficient computation of important SPD_n operations, and provides another means of generalizing Euclidean notions to SPD_n.

With respect to computation, if $P \in \mathrm{SPD}_n$ has orthogonal diagonalization $P = KDK^T$, we may compute its square root and logarithm as $\sqrt{P} = K\sqrt{D}K^T$ and $\log(P) = K\log(D)K^T$ where $\sqrt{D} = \mathrm{diag}(\sqrt{d_1}, \ldots, \sqrt{d_n})$ and $\log(D) = \mathrm{diag}(\log d_1, \ldots, \log d_n)$ for $D = \mathrm{diag}(d_1, \ldots, d_n)$. Similarly, if a tangent vector $U \in S_n$ has orthogonal diagonalization $U = K\Lambda K^T$ (here $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ not necessarily positive definite), the exponential map is computed as $\exp(U) = Ke^{\Lambda}K^T$, where $e^{\Lambda} = \mathrm{diag}(e^{\lambda_1}, \ldots e^{\lambda_n})$.

We verify this in the two lemmas below.

Lemma D.1. If $K \in O(n)$ and X is any $n \times n$ matrix, then $\exp(KXK^T) = K\exp(X)K^T$.

Proof. As K is orthogonal, $K^T = K^{-1}$. Conjugation is an automorphism of the algebra of $n \times n$ matrices, and so applying this to any partial sum of the exponential $\exp(X) = \sum_{n=0}^{\infty} \frac{1}{n!}X^n$ yields

$$\sum_{n=0}^{N} \frac{1}{n!}(KXK^{-1})^n = K\left(\sum_{n=0}^{N} \frac{1}{n!}X^n\right)K^{-1}.$$

Taking the limit of this equality as $N \to \infty$ gives the claimed result. $\square$

Lemma D.2. If $D = \mathrm{diag}(d_1, \ldots, d_n)$ is a diagonal matrix, then $\exp(D) = \mathrm{diag}(e^{d_1}, \ldots, e^{d_n})$.

Proof. The multiplication of diagonal matrices coincides with the elementwise product of their diagonal entries. Again applying this to any partial sum of the exponential of $D = \mathrm{diag}(d_1, \ldots, d_n)$ gives

$$\sum_{n=0}^{N} \frac{1}{n!} \mathrm{diag}(\ldots, d_i \ldots)^n = \mathrm{diag}\left(\ldots, \sum_{n=0}^{N} \frac{1}{n!} d_i^n, \ldots\right).$$

Taking the limit of this equality as $N \to \infty$ gives the claimed result. □

Metric and Isometries

The Riemannian metric on SPD_n is defined as follows: if $U, V \in S_n$ are tangent vectors based at $P \in \mathrm{SPD}_n$, their inner product is:

$$\langle U, V \rangle_P = \mathrm{tr}(P^{-1}UP^{-1}V).$$

Note that at the basepoint, this is just the standard matrix inner product $\langle U, V \rangle_I = \mathrm{tr}(UV^T)$ as U, V are symmetric. We now verify the $GL(n, \mathbb{R})$ action given by M acting as $P \mapsto MPM^T$ is an action by isometries of this metric.

Lemma D.3. The action $f(P) = MPM^T$ extends to tangent vectors U based at P without change in formula: $f_*(U) = MUM^T$

Proof. Let $P \in \mathrm{SPD}_n$ and $U \in S_n$ be a tangent vector based at P. Then by definition, $U = P_0'$ is the derivative of some path P_t of some path of matrices in SPD_n throguh $P_0 = P$. We compute the action of $P \to MPM^T$ on U by taking the derivative of its action on the path:

$$\frac{d}{dt}\Big|_{t=0} MP_tM = MP_t'M\Big|_{t=0} = MUM^T$$

□

Proposition D.1. For every $M \in GL(n; \mathbb{R})$ the transformation $M \mapsto MPM^T$ preserves the Riemannian metric on SPD_n.

Proof. Let $M \in GL(n; \mathbb{R})$ and choose arbitrary point $P \in \mathrm{SPD}_n$, and tangent vectors $U, V \in T_P\,\mathrm{SPD}_n$. We compute the pullback of the metric under the symmetry $f(P) =$

MPM^T. Computing directly from the definition an the previous lemma,

$$\begin{aligned}
f^*\langle U,V\rangle_P &= \langle f_*U, f_*V\rangle_{f(P)} \\
&= \langle MUM^T, MVM^T\rangle_{MPM^T} \\
&= \operatorname{tr}\left(\left(MPM^T\right)^{-1} MUM^T \left(MPM^T\right)^{-1} MVM^T\right) \\
&= \operatorname{tr}\left(M^{-T}P^{-1}UP^{-1}VM^T\right) \\
&= \operatorname{tr}\left(P^{-1}UP^{-1}V\right) \\
&= \langle U,V\rangle_P,
\end{aligned}$$

where the penultimate equality uses that trace is invariant under conjugacy.

□

This provides a vivid geometric interpretation of the previously discussed orthogonal diagonalization operation on SPD_n.

Corollary D.1. Given any $P \in \mathrm{SPD}_n$, there exists a symmetry fixing I which moves P to a diagonal matrix.

This subspace of diagonal matrices plays an essential role in working with SPD_n. As we verify below, the intrinsic geometry of this subspace of diagonal matrices inherited from the Riemannian metric on SPD_n is flat.

Proposition D.2. Let $\mathcal{D} \subset \mathrm{SPD}_n$ be the set of diagonal matrices, and define $f\colon \mathbb{R}^n \to \mathcal{D}$ by $f(x_1,\ldots,x_n) = \operatorname{diag}(e^{x_1},\ldots,e^{x_n})$. Then f is an isometry from the Euclidean metric on $\mathbb{R}^n$ to the metric on $\mathcal{D}$ induced from SPD_n.

Proof. We pull back the metric on $\mathcal{D}$ by f, and see that on $\mathbb{R}^n$ this results in the standard Euclidean metric. Given a point $x \in \mathbb{R}^n$ with tangent vectors $y, z \in \mathbb{R}^n$, we compute this as

$$f^*\langle y,z\rangle_x = \langle f_*y, f_*z\rangle_{f(x)}$$

From the definition of f, we see that the pushforward of y along f is $\operatorname{diag}(\ldots, e^{x_i}y_i, \ldots)$ and similarly for z. Thus we may compute directly and see the result is the standard dot product on $\mathbb{R}^n$.

$$\begin{aligned}
\langle \operatorname{diag}(e^{x_i}y_i), \operatorname{diag}(e^{x_i}z_i)\rangle_{\operatorname{diag}(e^{x_i})} &= \operatorname{tr}\left(\operatorname{diag}(e^{x_i}y_i)\operatorname{diag}(e^{-x_i})\operatorname{diag}(e^{x_i}y_i)\operatorname{diag}(e^{-x_i})\right) \\
&= \operatorname{tr}\left(\operatorname{diag}(y_iz_i)\right) \\
&= \sum_{i=1}^n y_iz_i
\end{aligned}$$

□

This subspace $\mathcal{D}$ is in fact a *maximal flat* for SPD_n, the largest dimensional totally geodesic Euclicean submanifold embedded in SPD_n. For more information on the general theory of symmetric spaces from which the notion of maximal flats arises, see Helgason Helgason (1978). For our purposes, it is only important to note the following fact.

Corollary D.2. The set of diagonal matrices in SPD_n is an isometrically and totally geodesically embedded copy of euclidean n-space.

Exponential Map on SPD_n

The Riemannian exponential map gives a connection between the Euclidean geometry of the tangent space S_n and the curved geometry of SPD_n. It assigns the tangent vector U to the point $Q = \exp(U)$ of SPD_n reached by traveling along the geodesic starting from the basepoint I in direction U for distance $\|U\|$.

As a consequence of non-positive curvature, $\exp$ is a diffeomorphism of S_n onto SPD_n, and so has an inverse: the Riemannian logarithm $\log\colon\ \mathrm{SPD}_n \to S_n$. See Ballmann et al. (1985) for a review of the general theory of manifolds of non-positive curvature. Together, this pair of functions allows one to freely move between 'tangent space coordinates' or the original 'manifold coordinates' which we exploit to transfer Euclidean optimization schemes to SPD_n (see §9.4.3).

Secondly, the geometry of SPD_n is so tightly tied to the algebra of $n \times n$ matrices that the Riemannian exponential agrees exactly with the usual matrix exponential, and the Riemannian logarithm is the matrix logarithm (because of this, we do not distinguish the two notationally), as we verify in the proposition below. Both of these are readily computable via orthogonal diagonalization, as given in §D. This is in stark contrast to general Riemannian manifolds, where the exponential map may have no simple formula.

Proposition D.3. Let $\exp_{\mathrm{Riem}}\colon S_n \to \mathrm{SPD}_n$ be the Riemannian exponential map based at $I \in \mathrm{SPD}_n$, and $\exp$ be the matrix exponential. Then $\exp_{\mathrm{Riem}} = \exp$.

Proof. Let $U \in S_n$ be a tangent vector to SPD_n at the basepoint I, and orthogonally diagonalize as $U = KDK^T$ for some $K \in O(n)$, $D = \mathrm{diag}(d_1, \ldots, d_n)$. As D is tangent to the maximal flat $\mathcal{D}$ of diagonal matrices, the geodesic segment $\exp_{Riem}(tD)$ must be a geodesic in $\mathcal{D}$, which we know from Lemma D.3 to be the coordinate-wise exponential of a straight line in $\mathbb{R}^n$. Precisely, this geodesic is $\mathrm{diag}(\ldots, e^{d_i t}, \ldots)$, and so the original geodesic with initial tangent $U = KDK^T$ is $\exp_{Riem}(tU) = K\,\mathrm{diag}(\ldots, e^{d_i t}, \ldots)K^T$ by

Lemma D.1. Specializing to $t = 1$, this gives the claim:

$$\begin{aligned}\exp_{Riem}(U) &= K \exp_{Riem}(D) K^T \\ &= K \operatorname{diag}(\ldots, e^{d_i}, \ldots) K^{-1} \\ &= K \exp(D) K^{-1} \\ &= \exp(KDK^{-1}) \\ &= \exp(U)\end{aligned}$$

□

This easily transfers to an understanding of the Riemannian exponential at an arbitrary point $P \in \mathrm{SPD}_n$, if we identify the tangent space at P with the symmetric matrices S_n as well.

Corollary D.3. The exponential based at an arbitrary point $P \in \mathrm{SPD}_n$ is given by

$$\exp_{Riem,P}(U) = \sqrt{P} \exp(\sqrt{P^{-1}} U \sqrt{P^{-1}}) \sqrt{P}$$

Proof. Given $P \in \mathrm{SPD}_n$ and tangent vector $U \in T_P \mathrm{SPD}_n$ identified with the set S_n of symmetric matrices, note that $X \mapsto \sqrt{P^{-1}} X \sqrt{P^{-1}}$ is a symmetry of SPD_n taking P to I and U to $\sqrt{P^{-1}} U \sqrt{P}^{-1}$. Using the fact that we understand the Riemannian exponential at the basepoint, we see $\exp_{Riem}(\sqrt{P^{-1}} U \sqrt{P^{-1}}) = \exp(\sqrt{P^{-1}} U \sqrt{P^{-1}})$. It only remains to translate the result back to P, giving the claimed formula. □

Proposition D.4. Let $\log_{\mathrm{Riem}} \colon \mathrm{SPD}_n \to S_n$ be the Riemannian logarithm map based at $0 \in S_n$, and $\log$ be the matrix logarithm (note that while the matrix logaritm is multivalued in general, it is uniquely defined on S_n). Then $\log_{\mathrm{Riem}} = \log$.

Proof. Defined as the inverse of $\exp_{Riem}$, the Riemannian logarithm must satisfy

$$\log_{Riem} \circ \exp_{Riem} = \mathrm{id}_{S_n}$$

Let $U \in S_n$ and orthogonally diagonalize as $U = KDK^T$. Applying the Riemannian exponential, we see $\log_{Riem}(K \exp(D) K^T) = KDK^T$. Recalling from Lemma D.3 the relation between isometries of SPD_n and their application on tangent vectors, we see that we may rewrite the left hand side as $\log_{Riem}(K \exp(D) K^T) = K \log_{Riem}(\exp(D)) K^T$. Appropriately cancelling the factors of K, K^T we arrive at the relationship

$$\log_{Riem}(\exp(D)) = D.$$

That is, restricted to the diagonal matrices, the Riemannian logarithm is an inverse of the matrix exponential, so Riemannian log equals matrix log. Re-absorbing the original factors of K shows the same to be true for any positive definite symmetric matrix; thus $\log_{Riem} = \log$. □

As for the exponential, conjugating by a symmetry moving I to an arbitrary point P, we may describe the Riemannian logarithm at any point of SPD_n.

Corollary D.4. The logarithm based at an arbitrary point $P \in \mathrm{SPD}_n$ is given by

$$\log_{Riem,P}(Q) = \sqrt{P}\log(\sqrt{P^{-1}}Q\sqrt{P^{-1}})\sqrt{P}$$

Vector-Valued Distance on SPD_n

Here we collect useful observations about the vector-valued distance on SPD_n, culminating in a proof of the fact that it is a complete invariant of pairs of points, as claimed in §9.2.1.

Proposition D.5. The vector-valued distance is well-defined: given any pair $P, Q \in \mathrm{SPD}_n$ of points and any two isometries taking P, Q to the basepoint, a diagonal matrix respectively, the diagonal matrices differ at most by a permutation of their entries.

Proof. We see heuristically that there is no remaining continuous degree of freedom by dimension count: the isometry group $GL(n;\mathbb{R})$ has dimension n^2, and we require $\dim(\mathrm{SPD}_n) = n(n+1)/2$ degrees of freedom to translate P to the origin, and a further $\dim O(n) = n(n-1)/2$ degrees of freedom to diagonalize the image of Q while fixing I. As $\dim GL(n;\mathbb{R}) = \dim \mathrm{SPD}_n + \dim O(n)$, there are no remaining continuous degrees of freedom. To see that the remaining ambiguity is precisely permutation of coordinates, note that conjugating a diagonal matrix by an orthogonal matrix results in another diagonal matrix only if the conjugating matrix is a permutation matrix. □

Proposition D.6. If two points $P, Q \in \mathrm{SPD}_n$ have the same vector-valued distance from the basepoint I, then there is an isometry fixing I taking P to Q.

Proof. For two matrices to have the same vector-valued distance from I is equivalent to those two matrices having the same set of eigenvalues. Let $\lambda_1, \ldots, \lambda_n$ be a list of these eigenvalues with multiplicity, and construct two orthonormal bases $(v_i), (w_i)$ of $\mathbb{R}^n$ as follows. For each i, let v_i be an eigenvector of P with eigenvalue λ_i, and w_i an eigenvector of Q with eigenvalue λ_i (in the case the eigenvalues are distinct, such bases are unique up to flipping the sign of each vector, but nontrivial choices must be made in the case of coincident eigenvalues). Given this pair of orthonormal bases, let $K \in O(n)$ be the orthogonal matrix which takes (v_i) to (w_i). It is then an easy observation of linear algebra to note that $Q = KPK^{-1}$, but recalling $K^T = K^{-1}$ we see this is interpreted in the geometry of SPD_n to say that there is an isometry $X \mapsto KXK^T$ fixing I and taking P to Q. □

Combining Propositions D.5 and D.6, after translating appropriately to the basepoint yields the following cornerstone of the theory, showing the vector-valued distance to be the *best possible* invariant.

Corollary D.5. The vector-valued distance is a complete invariant of pairs of points. Two pairs of points (P, Q) and (P', Q') cam be mapped to one another by an isometry if and only if they have the same vector-valued distance.

It is important to note that while the vector-valued distance is not *literally* a metric distance (it is vector-valued, instead of positive-real-number valued, for one) it enjoys some properties analogous to traditional metric distances. For a brief review of some of these (the vector-valued triangle inequality, etc) see Kapovich et al. (2017), and Kapovich et al. (2009).

One property distinguishing the vector-valued distance from traditional metrics is its assymmetry. We will wish to recall this relationship later on, and so prove it here for completeness.

Lemma D.4. For $P, Q \in \mathrm{SPD}_n$, the vector-valued distance satisfies

$$d_{vv}(P, Q) = -d_{vv}(Q, P)$$

with equality understood up to permutation of coordinates.

Proof. The computation of $d_{vv}(P, Q)$ differs from that of $d_{vv}(Q, P)$ in the first step, where we reduce it to computing a function of the eigenvalues of $P^{-1}Q$ or $Q^{-1}P$ respectively. Noting these are inverses of one another, their eigenvalues are reciprocals we may perform the following calculation, where $\{\lambda_i(X)\}$ denotes the eigenvalues of X.

$$\begin{aligned} d_{vv}(Q, P) &= \log(\ldots, \lambda_i(Q^{-1}P), \ldots) \\ &= \log(\ldots, \lambda_i((P^{-1}Q)^{-1}), \ldots) \\ &= \log(\ldots, \lambda_i((P^{-1}Q)^{-1}), \ldots) \\ &= \log\left(\ldots, \frac{1}{\lambda_i((P^{-1}Q))}, \ldots\right) \\ &= -\log(\ldots, \lambda_i((P^{-1}Q)), \ldots) \\ &= -d_{vv}(P, Q) \end{aligned}$$

□

Riemannian Distance on SPD_n

This Riemannian metric allows the computation of the length of curves $\gamma\colon [0,1] \to \mathrm{SPD}_n$ as

$$\mathrm{length}(\gamma) = \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t)\rangle_{\gamma(t)}}\, dt.$$

This in turn induces a distance function $d : \mathrm{SPD}_n \times \mathrm{SPD}_n \to \mathbb{R}$, by taking the infimum of the lengths of all paths joining two points:

$$d(P,Q) = \inf_{\substack{\gamma\colon [0,1]\to \mathrm{SPD}_n \\ \gamma(0)=P,\ \gamma(1)=Q}} \Big\{ \mathrm{length}(\gamma) \Big\}$$

While for general Riemannian manifolds such a distance function may be impossible to explicitly compute, the symmetries of SPD_n provide a readily computable formula.

Proposition D.7. The Riemannian distance from the basepoint I to a point $P \in \mathrm{SPD}_n$ is given by $d(I,P) = \sqrt{\sum_{i=0}^n \log(\lambda_i(P))}$ where $\{\lambda_i(P)\}$ are the eigenvalues of of P.

Proof. Let $P \in \mathrm{SPD}_n$ be arbitrary, and orthogonally diagonalize as $P = KDK^T$. As $K \in O(n)$, the isometry $X \mapsto KDK^T$ fixes I, so the distance $d(I,P)$ equals the distance $d(I,D)$. Note as this action of K is by conjugacy, the diagonal entries d_i of D are precisely the eigenvalues of P. As D lies in the totally geodesic Euclidean subspace $\mathcal{D}$, this distance is realized by the unique Euclidean geodesic connecting I to D. Using Lemma D.2, we may translate to familiar coordinates on $\mathbb{R}^n$ and notice this is the distance from the origin 0 to the point $x = (\log(d_1), \ldots, \log(d_n))$. That is, $d(I,D) = \sqrt{\sum_i \log(d_i)^2}$ as claimed. $\square$

This immediately generalizes to the distance between a pair of arbitrary points, via conjugating by a symmetry moving one to the origin. However, with a little more work one may get a simpler expression for the general distance.

Proposition D.8. The Riemannian distance between two arbitrary points $P, Q \in \mathrm{SPD}_n$ is given by $d(P,Q) = \sqrt{\sum_i \log(\lambda_i(P^{-1}Q))}$ where $\{\lambda_i(P^{-1}Q)\}$ are the eigenvalues of of $P^{-1}Q$.

Proof. If P, Q are arbitrary points in SPD_n, we may use an isometry to translate P to the basepoint, while simultaneously moving Q to $R = \sqrt{P^{-1}}Q\sqrt{P^{-1}}$. As isometries preserve distances, we have $d(P,Q) = d(I,R)$, and by Proposition D.7, this distance is completely determined by the eigenvalues of R. As these are invariant under conjugacy, we replace R with its conjugate by $\sqrt{P^{-1}}$ to get the matrix

$$\begin{aligned} R' &= \sqrt{P} R \sqrt{P}^{-1} \\ &= \sqrt{P^{-1}}\sqrt{P^{-1}} Q \sqrt{P^{-1}}\sqrt{P} \\ &= P^{-1}Q \end{aligned}$$

□

Finally, we give a means of computing the VVD in terms of the gyrocalculus operations as claimed in §9.3.

Proposition D.9. The vector-valued distance from P to Q is the vector of logarithms of the eigenvalues of $(\ominus P) \oplus Q$.

Proof. This is the matrix $(\ominus P) \oplus Q = P^{-1} \oplus Q = \sqrt{P^1} Q \sqrt{P^{-1}}$, which is conjugate to $P^{-1}Q$ (as in D.8), and so has the same eigenvalues. But the logarithm of these eigenvalues is precisely the vector value distance as defined in §9.2.2. □

Finsler Distances on SPD_n

As explained in §2.5.3, the Riemannian distance function on a manifold is completely determined by its Riemannian metric, a choice of inner product on the tangent bundle. Generalizing this, Finsler metrics are the class of distance functions which may be constructed from a smoothly varying choice of norm $\|\cdot\|_F$ on the tangent bundle (which need not be induced by an inner product). The basic theory proceeds in direct analogy to the Riemannian case: the length of a curve $\gamma\colon [0,1] \to \mathrm{SPD}_n$ with respect to a Finsler metric is still defined via integration of this norm along the path,and the distance between points by the infimum of this over all rectifiable curves joining them

$$\mathrm{length}_F(\gamma) = \int_0^1 \|\gamma'\|_F dt, \qquad d_F(P,Q) = \inf_{\substack{\gamma\colon [0,1]\to\mathrm{SPD}_n \\ \gamma(0)=P,\, \gamma(1)=Q}} \Big\{ \mathrm{length}_F(\gamma) \Big\}$$

The geometry of SPD_n allows the computaiton of all Finsler metrics directly from the vector-valued distance. As Riemannian metrics are in particular a special case of Finsler metrics, we begin by recasting our previous observations in this light. In §D we derived a formula for the Riemannian distance function directly from the infintesimal Riemannian metric. But in light of Corollary D.5, since the Riemannian distance is a function which depends only on its input points up to isometry, it must also be recoverable from the vector-valued distance. Indeed, looking at Proposition D.7 we see there is a simple rephrasing to this effect:

Corollary D.6. The Riemannian distance from the basepoint I to an arbitrary point $P \in \mathrm{SPD}_n$ is the Euclidean norm of the vector-valued distance from I to P.

One of the great advantages of higher rank symmetric spaces is the generalizations to which this rephrasing lends itself. Namely, the Euclidean metric is not special in this construction, and any sufficiently symmetric norm on $\mathbb{R}^n$ can induce a distance function on SPD_n in this way.

Proposition D.10. Let $\|-\|$ be any norm on $\mathbb{R}^n$ which is invariant under the permutation of coordinates. Then $\|-\|$ induces a distance function d on SPD_n by

$$d(P,Q)=\|d_{vv}(P,Q)\|$$

Proof. We first note this function is well-defined, as by Proposition D.5 the vector-valued distance of (P,Q) is well-defined up to permutation of coordinates, and our norm is invariant under this symmetry by hypothesis. To see that d is in fact a distance function on SPD_n, we now need to show it satisfies the axioms of a metric:

1. $d(P,Q)\geq 0,\ d(P,Q)=0 \implies P=Q$
2. $d(P,Q)=d(Q,P)$
3. $d(P,R)\leq d(P,Q)+d(Q,R)$

To check the identity of indescernibles (1), note that d is necessarily nonnegative as $\|-\|$ is, and if $d_(P,Q)=0$ then the norm of $d_{vv}(P,Q)$ is zero, so the vector-valued distance itself is zero. But as this is a complete invariant and $d_{vv}(P,P)=0$, this means $P=Q$.

Note that symmetry (2) is not automatic as the vector-valued distance itself is asymmetric. However recalling Lemma D.4, we see that changing the order causes only a global negative sign, and the central symmetry of $\|-\|$, as a virtue of being a norm, gives equality.

The triangle inequality (3) is more subtle, and requires an understanding of the triangle inequality for the vector-valued distance. See the dissertation of Planche Planche (1995), Chapter 6 and the work of Kapovich, Leeb and Millson Kapovich et al. (2009) for details. □

For our experiments, the most important such distances are induced by the ℓ_1 and ℓ_∞ norms on $\mathbb{R}^n$. For completeness, the resulting distance functions are described below.

Proposition D.11. The distance function induced from the ℓ_1 metric applied to the vector-valued distance can be computed as $d_{F_1}(P,Q)=\sum_{i=1}^n|\log\lambda_i(P^{-1}Q)|$, where $\lambda_i(P^{-1}Q)$ runs over the eigenvalues of $P^{-1}Q$.

Proof. The vector-valued distance $d_{vv}(P,Q)$ is the vector of logarithms of the eigenvalues of $R=P^{-1}Q$, and its ℓ^1 norm is the sum of their absolute values:

$$\|(\log(\lambda_1(R),\ldots,\lambda_n(R))\|_{\ell^1}=\sum_{i=1}^n|\log\lambda_i(R)|$$

where $\lambda_i(R)$ is the i^{th} eigenvalaue of R, in decreasing order. □

A similar calculation yields the formula for the F^∞ distance function.

Proposition D.12. The distance function induced from the ℓ_∞ metric applied to the vector-valued distance can be computed as $d_{F_\infty}(P,Q) = \lambda_1(P^{-1}Q)$ where $\lambda_1(-)$ returns the largest eigenvalue of the input matrix.

Relations with Other SPD_n Metrics

Other distances previously used in the literature can be reconstructed from the vector-valued distance, by applying a suitable function:

The *Affine Invariant metric* of Pennec et al. (2006) is nothing but the usual Riemannian metric discussed in §D.

The *symmetric Stein divergence* Sra (2012), is given by

$$S(P,Q) := \log\det\frac{P+Q}{2} - \frac{1}{2}\log\det(PQ)$$

This can be computed from the vector-valued distance

$$d_{vv}(P,Q) = \log(\lambda_1(P^{-1}Q),\dots,\lambda_n(P^{-1}Q))$$

by applying the function

$$\|v\|_S = \sum_{i=1}^{n}\log\frac{e^{-v_i/2}+e^{v_i/2}}{2}.$$

Indeed

$$\begin{aligned} S(P,Q) &= \log\det\tfrac{P+Q}{2} - \tfrac{1}{2}\log\det(PQ) \\ &= \log\det P\left(\tfrac{\mathrm{Id}+P^{-1}Q}{2}\right) - \log\det(P\sqrt{P^{-1}Q}) \\ &= \log\det\tfrac{\mathrm{Id}+P^{-1}Q}{2} - \log\det(\sqrt{P^{-1}Q}) \\ &= \textstyle\sum_{i=1}^{n}\log\lambda_i\left(\tfrac{\mathrm{Id}+P^{-1}Q}{2\sqrt{P^{-1}Q}}\right) \\ &= \textstyle\sum_{i=1}^{n}\log\left(\tfrac{\lambda_i(P^{-1}Q)^{-1/2}+\lambda_i(P^{-1}Q)^{1/2}}{2}\right) \end{aligned}$$

In particular we obtain, thanks to the vector-valued distance, a more direct proof of Sra (2015).

Instead the *Log-Euclidean* metric d_{LE} Arsigny et al. (2006b,a) is flat, and as such doesn't reflect the curved geometry of SPD. More precisely d_{LE} is the pushforward, through the exponential map $\exp_{Riem} : S_n \to \mathrm{SPD}$ of the Euclidean metric on S_n. As a result, for this choice (SPD_n, d_{LE}) is *isometric* to the flat manifold S_n. Since the group $GL(n,\mathbb{R})$ does not act by isometries on $(\mathrm{SPD}_n; d_{LE})$, and the distance is therefore not related to the vector-valued distance nor can be computed from it.

Similarly the *Bures-Wasserstein* metric d_{BW} inspired from quantum information theory

Bhatia et al. (2019) leads to a non-negatively curved manifold, and thus, again, has a different isometry group. More precisely

$$d_{BW}(P,Q) = \sqrt{\operatorname{tr}(P) + \operatorname{tr}(Q) - 2\sqrt{\operatorname{tr}(PQ)}}.$$

It is computed in (Bhatia et al., 2019, Page 15) that the group of isometries of (SPD_n, d_{BW}) is reduced to $O(n)$. As a result, once again, d_{BW} cannot be reconstructed from d_{vv}.

Computational Complexity of Operations

In this section we discuss the computational theoretical complexity of the different operations involved in the development of this work. We employ Big O notation[1]. Since in all cases operations are not nested, but are applied sequentially, the costs can be added resulting in a polynomial expression. Thus, by applying the properties of the notation, we disregard lower-order terms of the polynomial.

Matrix Operations: For $n \times n$ matrices, the associated complexity of each operation is as follows:[2]

- Addition and subtraction: $\mathcal{O}(n^2)$
- Multiplication: $\mathcal{O}(n^{2.4})$
- Inversion: $\mathcal{O}(n^{2.4})$
- Diagonalization: $\mathcal{O}(n^3)$

SPD Operations: For $n \times n$ SPD matrices, the associated complexity of each operation is as follows:

- Exp/Log map: $\mathcal{O}(n^3)$, due to diagonalizations.
- Gyro-Addition: $\mathcal{O}(n^{2.4})$, due to matrix multiplications
- Matrix Scaling: $\mathcal{O}(n^3)$, due to exp and log maps.
- Isometries: $\mathcal{O}(n^{2.4})$, due to matrix multiplications.

[1]https://en.wikipedia.org/wiki/Big_O_notation

[2]https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations

Distance Calculation: The full computation of the distance algorithm in SPD_n involves matrix square root, inverses, multiplications, and diagonalizations. Since they are applied sequentially, without affecting the dimensionality of the matrices, we can take the highest value as the asymptotic cost of the algorithm, which is $\mathcal{O}(n^3)$.

Appendix E

Code and Data Used in this Thesis

The code and data used in this thesis has been published with the following digital object identifiers:

- Federico López (2023). *Source code and data for the PhD Thesis "Learning Neural Graph Representations in Non-Euclidean Geometries"*. DOI: `10.11588/data/KOAMK4`, URL: `https://doi.org/10.11588/data/KOAMK4`

Bibliography

Abhishek, A., Anand, A., and Awekar, A. Fine-grained entity type classification by jointly learning representations and label embeddings. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 797–807, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1075.

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009. ISBN 9781400830244. doi: doi:10.1515/9781400830244. URL https://doi.org/10.1515/9781400830244.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory*, ICDT '01, pp. 420–434, Berlin, Heidelberg, 2001. Springer-Verlag. ISBN 3540414568.

Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., and Smola, A. J. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pp. 37–48, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351. doi: 10.1145/2488388.2488393. URL https://doi.org/10.1145/2488388.2488393.

Ai, Q., Azizi, V., Chen, X., and Zhang, Y. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9), September 2018. ISSN 1999-4893. doi: 10.3390/a11090137.

Allen, C., Balazevic, I., and Hospedales, T. Interpreting knowledge graph relation representation from word embeddings. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=gLWj29369lW.

Arsigny, V., Fillard, P., Pennec, X., and Ayache, N. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, 2006a. doi: https://doi.org/10.1002/mrm.20965. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mrm.20965.

Arsigny, V., Fillard, P., Pennec, X., and Ayache, N. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM J. Matrix Anal. Appl.*, 29(1):328–347, 2006b. URL http://dblp.uni-trier.de/db/journals/siammax/siammax29.html#ArsignyFPA06.

Bachmann, G., Becigneul, G., and Ganea, O.-E. Constant curvature graph convolutional networks. In *37th International Conference on Machine Learning (ICML)*, 2020.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Balazevic, I., Allen, C., and Hospedales, T. Multi-relational poincaré graph embeddings. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 4463–4473. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f8b932c70d0b2e6bf071729a4fa68dfc-Paper.pdf.

Ballmann, W., Brin, M., and Eberlein, P. Structure of manifolds of nonpositive curvature. I. *Annals of Mathematics*, 122(1):171–203, 1985. ISSN 0003486X. URL http://www.jstor.org/stable/1971373.

Bécigneul, G. and Ganea, O. Riemannian adaptive optimization methods. In *7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019. URL https://openreview.net/forum?id=r1eiqi09K7.

Belkin, M. and Niyogi, P. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pp. 585–591, Cambridge, MA, USA, 2001. MIT Press.

Bhatia, R., Jain, T., and Lim, Y. On the Bures–Wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191, 2019. doi: 10.1016/j.exmath.2018.01.002. URL https://app.dimensions.ai/details/publication/pub.1100291552andhttp://arxiv.org/pdf/1712.01504.

Boguñá, M., Papadopoulos, F., and Krioukov, D. Sustaining the internet with hyperbolic mapping. *Nature Communications*, 1, 62, 2010.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605581026. doi: 10.1145/1376616.1376746. URL https://doi.org/10.1145/1376616.1376746.

Bonnabel, S. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58, 11 2011. doi: 10.1109/TAC.2013.2254619.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 26, pp. 2787–2795. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

Borges, J. L. *Other inquisitions*. University of Texas Press, 1964.

Brams, A. H., Jakobsen, A. L., Jendal, T. E., Lissandrini, M., Dolog, P., and Hose, K. Mindreader: Recommendation over knowledge graph entities with explicit user ratings. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pp. 2975–2982, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412759. URL https://doi.org/10.1145/3340531.3412759.

Bridson, M. and Häfliger, A. *Metric Spaces of Non-Positive Curvature*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2011. ISBN 9783540643241. URL https://books.google.de/books?id=3DjaqB08AwAC.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Brooks, D., Schwander, O., Barbaresco, F., Schneider, J.-Y., and Cord, M. Riemannian batch normalization for SPD neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 15489–15500. Curran Associates, Inc., 2019a. URL https://proceedings.neurips.cc/paper/2019/file/6e69ebbfad976d4637bb4b39de261bf7-Paper.pdf.

Brooks, D. A., Schwander, O., Barbaresco, F., Schneider, J.-Y., and Cord, M. Exploring complex time-series representations for riemannian machine learning of radar data. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3672–3676, 2019b. doi: 10.1109/ICASSP.2019.8683056.

Candès, E. and Recht, B. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, June 2012. ISSN 0001-0782. doi: 10.1145/2184319.2184343. URL https://doi.org/10.1145/2184319.2184343.

Cannon, J. W., Floyd, W. J., Kenyon, R., and Parry, W. R. *Hyperbolic Geometry*, volume 31. Flavors of Geometry, 1997.

Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

Cao, S., Lu, W., and Xu, Q. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pp. 891–900, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450337946. doi: 10.1145/2806416.2806512. URL https://doi.org/10.1145/2806416.2806512.

Carreira, J., Caseiro, R., Batista, J., and Sminchisescu, C. Semantic segmentation with second-order pooling. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C. (eds.), *Computer Vision – ECCV 2012*, pp. 430–443, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33786-4.

Cayley, A. Sur quelques propriétés des déterminants gauches. *Journal für die reine und angewandte Mathematik*, 32:119–123, 1846. URL http://www.digizeitschriften.de/dms/img/?PID=GDZPPN002145308.

Chakraborty, R., Yang, C.-H., Zhen, X., Banerjee, M., Archer, D., Vaillancourt, D., Singh, V., and Vemuri, B. A statistical recurrent model on the manifold of symmetric positive definite matrices. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/7070f9088e456682f0f84f815ebda761-Paper.pdf.

Chakraborty, R., Bouza, J., Manton, J., and Vemuri, B. C. Manifoldnet: A deep neural network for manifold-valued data with applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020. doi: 10.1109/TPAMI.2020.3003846.

Chamberlain, B., Deisenroth, M., and Clough, J. Neural embeddings of graphs in hyperbolic space. In *Proceedings of the 13th International Workshop on Mining and Learning with Graphs (MLG)*, 2017.

Chamberlain, B. P., Hardwick, S. R., Wardrope, D. R., Dzogang, F., Daolio, F., and Vargas, S. Scalable hyperbolic recommender systems. *CoRR*, abs/1902.08648, 2019. URL http://arxiv.org/abs/1902.08648.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems 32*, pp. 4869–4880. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/0415740eaa4d9decbc8da001d3fd805f-Paper.pdf.

Chami, I., Abu-El-Haija, S., Perozzi, B., Ré, C., and Murphy, K. Machine learning on graphs: A model and comprehensive taxonomy. *CoRR*, abs/2005.03675, 2020a. URL https://arxiv.org/abs/2005.03675.

Chami, I., Gu, A., Chatziafratis, V., and Ré, C. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b.

Chami, I., Wolf, A., Juan, D.-C., Sala, F., Ravi, S., and Ré, C. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6901–6914, Online, July 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.617. URL https://www.aclweb.org/anthology/2020.acl-main.617.

Chau, D. H., Nachenberg, C., Wilhelm, J., Wright, A., and Faloutsos, C. Polonium: Tera-scale graph mining and inference for malware detection. In *SIAM INTERNATIONAL CONFERENCE ON DATA MINING (SDM)*, pp. 131–142, 2011.

Chen, B., Huang, X., Xiao, L., Cai, Z., and Jing, L. Hyperbolic interaction model for hierarchical multi-label classification. *CoRR*, abs/1905.10802, May 2019. URL https://arxiv.org/abs/1905.10802.

Chen, B., Fu, Y., Xu, G., Xie, P., Tan, C., Chen, M., and Jing, L. Probing {bert} in hyperbolic spaces. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=17VnwXYZyhH.

Chen, Y., Yang, M., Zhang, Y., Zhao, M., Meng, Z., Hao, J., and King, I. Modeling scale-free graphs for knowledge-aware recommendation, 2021b.

Chen, Z., Hendrix, W., and Samatova, N. F. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems*, 39(1):59–85, Aug 2012. ISSN 1573-7675. doi: 10.1007/s10844-011-0183-2. URL https://doi.org/10.1007/s10844-011-0183-2.

Cho, H., DeMeo, B., Peng, J., and Berger, B. Large-margin classification in hyperbolic space. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1832–1840. PMLR, 16–18 Apr 2019. URL http://proceedings.mlr.press/v89/cho19a.html.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Choi, E., Levy, O., Choi, Y., and Zettlemoyer, L. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 87–96, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-1009.

Cohen, N., Coudert, D., and Lancin, A. On computing the gromov hyperbolicity. *ACM J. Exp. Algorithmics*, 20, August 2015. ISSN 1084-6654. doi: 10.1145/2780652. URL https://doi.org/10.1145/2780652.

Cox, M. A. A. and Cox, T. F. *Multidimensional Scaling*, pp. 315–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-33037-0. doi: 10.1007/978-3-540-33037-0_14. URL http://dx.doi.org/10.1007/978-3-540-33037-0_14.

Cruceru, C., Becigneul, G., and Ganea, O.-E. Computationally tractable Riemannian manifolds for graph embeddings. In *37th International Conference on Machine Learning (ICML)*, 2020.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, December 1989. ISSN 0932-4194. doi: 10.1007/BF02551274. URL http://dx.doi.org/10.1007/BF02551274.

Davidson, E. H., Rast, J. P., Oliveri, P., Ransick, A., Calestani, C., Yuh, C.-H., Minokawa, T., Amore, G., Hinman, V., Arenas-Mena, C., Otim, O., Brown, C. T., Livi, C. B., Lee, P. Y., Revilla, R., Rust, A. G., Pan, Z. j., Schilstra, M. J., Clarke, P. J. C., Arnone, M. I., Rowen, L., Cameron, R. A., McClay, D. R., Hood, L., and Bolouri, H. A genomic regulatory network for development. *Science*, 295(5560):1669–1678, 2002. ISSN 0036-8075. doi: 10.1126/science.1069883. URL https://science.sciencemag.org/content/295/5560/1669.

Defferrard, M., Milani, M., Gusset, F., and Perraudin, N. DeepSphere: A graph-based spherical CNN. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1e3OlStPB.

Del Corro, L., Abujabal, A., Gemulla, R., and Weikum, G. Finet: Context-aware fine-grained named entity typing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 868–878, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1103. URL https://www.aclweb.org/anthology/D15-1103.

Dettmers, T., Pasquale, M., Pontus, S., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pp. 1811–1818, February 2018. URL https://arxiv.org/abs/1707.01476.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

Dhingra, B., Shallue, C., Norouzi, M., Dai, A., and Dahl, G. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pp. 59–69, New Orleans, Louisiana, USA, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-1708. URL https://www.aclweb.org/anthology/W18-1708.

Dong, Z., Jia, S., Zhang, C., Pei, M., and Wu, Y. Deep manifold learning of symmetric positive definite matrices with application to face recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 4009–4015. AAAI Press, 2017.

Donnat, C., Zitnik, M., Hallac, D., and Leskovec, J. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pp. 1320–1329, New

York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3220025. URL https://doi.org/10.1145/3219819.3220025.

Donoho, D. L. and Tsaig, Y. Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. *IEEE Trans. Information Theory*, 54(11):4789–4812, 2008.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Dunham, D. Transformation of hyperbolic escher patterns. *Visual Mathematics*, 1(1):0–0, 1999. URL http://eudml.org/doc/256761.

Edelman, A., Arias, T. A., and Smith, S. T. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl.*, 20(2):303–353, April 1999. ISSN 0895-4798. doi: 10.1137/S0895479895290954. URL https://doi.org/10.1137/S0895479895290954.

Euler, L. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

Falkenberg, A. Method to calculate the inverse of a complex matrix using real matrix inversion. 2007.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *The World Wide Web Conference*, WWW '19, pp. 417–426, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313488. URL https://doi.org/10.1145/3308558.3313488.

Fournier, H., Ismail, A., and Vigneron, A. Computing the gromov hyperbolicity of a discrete metric space. *Inf. Process. Lett.*, 115(6):576–579, June 2015. ISSN 0020-0190. doi: 10.1016/j.ipl.2015.02.002. URL https://doi.org/10.1016/j.ipl.2015.02.002.

Gainza, P., Sverrisson, F., Monti, F., Rodolà, E., Boscaini, D., Bronstein, M., and Correia, B. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17:1–9, 02 2020. doi: 10.1038/s41592-019-0666-6.

Ganea, O. *Non-Euclidean Neural Representation Learning of Words, Entities and Hierarchies*. PhD thesis, ETH Zürich, Zürich, Switzerland, 2019.

Ganea, O., Becigneul, G., and Hofmann, T. Hyperbolic entailment cones for learning hierarchical embeddings. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th*

International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pp. 1646–1655, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018a. PMLR. URL http://proceedings.mlr.press/v80/ganea18a.html.

Ganea, O., Becigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems 31*, pp. 5345–5355. Curran Associates, Inc., 2018b. URL http://papers.nips.cc/paper/7780-hyperbolic-neural-networks.pdf.

Gao, J. and Guibas, L. Geometric algorithms for sensor networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1958):27–51, 2012. doi: 10.1098/rsta.2011.0215. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2011.0215.

Gao, Z., Wu, Y., Bu, X., Yu, T., Yuan, J., and Jia, Y. Learning a robust representation via a deep network on symmetric positive definite manifolds. *Pattern Recognition*, 92:1–12, 2019. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2019.03.007. URL https://www.sciencedirect.com/science/article/pii/S0031320319301062.

Gillick, D., Lazic, N., Ganchev, K., Kirchner, J., and Huynh, D. Context-Dependent Fine-Grained Entity Type Tagging. *ArXiv e-prints*, December 2014.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Godec, P. Graph embeddings: The summary, Dec 2018. URL https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007.

Goh, K.-I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., and Barabási, A.-L. The human disease network. *Proceedings of the National Academy of Sciences*, 104(21):8685–8690, 2007.

Goodfellow, I. J., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

Grattarola, D., Zambon, D., Livi, L., and Alippi, C. Change detection in graph streams by learning graph embeddings on constant-curvature manifolds. *IEEE Trans. Neural Networks Learn. Syst.*, 31(6):1856–1869, 2020. doi: 10.1109/TNNLS.2019.2927301. URL https://doi.org/10.1109/TNNLS.2019.2927301.

Gromov, M. *Hyperbolic Groups*, pp. 75–263. Springer New York, New York, NY, 1987. ISBN 978-1-4613-9586-7. doi: 10.1007/978-1-4613-9586-7_3. URL https://doi.org/10.1007/978-1-4613-9586-7_3.

Grover, A. and Leskovec, J. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 855–864, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939754. URL https://doi.org/10.1145/2939672.2939754.

Gu, A., Sala, F., Gunel, B., and Ré, C. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJxeWnCcF7.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., and de Freitas, N. Hyperbolic attention networks. In *7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019. URL https://openreview.net/forum?id=rJxHsjRqFQ.

Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., and He, Q. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020. doi: 10.1109/TKDE.2020.3028705.

Ha Quang, M., San Biagio, M., and Murino, V. Log-Hilbert-Schmidt metric between positive definite operators on Hilbert spaces. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/f7664060cc52bc6f3d620bcedc94a4b6-Paper.pdf.

Hagberg, A. A., Schult, D. A., and Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In Varoquaux, G., Vaught, T., and Millman, J. (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.

Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Harandi, M. T., Salzmann, M., and Hartley, R. From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. (eds.), *Computer Vision – ECCV 2014*, pp. 17–32, Cham, 2014. Springer International Publishing.

Harper, F. M. and Konstan, J. A. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pp. 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052569. URL https://doi.org/10.1145/3038912.3052569.

Helgason, S. *Differential geometry, Lie groups, and symmetric spaces*. Academic Press New York, 1978. ISBN 0123384605.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989. ISSN 0893-6080.

Hsieh, C.-K., Yang, L., Cui, Y., Lin, T.-Y., Belongie, S., and Estrin, D. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pp. 193–201, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052639. URL https://doi.org/10.1145/3038912.3052639.

Hu, Z., Huang, P., Deng, Y., Gao, Y., and Xing, E. Entity hierarchy embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1292–1300, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1125. URL https://www.aclweb.org/anthology/P15-1125.

Huang, Z. and Gool, L. V. A Riemannian network for SPD matrix learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 2036–2042. AAAI Press, 2017.

Huang, Z., Wang, R., Shan, S., and Chen, X. Learning Euclidean-to-Riemannian metric for point-to-set classification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1677–1684, 2014. doi: 10.1109/CVPR.2014.217.

Huang, Z., Wang, R., Shan, S., Li, X., and Chen, X. Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 720–729. JMLR.org, 2015.

Huang, Z., Wu, J., and Gool, L. V. Building deep networks on Grassmann manifolds. In McIlraith, S. A. and Weinberger, K. Q. (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 3279–3286. AAAI Press, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16846`.

Ionescu, C., Vantzos, O., and Sminchisescu, C. Matrix backpropagation for deep networks with structured layers. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2965–2973, 2015. doi: 10.1109/ICCV.2015.339.

Kapovich, M., Leeb, B., and Millson, J. Convex functions on symmetric spaces, side lengths of polygons and the stability inequalities for weighted configurations at infinity. *J. Differential Geom.*, 2009.

Kapovich, M., Leeb, B., and Porti, J. Anosov subgroups: dynamical and geometric characterizations. *European Journal of Mathematics*, 3(3):808–898, 2017. doi: 10.1007/s40879-017-0192-y. URL `https://doi.org/10.1007/s40879-017-0192-y`.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, abs/1412.6980, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Kochurov, M., Karimov, R., and Kozlukov, S. Geoopt: Riemannian optimization in PyTorch. *ArXiv*, abs/2005.02819, 2020.

Kolyvakis, P., Kalousis, A., and Kiritsis, D. Hyperbolic knowledge graph embeddings for knowledge base completion. In Harth, A., Kirrane, S., Ngonga Ngomo, A.-C., Paulheim, H., Rula, A., Gentile, A. L., Haase, P., and Cochez, M. (eds.), *The Semantic Web*, pp. 199–214, Cham, 2020. Springer International Publishing.

Krioukov, D., Papadopoulos, F., Vahdat, A., and Boguñá, M. On Curvature and Temperature of Complex Networks. *Physical Review E*, 80(035101), Sep 2009.

Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguñá, M. Hyperbolic geometry of complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 82:036106, 09 2010. doi: 10.1103/PhysRevE.82.036106.

Kruskal, J. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

Lacroix, T., Usunier, N., and Obozinski, G. Canonical tensor decomposition for knowledge base completion. In Dy, J. and Krause, A. (eds.), *Proceedings of Machine Learning Research*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2863–2872, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/lacroix18a.html.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://www.aclweb.org/anthology/N16-1030.

Law, M. T. and Stam, J. Ultrahyperbolic representation learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/123b7f02433572a0a560e620311a469c-Abstract.html.

Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A.-L., Brewer, D., Christakis, N., Contractor, N., Fowler, J., and Gutmann, M. Life in the network: The coming age of computational social science. *Science (New York)*, 323, 01 2009.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1 (4):541–551, December 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.4.541. URL https://doi.org/10.1162/neco.1989.1.4.541.

Lee, J. M. *Riemannian Manifolds: An Introduction to Curvature*. Graduate Texts in Mathematics. Springer New York, 1997. ISBN 9780387982717. URL https://books.google.de/books?id=ZRQgH7FQafgC.

Lee, J. M. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer New York, 2012. ISBN 978-1-4899-9475-2.

Lee, K., He, L., Lewis, M., and Zettlemoyer, L. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 188–197, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1018. URL https://www.aclweb.org/anthology/D17-1018.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015. URL http://dblp.uni-trier.de/db/journals/semweb/semweb6.html#LehmannIJJKMHMK15.

Leimeister, M. and Wilson, B. J. Skip-gram word embeddings in hyperbolic space. *CoRR*, abs/1809.01498, 2018. URL http://arxiv.org/abs/1809.01498.

Li, J., Zhang, L., Meng, F., and Li, F. Recommendation algorithm based on link prediction and domain knowledge in retail transactions. *Procedia Computer Science*, 31:875 – 881, 2014. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2014.05.339. URL http://www.sciencedirect.com/science/article/pii/S187705091400516X. 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014.

Li, P., Xie, J., Wang, Q., and Gao, Z. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 947–955, 2018. doi: 10.1109/CVPR.2018.00105.

Li, Y. and Lu, R. Locality preserving projection on SPD matrix lie group: algorithm and analysis. *Sci. China Inf. Sci.*, 61(9):092104:1–092104:15, 2018. doi: 10.1007/s11432-017-9233-4. URL https://doi.org/10.1007/s11432-017-9233-4.

Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. Graph matching networks for learning the similarity of graph structured objects. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of

Proceedings of Machine Learning Research, pp. 3835–3845. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/li19d.html.

Ling, X. and Weld, D. S. Fine-grained entity recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pp. 94–100. AAAI Press, 2012. URL http://dl.acm.org/citation.cfm?id=2900728.2900742.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems 32*, pp. 8228–8239. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9033-hyperbolic-graph-neural-networks.pdf.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., and Song, L. SphereFace: Deep hypersphere embedding for face recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6738–6746, 2017. doi: 10.1109/CVPR.2017.713.

López, F. and Strube, M. A fully hyperbolic neural model for hierarchical multi-class classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 460–475, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.findings-emnlp.42.

López, F., Heinzerling, B., and Strube, M. Fine-grained entity typing in hyperbolic space. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pp. 169–180, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4319. URL https://www.aclweb.org/anthology/W19-4319.

López, F., Pozzetti, B., Trettel, S., Strube, M., and Wienhard, A. Vector-valued distance and gyrocalculus on the space of symmetric positive definite matrices. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 34. Curran Associates, Inc., 2021a.

López, F., Pozzetti, B., Trettel, S., Strube, M., and Wienhard, A. Symmetric spaces for graph embeddings: A finsler-riemannian approach. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7090–7101. PMLR, 18–24 Jul 2021b. URL http://proceedings.mlr.press/v139/lopez21a.html.

López, F., Pozzetti, B., Trettel, S., and Wienhard, A. Hermitian symmetric spaces for graph embeddings, 2021c.

López, F., Scholz, M., Yung, J., Pellat, M., Strube, M., and Dixon, L. Augmenting the user-item graph with textual similarity models, 2021d.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Ma, W., Zhang, M., Cao, Y., Jin, W., Wang, C., Liu, Y., Ma, S., and Ren, X. Jointly learning explainable rules for recommendation with knowledge graph. In *The World Wide Web Conference*, WWW '19, pp. 1210–1221, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366748. doi: 10.1145/3308558.3313607. URL https://doi.org/10.1145/3308558.3313607.

Ma, Y., Cambria, E., and GAO, S. Label embedding for zero-shot fine-grained named entity typing. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pp. 171–180, Osaka, Japan, December 2016. URL https://www.aclweb.org/anthology/C16-1017.

Mao, Y., Wang, R., Shan, S., and Chen, X. Cosonet: Compact second-order network for video face recognition. In Jawahar, C. V., Li, H., Mori, G., and Schindler, K. (eds.), *Computer Vision – ACCV 2018*, pp. 51–67, Cham, 2019. Springer International Publishing. ISBN 978-3-030-20893-6.

Martinet, L.-E., Kramer, M. A., Viles, W., Perkins, L. N., Spencer, E., Chu, C. J., Cash, S. S., and Kolaczyk, E. D. Robust dynamic community detection with applications to human brain functional networks. *Nature Communications*, 11(1):2785, Jun 2020. ISSN 2041-1723. doi: 10.1038/s41467-020-16285-7. URL https://doi.org/10.1038/s41467-020-16285-7.

McAuley, J. and Leskovec, J. Learning to discover social circles in ego networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 539–547, Red Hook, NY, USA, 2012. Curran Associates Inc.

McAuley, J. and Leskovec, J. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pp. 165–172, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324090. doi: 10.1145/2507157.2507163. URL https://doi.org/10.1145/2507157.2507163.

McCann, B., Bradbury, J., Xiong, C., and Socher, R. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pp. 6297–6308, 2017.

McPherson, M., Smith-Lovin, L., and Cook, J. M. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444, 2001. doi: 10.1146/annurev.

soc.27.1.415. URL `http://arjournals.annualreviews.org/doi/abs/10.1146/annurev.soc.27.1.415`.

Meng, Y., Huang, J., Wang, G., Zhang, C., Zhuang, H., Kaplan, L., and Han, J. Spherical text embedding. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 8208–8217. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/043ab21fc5a1607b381ac3896176dac6-Paper.pdf`.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pp. 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

Miller, G. A. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992. URL `https://www.aclweb.org/anthology/H92-1116`.

Mishra, P., Piktus, A., Goossen, G., and Silvestri, F. Node masking: Making graph neural networks generalize and scale better, 2021.

Moor, M., Horn, M., Rieck, B., and Borgwardt, K. Topological autoencoders. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7045–7054. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/moor20a.html`.

Murty, S., Verga, P., Vilnis, L., and McCallum, A. Finer grained entity typing with typenet. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, 2017. URL `http://arxiv.org/abs/1711.05795`.

Murty, S., Verga, P., Vilnis, L., Radovanovic, I., and McCallum, A. Hierarchical losses and new resources for fine-grained entity typing and linking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 97–109, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P18-1010`.

Nash, J. The imbedding problem for riemannian manifolds. *Annals of Mathematics*, 63(1): 20–63, 1956.

Nguyen, X. S., Brun, L., Lezoray, O., and Bougleux, S. A neural network based on SPD manifold learning for skeleton-based hand gesture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Ni, C., Lin, Y., Gao, J., David Gu, X., and Saucan, E. Ricci curvature of the internet topology. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 2758–2766, 2015. doi: 10.1109/INFOCOM.2015.7218668.

Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL `https://www.aclweb.org/anthology/D19-1018`.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6341–6350. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/59dfa2df42d9e3d41f5b02bfc32229dd-Paper.pdf`.

Nickel, M. and Kiela, D. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3779–3788, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/nickel18a.html`.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016. doi: 10.1109/JPROC.2015.2483592.

Nielsen, F. and Sun, K. *Clustering in Hilbert's Projective Geometry: The Case Studies of the Probability Simplex and the Elliptope of Correlation Matrices*, pp. 297–331. Springer International Publishing, Cham, 2019. ISBN 978-3-030-02520-5. doi: 10.1007/978-3-030-02520-5_11. URL `https://doi.org/10.1007/978-3-030-02520-5_11`.

Nooy, W. D., Mrvar, A., and Batagelj, V. *Exploratory Social Network Analysis with Pajek*. Cambridge University Press, USA, 2011. ISBN 0521174805.

Ollivier, Y. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810 – 864, 2009. ISSN 0022-1236. doi: https://doi.org/10.1016/j.jfa.

2008.11.001. URL http://www.sciencedirect.com/science/article/pii/S002212360800493X.

Onoe, Y. and Durrett, G. Learning to denoise distantly-labeled data for entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 2407–2417, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1250.

Ontrup, J. and Ritter, H. Hyperbolic self-organizing maps for semantic navigation. In Dietterich, T., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems*, volume 14, pp. 1417–1424. MIT Press, 2002. URL https://proceedings.neurips.cc/paper/2001/file/093b60fd0557804c8ba0cbf1453da22f-Paper.pdf.

Pandit, S., Chau, D. H., Wang, S., and Faloutsos, C. Netprobe: A fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 201–210, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242600. URL https://doi.org/10.1145/1242572.1242600.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-librar
pdf.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pennec, X., Fillard, P., and Ayache, N. A Riemannian framework for tensor computing. *Int. J. Comput. Vision*, 66(1):41–66, January 2006. ISSN 0920-5691. doi: 10.1007/s11263-005-3222-z. URL https://doi.org/10.1007/s11263-005-3222-z.

Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pp. 701–710, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329569. doi: 10.1145/2623330.2623732. URL https://doi.org/10.1145/2623330.2623732.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Pham, T. N., Li, X., Cong, G., and Zhang, Z. A general graph-based model for recommendation in event-based social networks. In *2015 IEEE 31st International Conference on Data Engineering*, pp. 567–578, 2015. doi: 10.1109/ICDE.2015.7113315.

Planche, P. *Géométrie de Finsler sur les espaces symétriques*. PhD thesis, Université de Genève, Geneve, Switzerland, 7 1995.

Ratliff, N. D., Wyk, K. V., Xie, M., Li, A., and Rana, M. A. Generalized nonlinear and Finsler geometry for robotics. *CoRR*, abs/2010.14745, 2020. URL https://arxiv.org/abs/2010.14745.

Rau, L. F. Extracting company names from text. volume 1, pp. 29–32. IEEE, 1991.

Reif, E., Yuan, A., Wattenberg, M., Viegas, F. B., Coenen, A., Pearce, A., and Kim, B. Visualizing and measuring the geometry of bert. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/159c1ffe5b61b41b3c4d8f4c2150f6c4-Paper.pdf.

Ren, X., He, W., Qu, M., Huang, L., Ji, H., and Han, J. Afet: Automatic fine-grained entity typing by hierarchical partial-label embedding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1369–1378, Austin, Texas, November 2016a. Association for Computational Linguistics. doi: 10.18653/v1/D16-1144. URL https://www.aclweb.org/anthology/D16-1144.

Ren, X., He, W., Qu, M., Voss, C. R., Ji, H., and Han, J. Label noise reduction in entity typing by heterogeneous partial-label embedding. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 1825–1834, New York, NY, USA, 2016b. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939822. URL http://doi.acm.org/10.1145/2939672.2939822.

Ricci, G. Direzioni e invarianti principali in una varieta qualunque. *Atti Reale Ist. Veneto*, 63:1233–1239, 1904.

Rossi, R. A. and Ahmed, N. K. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL http://networkrepository.com.

Roweis, S. T. and Saul, L. K. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323. URL http://www.sciencemag.org/cgi/content/abstract/290/5500/2323.

Rubin-Delanchy, P. Manifold structure in graph embeddings, 2020. URL https://arxiv.org/abs/2006.05168.

Said, S., Bombrun, L., Berthoumieu, Y., and Manton, J. H. Riemannian Gaussian distributions on the space of symmetric positive definite matrices. *IEEE Transactions on Information Theory*, 63(4):2153–2170, 2017. doi: 10.1109/TIT.2017.2653803.

Sala, F., De Sa, C., Gu, A., and Re, C. Representation tradeoffs for hyperbolic embeddings. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4460–4469, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/sala18a.html.

Sang, E. F. T. K. and De Meulder, F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pp. 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1119176.1119195. URL https://doi.org/10.3115/1119176.1119195.

Sanh, V., Wolf, T., and Ruder, S. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *AAAI*, volume abs/1811.06031, 2019. URL http://arxiv.org/abs/1811.06031.

Seth, Y. Introduction to question answering over knowledge graphs, Oct 2019. URL https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/.

Shen, Z. Riemann-Finsler geometry with applications to information geometry. *Chinese Annals of Mathematics, Series B*, 27:73–94, 08 2006. doi: 10.1007/s11401-005-0333-3.

Shimaoka, S., Stenetorp, P., Inui, K., and Riedel, S. An attentive neural architecture for fine-grained entity type classification. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pp. 69–74, San Diego, CA, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-1313. URL https://www.aclweb.org/anthology/W16-1313.

Shimaoka, S., Stenetorp, P., Inui, K., and Riedel, S. Neural architectures for fine-grained entity type classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1271–1280, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1119.

Shimizu, R., Mukuta, Y., and Harada, T. Hyperbolic neural networks++. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ec85b0tUwbA.

Siegel, C. L. Symplectic geometry. *American Journal of Mathematics*, 65(1):1–86, 1943. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2371774.

Simoncelli, E. P. and Olshausen, B. Natural image statistics and neural representation. *Annual Review of Neuroscience*, 24:1193–1216, 2001.

Skopek, O., Ganea, O.-E., and Becigneul, G. Mixed-curvature variational autoencoders. In *8th International Conference on Learning Representations (ICLR)*, April 2020. URL https://openreview.net/pdf?id=S1g6xeSKDS.

Sra, S. A new metric on the manifold of kernel matrices with application to matrix geometric means. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/98dce83da57b0395e163467c9dae521b-Paper.pdf.

Sra, S. Positive definite matrices and the S-divergence. *Proceedings of the American Mathematical Society*, 2015. doi: 10.1090/proc/12953. Published electronically: October 22, 2015.

Srinivasan, B. and Ribeiro, B. On the equivalence between positional node embeddings and structural graph representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxzFySKwH.

Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pp. 697–706, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-654-7. doi: 10.1145/1242572.1242667. URL http://doi.acm.org/10.1145/1242572.1242667.

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkgEQnRqYQ.

Takagi, T. On an algebraic problem related to an analytic theorem of carathéodory and fejér and on an allied theorem of Landau. *Japanese journal of mathematics :transactions and abstracts*, 1:83–93, 1924. doi: 10.4099/jjm1924.1.0_83.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pp. 1067–1077, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee. ISBN 9781450334693. doi: 10.1145/2736277.2741093. URL https://doi.org/10.1145/2736277.2741093.

Tay, Y., Tuan, L. A., and Hui, S. C. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pp. 583–591, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5581-0. doi: 10.1145/3159652.3159664. URL http://doi.acm.org/10.1145/3159652.3159664.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319, 2000.

Tifrea, A., Becigneul, G., and Ganea, O.-E. Poincare glove: Hyperbolic word embeddings. In *7th International Conference on Learning Representations, ICLR*, New Orleans, LA, USA, May 2019. URL https://openreview.net/forum?id=Ske5r3AqK7.

Tosato, D., Farenzena, M., Spera, M., Murino, V., and Cristani, M. Multi-class classification on Riemannian manifolds for video surveillance. In Daniilidis, K., Maragos, P., and Paragios, N. (eds.), *Computer Vision - ECCV 2010, 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part II*, volume 6312 of *Lecture Notes in Computer Science*, pp. 378–391. Springer, 2010. doi: 10.1007/978-3-642-15552-9_28. URL https://doi.org/10.1007/978-3-642-15552-9_28.

Toutanova, K. and Chen, D. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4007. URL https://www.aclweb.org/anthology/W15-4007.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pp. 2071–2080. JMLR.org, 2016.

Tuzel, O., Porikli, F., and Meer, P. Region covariance: A fast descriptor for detection and classification. In Leonardis, A., Bischof, H., and Pinz, A. (eds.), *Computer Vision – ECCV 2006*, pp. 589–600, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

Tuzel, O., Porikli, F., and Meer, P. Pedestrian detection via classification on Riemannian manifolds. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1713–1727, 2008. URL http://dblp.uni-trier.de/db/journals/pami/pami30.html#TuzelPM08.

Ungar, A. Gyrovector spaces and their differential geometry. *Nonlinear Functional Analysis and Applications*, 10, 01 2005.

Ungar, A. A. *A Gyrovector Space Approach to Hyperbolic Geometry*. Morgan & Claypool, 2008a.

Ungar, A. A. *Analytic Hyperbolic Geometry and Albert Einstein's Special Theory of Relativity*. World Scientific, 2008b.

Ungar, A. A. *Barycentric Calculus in Euclidean and Hyperbolic Geometry: A Comparative Introduction*. World Scientific, 2010.

Ungar, A. A. *Beyond Pseudo-Rotations in Pseudo-Euclidean Spaces*. Mathematical Analysis and its Applications. Academic Press, 2018.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Vemulapalli, R. and Jacobs, D. Riemannian metric learning for symmetric positive definite matrices. *ArXiv*, abs/1501.02393, 2015.

Verbeek, K. and Suri, S. Metric embedding, hyperbolic space, and social networks. In *Proceedings of the Thirtieth Annual Symposium on Computational Geometry*, SOCG'14, pp. 501–510, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325943. doi: 10.1145/2582112.2582139. URL https://doi.org/10.1145/2582112.2582139.

Verbeek, K. and Suri, S. Metric embedding, hyperbolic space, and social networks. *Computational Geometry*, 59:1 – 12, 2016. ISSN 0925-7721. doi: https://doi.org/10.1016/j.comgeo.2016.08.003. URL http://www.sciencedirect.com/science/article/pii/S0925772116300712.

Vermeer, J. A geometric interpretation of Ungar's addition and of gyration in the hyperbolic plane. *Topology and Its Applications: a journal devoted to general, geometric, set-theoretic and algebraic topology*, 152(3):226–242, 2005. ISSN 0166-8641. doi: doi: 10.1016/j.topol.2004.10.012.

Veselkov, K., Gonzalez, G., Aljifri, S., Galea, D., Mirnezami, R., Youssef, J., Bronstein, M., and Laponogov, I. Hyperfoods: Machine intelligent mapping of cancer-beating molecules in foods. *Sci Rep*, 3(9), 2019. doi: 10.1038/s41598-019-45349-y.

Vinh Tran, L., Tay, Y., Zhang, S., Cong, G., and Li, X. HyperML: A boosting metric learning approach in hyperbolic space for recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, pp. 609–617, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368223. doi: 10.1145/3336191.3371850. URL https://doi.org/10.1145/3336191.3371850.

Wang, C., Guo, Y., and Song, X. Head pose estimation via manifold learning. In Bracken, P. (ed.), *Manifolds*, chapter 6. IntechOpen, Rijeka, 2017. doi: 10.5772/65903. URL `https://doi.org/10.5772/65903`.

Wang, M., Qiu, L., and Wang, X. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3), 2021. ISSN 2073-8994. doi: 10.3390/sym13030485. URL `https://www.mdpi.com/2073-8994/13/3/485`.

Wang, W., Wang, R., Huang, Z., Shan, S., and Chen, X. Discriminant analysis on Riemannian manifold of Gaussian distributions for face recognition with image sets. *IEEE Transactions on Image Processing*, 27(1):151–163, 2018. doi: 10.1109/TIP.2017.2746993.

Wilson, R. C., Hancock, E. R., Pekalska, E., and Duin, R. P. W. Spherical and hyperbolic embeddings of data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2255–2269, 2014.

Wu, Y., Jia, Y., Li, P., Zhang, J., and Yuan, J. Manifold kernel sparse representation of symmetric positive-definite matrices and its applications. *IEEE Transactions on Image Processing*, 24(11):3729–3741, 2015. doi: 10.1109/TIP.2015.2451953.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019. URL `http://arxiv.org/abs/1901.00596`.

Xin, J., Lin, Y., Liu, Z., and Sun, M. Improving neural fine-grained entity typing with knowledge attention. In *AAAI Conference on Artificial Intelligence*, 2018. URL `https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16321`.

Xiong, W., Wu, J., Lei, D., Yu, M., Chang, S., Guo, X., and Wang, W. Y. Imposing label-relational inductive bias for extremely fine-grained entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 773–784, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1084. URL `https://www.aclweb.org/anthology/N19-1084`.

Xu, J. and Durrett, G. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Xu, M. Understanding graph embedding methods and their applications. *CoRR*, abs/2012.08019, 2020. URL `https://arxiv.org/abs/2012.08019`.

Xu, P. and Barbosa, D. Neural fine-grained entity type classification with hierarchy-aware loss. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 16–25, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1002. URL `https://www.aclweb.org/anthology/N18-1002`.

Yaghoobzadeh, Y., Adel, H., and Schütze, H. Noise mitigation for neural entity typing and relation extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 1183–1194, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/E17-1111`.

Yang, T., Sha, L., and Hong, P. *NagE: Non-Abelian Group Embedding for Knowledge Graphs*, pp. 1735–1742. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450368599. URL `https://doi.org/10.1145/3340531.3411875`.

Yavuz, S., Gur, I., Su, Y., Srivatsa, M., and Yan, X. Improving semantic parsing via answer type inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 149–159, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1015. URL `https://www.aclweb.org/anthology/D16-1015`.

Yin, M., Guo, Y., Gao, J., He, Z., and Xie, S. Kernel sparse subspace clustering on symmetric positive definite manifolds. In *CVPR*, pp. 5157–5164. IEEE Computer Society, 2016. ISBN 978-1-4673-8851-1. URL `http://dblp.uni-trier.de/db/conf/cvpr/cvpr2016.html#YinGGHX16`.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pp. 974–983, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355520. doi: 10.1145/3219819.3219890. URL `https://doi.org/10.1145/3219819.3219890`.

Yogatama, D., Gillick, D., and Lazic, N. Embedding methods for fine grained entity type classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 291–296, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2048. URL `https://www.aclweb.org/anthology/P15-2048`.

Yosef, M. A., Bauer, S., Hoffart, J., Spaniol, M., and Weikum, G. HYENA: Hierarchical type classification for entity names. In *Proceedings of COLING 2012: Posters*, pp. 1361–1370, Mumbai, India, December 2012. URL https://www.aclweb.org/anthology/C12-2133.

Zhang, F., Yuan, N. J., Lian, D., Xie, X., and Ma, W.-Y. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 353–362, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939673. URL https://doi.org/10.1145/2939672.2939673.

Zhang, S., Tay, Y., Yao, L., and Liu, Q. Quaternion knowledge graph embeddings. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 2735–2745. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/d961e9f236177d65d21100592edb0769-Paper.pdf.

Zhang, T., Zheng, W., Cui, Z., and Li, C. Deep manifold-to-manifold transforming network. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 4098–4102, 2018a. doi: 10.1109/ICIP.2018.8451626.

Zhang, Y., Ai, Q., Chen, X., and Wang, P. Learning over knowledge-base embeddings for recommendation. *CoRR*, abs/1803.06540, 2018b. URL http://arxiv.org/abs/1803.06540.

Zhou, B., Khashabi, D., Tsai, C.-T., and Roth, D. Zero-shot open entity typing as type-compatible grounding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2065–2076, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1231.

Zhou, J., Xu, Z., Rush, A. M., and Yu, M. Automating botnet detection with graph neural networks. *AutoML for Networking and Systems Workshop of MLSys 2020 Conference*, 2020.

Zitnik, M., Agrawal, M., and Leskovec, J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinform.*, 34(13):i457–i466, 2018. doi: 10.1093/bioinformatics/bty294. URL https://doi.org/10.1093/bioinformatics/bty294.