ON THE GEOMETRY OF DATA REPRESENTATIONS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES OF ETH ZÜRICH
(Dr. sc. ETH Zürich)

presented by

GARY BÉCIGNEUL

Master in Mathematics
University of Cambridge

born on 11.11.1994
citizen of France

Accepted on the recommendation of

Prof. Dr. Thomas Hofmann (ETH Zürich)
Prof. Dr. Gunnar Rätsch (ETH Zürich)
Prof. Dr. Tommi Jaakkola (MIT)
Prof. Dr. Regina Barzilay (MIT)

2020
The vast majority of state-of-the-art Machine Learning (ML) methods nowadays internally represent the input data as being embedded in a “continuous” space, i.e. as sequences of floats, where nearness in this space is meant to define semantic or statistical similarity w.r.t to the task at hand. As a consequence, the choice of which metric is used to measure nearness, as well as the way data is embedded in this space, currently constitute some of the cornerstones of building meaningful data representations.

Characterizing which points should be close to each other in a given set essentially defines a “geometry”. Interestingly, certain geometric properties may be incompatible in a non-trivial manner — or put another way, selecting desired geometric properties may have non-trivial implications. The investigation of which geometric properties are desirable in a given context, and how to enforce them, is one of the motivations underlying this thesis.

Initially motivated by uncovering how Convolutional Neural Network (CNN) disentangle representations to make tangled classes linearly separable, we start our investigation by studying how invariance to nuisance deformations may help untangling classes in a classification problem, by locally contracting and flattening group orbits within data classes.

We then take interest into directly representing data in a Riemannian space of choice, with a particular emphasis on hyperbolic spaces, which is known to be better suited to represented tree-like graphs. We develop a word embedding method generalizing GloVe, as well as a new way of measuring entailment between concepts. In the hyperbolic space, we also develop the tools needed to build neural networks, including matrix multiplication, pointwise non-linearity, Multinomial Logistic Regression (MLR) and Gated Recurrent Unit (GRU).

Since many more optimization tools are available for Euclidean domains than for the hyperbolic space, we needed to adapt some of the most powerful adaptive schemes — Adagrad, Adam, Amsgrad — to such spaces, in order to let our hyperbolic models have a chance to outperform their Euclidean counterparts. We also provide convergence
guarantees for these new methods, which recover those already known for the particular case of the Euclidean space.

Finally, the growing prominence of graph-like data led us to extend some of the most successful Graph Neural Network (GNN) architectures. First, we start by generalizing Graph Convolutional Network (GCN)s to hyperbolic and spherical spaces. We then leveraged Optimal Transport (OT) geometry to turn current architectures into a universal approximator, by dispensing with the last node aggregation step yielding the final graph embedding.

We hope that this work will help motivate further investigations in these geometrically-flavored directions.
RÉSUMÉ

La grande majorité des méthodes d’Apprentissage Automatique (AA) atteignant l’état-de-l’art de nos jours représente les données reçues en entrée dans un espace “continu”, i.e. comme une séquence de nombres décimaux, dans un espace où la proximité est vouée à définir la similarité sémantique ou statistique par rapport à la tâche. En conséquence, le choix de la métrique utilisée pour mesurer la proximité, ainsi que la façon dont les données sont plongées dans cet espace, constituent actuellement l’une des pierres angulaires de l’édification de bonnes représentations des données.

Charactériser quels points doivent être proches les uns des autres dans un ensemble définit essentiellement ce qu’on appelle une géométrie. Il se trouve que certaines propriétés géométriques peuvent être incompatibles d’une façon non-triviale – ou, dit autrement, la sélection de certaines propriétés géométriques peut avoir des implications non-triviales. La recherche de quelles propriétés géométriques sont souhaitables dans un contexte donné, et de comment les obtenir, est l’une des motivations principales sous-jacentes à cette thèse.

Initiallement motivé par la découverte de comment les réseaux de neurones à convolutions parviennent à désenchevêtrer les données afin de rendre des classes enchevêtrées lignearement séparables, nous commençons notre recherche en étudiant comment l’invariance à des transformations nuisibles peut aider à désenchevêtrer les classes d’un problème de classification, en contractant et en aplatisant localement des orbites de groupes au sein des classes de données.

Nous nous intéressons ensuite à représenter directement les données dans un espace Riemannien de choix, avec une emphase particulière sur les espaces hyperboliques, qui sont connus pour être mieux adaptés à représenter les graphes ayant une structure arborescente sous-jacente. Nous développons une méthode de word embedding généralisant GloVe, ainsi qu’une nouvelle façon de mesurer les relations asymétriques d’inclusion sémantique entre les conceptes. Nous développons également les outils nécessaires à la construction de réseaux de neurones dans les espaces hyperboliques: multiplication matricielle, application de non-linéarité ponctuelle, régression multi-logistique et GRU.
Comme il existe bien davantage d’outils d’optimisation pour les espaces Euclidiens que pour les espaces hyperboliques, nous avions besoin d’adapter certaines des méthodes adaptives les plus performantes — Adagrad, Adam, Amsgad — à ces espaces, afin de permettre à nos modèles hyperboliques d’avoir une chance de performer supérieurement à leur analogue Euclidien. Nous prouvons ainsi des garanties de convergence pour ces nouvelles méthodes, qui recouvrent celles déjà connues pour le cas particulier de l’espace Euclidien.

Enfin, la présence accrue de données sous forme de graphe nous a conduit à étendre certaines des architectures de réseaux de neurones de graphes les plus puissantes. En premier lieu, nous commençons par généraliser les réseaux de neurones de graphes à convolutions, aux espaces hyperboliques et sphériques. Ensuite, nous faisons appel à la géométrie du transport optimal pour transformer les architectures courantes en approximateur universel, en supprimant la dernière agrégation des représentations internes des noeuds du graphe qui avant résultait en la représentation finale du graphe.

Nous espérons que ceci contribuera à motiver davantage d’explorations dans ces directions de recherche à tendance géométrique.
The material presented in this thesis has in parts been published in the following publications:


In part currently under review:


And in part unpublished:


1Equal contribution.
The following publications were part of my PhD research and present results that are supplemental to this work or build upon its results, but are not covered in this dissertation:


   ICLR 2020: International Conference on Learning Representations. [SGB20].

    AISTATS 2020: International Conference on Artificial Intelligence and Statistics.”
    [Ali+20b].

    AAAI 2021: Association for the Advancement of Artificial Intelligence.
    [CBG21].

Currently under review are also other parts of my PhD research and present results that are supplemental to this work or build upon its results, but are not covered in this dissertation:

12. Louis Abraham, Gary Bécigneul, Benjamin Coleman, Anshumali Shrivastava, Bernhard Schölkopf and Alexander J Smola.
“Bloom Origami Assays: Practical Group Testing”
*Under Review at AISTATS 2021.*
[Abr+20].

“Practical Accelerated Optimization on Riemannian Manifolds.”
*Under Review at AISTATS 2021.*
[Ali+20a].

Finally, the following pieces of work were also part of my PhD research but remained unpublished:

“Learning a Riemannian Metric with an Energy-based GAN.”
*Unpublished.*
[BKH18].

15. Yannic Kilcher \(^1\), Gary Bécigneul \(^1\), Thomas Hofmann.
“Escaping Flat Areas via Function Preserving Structural Network Modifications.”
[KBH18].

16. Yannic Kilcher \(^1\), Gary Bécigneul \(^1\), Thomas Hofmann.
“Parametrizing Filters of a CNN with a GAN.”
[KBH17].
ACKNOWLEDGMENTS

First, I thank my professor Thomas Hofmann for the continuous support during my PhD studies at ETH. His trust in my success and the provided scientific freedom and guidance were decisive to the result of this thesis. Next, I would also like to express my gratitude to Octavian-Eugen Ganea, with whom I brainstormed almost daily.

I am also very grateful to Tommi Jaakkola and Regina Barzilay who warmly welcomed me in their lab to work with them at MIT, showed me new problems and inspired me, and to Benson Chen who also worked with us.

Many thanks to Bernhard Schölkopf for thinking about me to help design a group testing method and for his close collaboration; to Louis Abraham for solving non-trivial problems on a short notice, and Ben Coleman, Anshumali Shrivastava and Alexander Smola in this team work.

I would also like to acknowledge my co-workers during this PhD: Alexandru Tifrea, Gregor Bachmann, Foivos Alimisis, Antonio Orvieto, Aurelien Lucchi, Andreas Bloch, Calin Cruceru, Ondrej Skopek, Sylvain Gelly, Yannic Kilcher, Octav Dragoi, Philipp Wirth, Panayiotis Panayiotou, Charles Ladari, Severin Bahman, Jovan Andonov, Igor Petrovski, Gokul Santhanam, Yashas Annadani, Aliaksei Severyn, and all my Data Analytics Lab, CSAIL and MPI colleagues. Thank you!

I am especially grateful to Grégoire Fanneau who gave birth to my interest for mathematics in high school, to Prof. Jean-Claude Jacquens and to Prof. Luc Albert, who taught me mathematics and computer science and challenged me in French preparatory classes. Also, many thanks to Simon Janin for bringing light to the blindspots present in my workflow and for coaching me over the years. Thanks to Rémi Saidi for increasing my standards of mathematical rigor. Thanks to Victor Perez, Nicole and Dominique Secret for believing in me and supporting my undergraduate studies.

Last but not least, all my love to my mother for her encouragement, trust and support from day one.

I dedicate all my work to my little brother, Tom Bécigneul, who left us too early, but whose joy of life continues to inspire me.
# CONTENTS

## ACRONYMS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 INTRODUCTION, MOTIVATIONS &amp; GOALS</td>
<td>3</td>
</tr>
<tr>
<td>1.1 A Primer on Machine Learning</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Euclidean Representations</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Power &amp; Ubiquity</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Limitations</td>
<td>4</td>
</tr>
<tr>
<td>1.3 Beyond Euclidean Geometry</td>
<td>6</td>
</tr>
<tr>
<td>1.3.1 Characterizing the Geometry of Discrete Data</td>
<td>6</td>
</tr>
<tr>
<td>1.3.2 Non-Euclidean Properties of Data Embedded in a Euclidean Space</td>
<td>7</td>
</tr>
<tr>
<td>1.3.3 Hyperbolic Representations</td>
<td>8</td>
</tr>
<tr>
<td>1.3.4 Riemannian Optimization</td>
<td>9</td>
</tr>
<tr>
<td>1.3.5 Graph-like Data</td>
<td>10</td>
</tr>
<tr>
<td>1.4 Thesis Contributions</td>
<td>13</td>
</tr>
<tr>
<td>2 GROUP INVARIANCE &amp; CURVATURE OF REPRESENTATIONS</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Background Material</td>
<td>16</td>
</tr>
<tr>
<td>2.1.1 Groups, Lie Algebras &amp; Haar Measures</td>
<td>16</td>
</tr>
<tr>
<td>2.1.2 $I$-theory</td>
<td>18</td>
</tr>
<tr>
<td>2.2 Main Results: Formal Framework and Theorems</td>
<td>21</td>
</tr>
<tr>
<td>2.2.1 Group Pooling Results in Orbit Contraction</td>
<td>21</td>
</tr>
<tr>
<td>2.2.2 Group Pooling Results in Orbit Flattening</td>
<td>25</td>
</tr>
<tr>
<td>2.2.3 Orbit Flattening of $\mathbb{R}^2 \ltimes SO(2)$</td>
<td>30</td>
</tr>
<tr>
<td>2.3 Numerical Experiments: Shears and Translations on Images</td>
<td>32</td>
</tr>
<tr>
<td>2.3.1 Synthetic Orbit Contraction in Pixel Space</td>
<td>32</td>
</tr>
<tr>
<td>2.3.2 CIFAR-10: Orbit Contraction in Logit Space</td>
<td>36</td>
</tr>
<tr>
<td>2.4 Conclusive Summary</td>
<td>36</td>
</tr>
<tr>
<td>2.5 Reminders of Known Results</td>
<td>37</td>
</tr>
<tr>
<td>3 HYPERBOLIC SPACES: EMBEDDINGS &amp; CLASSIFICATION</td>
<td>39</td>
</tr>
<tr>
<td>3.1 Mathematical Preliminaries</td>
<td>40</td>
</tr>
<tr>
<td>3.1.1 Reminders of Differential Geometry</td>
<td>40</td>
</tr>
<tr>
<td>3.1.2 A Brief Introduction to Hyperbolic Geometry</td>
<td>41</td>
</tr>
<tr>
<td>3.1.3 Riemannian Geometry of Gyrovector Spaces</td>
<td>43</td>
</tr>
<tr>
<td>3.2 Word Embeddings</td>
<td>50</td>
</tr>
<tr>
<td>3.2.1 Adapting GLOVe to Metric Spaces</td>
<td>50</td>
</tr>
</tbody>
</table>
## 3.2 Fisher Information and Hyperbolic Products

### 3.2.2 Training Details

### 3.2.3 Similarity & Analogy

### 3.2.5 Hypervnymy: A New Way of Assessing Entailment

### 3.2.6 Motivating Hyperbolic GloVe via $\delta$-hyperbolicity

## 3.3 Classification & Regression

### 3.3.1 Matrix Multiplications & Pointwise non-Linearities

### 3.3.2 Recurrent Networks & Gated Recurrent Units

### 3.3.3 Multinomial Logistic Regression

## 3.4 Additional Related Work

## 3.5 Conclusive Summary

## 4 Adaptive Optimization in Non-Euclidean Geometry

### 4.1 Preliminaries and Notations

#### 4.1.1 Elements of Geometry

#### 4.1.2 Riemannian Optimization

#### 4.1.3 Amsgrad, Adam, Adagrad

### 4.2 Adaptive Schemes in Riemannian Manifolds

#### 4.2.1 Obstacles in the General Setting

#### 4.2.2 Adaptivity Across Manifolds in a Cartesian Product

### 4.3 Algorithms & Convergence Guarantees

#### 4.3.1 Rmsgrad

#### 4.3.2 RadamNc

#### 4.3.3 Intuition Behind the Proofs

### 4.4 Empirical Validation

### 4.5 Additional Related Work

### 4.6 Conclusive Summary

### 4.7 Reminders of Knowns Results

## 5 Non-Euclidean Geometry for Graph-like Data

### 5.1 Background & Mathematical Preliminaries

#### 5.1.1 The Geometry of Spaces of Constant Curvature

#### 5.1.2 Optimal Transport Geometry

#### 5.1.3 GCN: Graph Convolutional Networks

#### 5.1.4 Directed Message Passing Neural Networks

### 5.2 $\kappa$-GCN: Constant Curvature GCN

#### 5.2.1 Tools for a $\kappa$-GCN

#### 5.2.2 $\kappa$-Right-Matrix-Multiplication

#### 5.2.3 $\kappa$-Left-Matrix-Multiplication: Midpoint Extension

#### 5.2.4 Logits

#### 5.2.5 Final Architecture of the $\kappa$-GCN
Concerning the drawing on the left-hand side, the blue and red areas correspond to the compact neighborhood \( G_0 \) centered in \( f \) and \( L_g f \) respectively, the grey area represents only a visible subpart of the whole group orbit, the thick, curved line is a geodesic between \( f \) and \( L_g f \) inside the orbit \( G \cdot f \), and the dotted line represents the line segment between \( f \) and \( L_g f \) in \( L^2(\mathbb{R}^2) \), whose size is given by the euclidean distance \( \| L_g f - f \|_2 \). See Figures 2.2 and 2.3.

Illustration of orbits (first part).

Illustration of orbits (second part).

Translation orbits in the logit space of a CNN (CIFAR-10).

Isometric deformation \( \varphi \) of \( \mathbb{D}^2 \) (left end) into \( \mathbb{H}^2 \) (right end).

We show here one of the \( \mathbb{D}^2 \) spaces of 20D word embeddings embedded in \( (\mathbb{D}^2)^{10} \) with our unsupervised hyperbolic Glove algorithm. This illustrates the three steps of applying the isometry. From left to right: the trained embeddings, raw; then after centering; then after rotation; finally after isometrically mapping them to \( \mathbb{H}^2 \). The isometry was obtained with the weakly-supervised method WordNet 400 + 400. Legend: WordNet levels (root is 0). Model: \( h = (\cdot)^2 \), full vocabulary of 190k words. More of these plots for other \( \mathbb{D}^2 \) spaces are shown in [TBG19].
Figure 3.3 Different hierarchies captured by a 10x2D model with $h(x) = x^2$, in some selected 2D half-planes. The $y$ coordinate encodes the magnitude of the variance of the corresponding Gaussian embeddings, representing word generality/specificity. Thus, this type of Nx2D models offer an amount of interpretability. ........................................ 62

Figure 3.4 Left: Gaussian variances of our learned hyperbolic embeddings (trained unsupervised on co-occurrence statistics, isometry found with “Unsupervised 1k+1k”) correlate with WordNet levels; Right: performance of our embeddings on hypernymy (HyperLex dataset) evolve when we increase the amount of supervision $x$ used to find the correct isometry in the model WN $x + x$. As can be seen, a very small amount of supervision (e.g. 20 words from WordNet) can significantly boost performance compared to fully unsupervised methods. ........................................ 63

Figure 3.5 Test accuracies for various models and four datasets. “Eucl” denotes Euclidean, “Hyp” denotes hyperbolic. All word and sentence embeddings have dimension 5. We highlight in bold the best method (or methods, if the difference is less than 0.5%). Implementation of these experiments was done by co-authors in [GBH18c]. .......................... 77

Figure 3.6 PREFIX-30% accuracy and first (premise) sentence norm plots for different runs of the same architecture: hyperbolic GRU followed by hyperbolic FFNN and hyperbolic/Euclidean (half-half) MLR. The X axis shows millions of training examples processed. Implementation of these experiments was done by co-authors in [GBH18c]. 78
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>PREFIX-30% accuracy and first (premise) sentence norm plots for different runs of the same architecture: Euclidean GRU followed by Euclidean FFNN and Euclidean MLR. The X axis shows millions of training examples processed. Implementation of these experiments was done by co-authors in [GBH18c].</td>
</tr>
<tr>
<td>3.8</td>
<td>PREFIX-30% accuracy and first (premise) sentence norm plots for different runs of the same architecture: hyperbolic RNN followed by hyperbolic FFNN and hyperbolic MLR. The X axis shows millions of training examples processed. Implementation of these experiments was done by co-authors in [GBH18c].</td>
</tr>
<tr>
<td>3.9</td>
<td>An example of a hyperbolic hyperplane in $\mathbb{D}^3_1$ plotted using sampling. The red point is $\mathbf{p}$. The shown normal axis to the hyperplane through $\mathbf{p}$ is parallel to $\mathbf{a}$.</td>
</tr>
<tr>
<td>3.10</td>
<td>Hyperbolic (left) vs Direct Euclidean (right) binary MLR used to classify nodes as being part in the GROUP.N.01 subtree of the WordNet noun hierarchy solely based on their Poincaré embeddings. The positive points (from the subtree) are in red, the negative points (the rest) are in yellow and the trained positive separation hyperplane is depicted in green.</td>
</tr>
<tr>
<td>3.11</td>
<td>Test F1 classification scores (%) for four different subtrees of WordNet noun tree. 95% confidence intervals for 3 different runs are shown for each method and each dimension. “Hyp” denotes our hyperbolic MLR, “Eucl” denotes directly applying Euclidean MLR to hyperbolic embeddings in their Euclidean parametrization, and $\log_0$ denotes applying Euclidean MLR in the tangent space at $0$, after projecting all hyperbolic embeddings there with $\log_0$. Implementation of these experiments was done by co-authors in [GBH18c].</td>
</tr>
<tr>
<td>4.1</td>
<td>Comparison of the Riemannian and Euclidean versions of AMSGRAD.</td>
</tr>
</tbody>
</table>
Figure 4.2 Results for methods doing updates with the exponential map. From left to right we report: training loss, MAP on the train set, MAP on the validation set. ................................. 109

Figure 4.3 Results for methods doing updates with the retraction. From left to right we report: training loss, MAP on the train set, MAP on the validation set. ................................. 110

Figure 5.1 Geodesics in the Poincaré disk (left) and the stereographic projection of the sphere (right). ................................. 115

Figure 5.2 Heatmap of the distance function $d_\kappa(x, \cdot)$ in $\text{st}_\kappa^2$ for $\kappa = -0.254$ (left) and $\kappa = 0.248$ (right). ................................. 116

Figure 5.3 We illustrate, for a given 2D point cloud, the optimal transport plan obtained from minimizing the Wasserstein costs; $c(\cdot, \cdot)$ denotes the Euclidean distance (top) or squared difference (bottom). $A, B$ are the Euclidean distance matrices obtained from point clouds $X, Y$. A higher dotted-line thickness illustrates a greater mass transport. ................................. 122

Figure 5.4 Left: Spherical gyromidpoint of four points. Right: Möbius gyromidpoint in the Poincaré model defined by [Ung08] and alternatively, here in eq. (5.27). ................................. 126

Figure 5.5 Weighted Euclidean midpoint $\alpha x + \beta y$ ................................. 128

Figure 5.6 Heatmap of the distance from a $\text{st}_\kappa^2$-hyperplane to $x \in \text{st}_\kappa^2$ for $\kappa = -0.254$ (left) and $\kappa = 0.248$ (right) ................................. 128

Figure 5.7 Histogram of Curvatures from “Deviation of Parallogram Law” ................................. 137

Figure 5.8 Intuition for our Wasserstein prototype model. We assume that a few prototypes, e.g. some functional groups, highlight key facets or structural features of graphs in a particular graph classification/regression task at hand. We then express graphs by relating them to these abstract prototypes represented as free point cloud parameters. Note that we do not learn the graph structure of the prototypes. ................................. 140
Figure 5.9  2D embeddings of prototypes and of real molecule samples with (right, with weight 0.1) and without (left) contrastive regularization for runs using the exact same random seed. Points in the prototypes tend to cluster and collapse more when no regularization is used, implying that the optimal transport plan is no longer uniquely discriminative. Prototype 1 (red) is enlarged for clarity: without regularization (left), it is clumped together, but with regularization (right), it is distributed across the space. .......................... 142

Figure 5.10  Left: GNN model; Right: ProtoW-L2 model; Comparison of the correlation between graph embedding distances (X axis) and label distances (Y axis) on the ESOL dataset. .......................... 156

Figure 5.11  2D heatmaps of T-SNE [MH08] projections of molecular embeddings (before the last linear layer) w.r.t. their associated predicted labels. Heat colors are interpolations based only on the test molecules from each dataset. Comparing (a) vs (b) and (c) vs (d), we can observe a smoother space of our model compared to the GNN baseline as explained in the main text. .......................... 157

LIST OF TABLES

Table 3.1  Word similarity results for 100-dimensional models. Highlighted: the best and the 2nd best. Implementation of these experiments was done in collaboration with co-authors in [TBG19]. .... 54

Table 3.2  Nearest neighbors (in terms of Poincaré distance) for some words using our 100D hyperbolic embedding model. .......................... 55
Table 3.3 Word analogy results for 100-dimensional models on the Google and MSR datasets. Highlighted: the best and the 2nd best. Implementation of these experiments was done in collaboration with co-authors in [TBG19]. 57

Table 3.4 Some words selected from the 100 nearest neighbors and ordered according to the hypernymy score function for a 50x2D hyperbolic embedding model using \( h(x) = x^2 \) 62

Table 3.5 WBLESS results in terms of accuracy for 3 different model types ordered according to their difficulty. 64

Table 3.6 Hyperlex results in terms of Spearman correlation for 3 different model types ordered according to their difficulty. 64

Table 3.7 Average distances, \( \delta \)-hyperbolicities and ratios computed via sampling for the metrics induced by different \( h \) functions, as defined in eq. (3.51). 67

Table 3.8 Similarity results on the unrestricted (190k) vocabulary for various \( h \) functions. This table should be read together with Table 3.7. 67

Table 3.9 Various properties of similarity benchmark datasets. The frequency index indicates the rank of a word in the vocabulary in terms of its frequency: a low index describes a frequent word. The median of indexes seems to best discriminate WordSim from SimLex and SimVerb. 68

Table 5.1 Minimum achieved average distortion of the different models. \( H \) and \( S \) denote hyperbolic and spherical models respectively. 135

Table 5.2 Node classification: Average accuracy across 5 splits with estimated uncertainties at 95 percent confidence level via bootstrapping on our datasets. \( H \) and \( S \) denote hyperbolic and spherical models respectively. Implementation of these experiments was done in collaboration with co-authors in [BBG20]. 136
Table 5.3  
Average curvature obtained for node classification. \( H \) and \( S \) denote hyperbolic and spherical models respectively. Curvature for Pubmed was fixed for the product model. Implementation of these experiments was done in collaboration with co-authors in \[BBG20\].

Table 5.4  
Results of different models on the property prediction datasets. **Best** in bold, _second_ best underlined. Proto methods are ours. Lower RMSE is better, while higher AUC is better. The prototype-based models generally outperform the GNN, and the Wasserstein models perform better than the model using only simple Euclidean distances, suggesting that the Wasserstein distance provides more powerful representations. Wasserstein models trained with contrastive regularization as described in section 5.4.2 outperform those without. Implementation of these experiments was done in collaboration with co-authors in \[Béc+20\].

Table 5.5  
The Spearman and Pearson correlation coefficients on the ESOL dataset for the GNN and ProtoW-L2 model w.r.t. the pairwise difference in embedding vectors and labels. Implementation of these experiments was done in collaboration with co-authors in \[Béc+20\].

Table 5.6  
The parameters for our models (the prototype models all use the same GNN base model), and the values that we used for hyperparameter search. When there is only a single value in the search list, it means we did not search over this value, and used the specified value for all models.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>AA</td>
<td>Apprentissage Automatique</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>GNN</td>
<td>Graph Neural Network</td>
</tr>
<tr>
<td>GCN</td>
<td>Graph Convolutional Network</td>
</tr>
<tr>
<td>DMPNN</td>
<td>Directed Message Passing Neural Network</td>
</tr>
<tr>
<td>OT</td>
<td>Optimal Transport</td>
</tr>
<tr>
<td>OTGNN</td>
<td>Optimal Transport Graph Neural Network</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>RSGD</td>
<td>Riemannian Stochastic Gradient Descent</td>
</tr>
<tr>
<td>NCE</td>
<td>Noise Contrastive Estimation</td>
</tr>
<tr>
<td>FFNN</td>
<td>Feed-forward Neural Network</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>GRU</td>
<td>Gated Recurrent Unit</td>
</tr>
<tr>
<td>MLR</td>
<td>Multinomial Logistic Regression</td>
</tr>
<tr>
<td>MDS</td>
<td>Multidimensional Scaling</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
</tbody>
</table>
ACRONYMS

S
1.1 A PRIMER ON MACHINE LEARNING

In the last two decades, the combined increased availability of data on the one hand and computational power on the other hand has given rise to a pletora of new powerful statistical methods. *Machine Learning* grew wider as a field in opposite directions at the same time: highly engineered, “black box”, uninterpretable models with millions, sometimes billions of parameters partly took over the state-of-the-art spotlight [Bro+20]; [Dev+18]; [KSH12], while most new fundamental contributions were becoming more and more mathematical.

It is worth reminding that what most of these successful models happen to have in common is to either take as input data represented as a Euclidean embedding (be it images, words or waveforms) or to internally represent symbolic data as such. This tendency has naturally motivated the development of general-purpose representations to be used in downstream tasks, the most striking example being perhaps word embeddings with Word2Vec [Mik+13b], GloVe [PSM14] and FastText [Boj+16]. More broadly, finding “good” data representations and understanding what good means has quickly made itself one of the cornerstones of machine learning: Y. Bengio has even described the disentangling of the underlying factors explaining the data as “pre-solving any possible task relevant to the observed data” [Ben13], which has recently been emphasized when a step in this direction won the Best Paper Award at ICML 2019 [Loc+19].

Acknowledging the importance of adequately representing data, we decided to concentrate our efforts in this direction. The observation of how relatively unexplored were non-Euclidean geometries in the above mentionned pipelines, together with a strong taste for geometry, has led us through the story we tell in this dissertation.
INTRODUCTION, MOTIVATIONS & GOALS

1.2 EUCLIDEAN REPRESENTATIONS

1.2.1 Power & Ubiquity

As computers allow for significant amounts of operations on numbers, representing data as finite sequences or grids of numbers quickly appeared as a natural choice. When the necessity to measure similarity or discrepancy between data points came to light, one of the most canonical choices was to generalize to higher dimensions the way we naturally compute distances in our intuition-friendly, 3-dimensional physical world: via Euclidean Geometry. From a computational standpoint, this geometry has numerous advantages: (i) many important quantities such as distances, inner-products and geodesics can be computed very efficiently; (ii) linearity of the Euclidean space provides us with a tremendous amount of mathematical tools, e.g. Linear Algebra or Probability Theory; (iii) reasoning in this space matches in many ways our human intuition, compared to other, more exotic geometries.

Euclidean representations are at the heart of most state-of-the-art models available today, from language modeling [Dev+18]; [Rad+19] to image classification [He+16]; [KSH12], generation [Goo+14], machine translation [BCB14]; [Vas+17], speech recognition [GMH13] or recommender systems [He+17].

1.2.2 Limitations

Despite its success, as almost any particular model choice, Euclidean Geometry may have certain limitations. Or put another way: non-Euclidean geometries may constitute a superior choice in a given setting.

Indeed, recent research has proven that many types of complex data (e.g. graph data) from a multitude of fields (e.g. Biology, Network Science, Computer Graphics or Computer Vision) exhibit a highly non-Euclidean latent anatomy [Bro+17]. In such cases, where Euclidean properties such as shift-invariance are not desired, the Euclidean space does not provide the most powerful or meaningful geometrical representations.

Moreover, as canonical as Euclidean Geometry may seem, it actually constraints the geometric properties of certain elementary objects:

1. The sum of angles of a triangle is always $\pi$. 

1
2. The volume of a ball grows polynomially with its radius.

3. Euclidean inner-product defines a positive definite kernel:

\[ \forall \alpha \in \mathbb{R}^n \setminus \{0_n\}, \forall (x_i)_{1 \leq i \leq n} \in (\mathbb{R}^n)^n : \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle x_i, x_j \rangle > 0. \]  

(1.1)

which, as one can show (see section 5.4.5), can be reformulated equivalently as the fact that the squared Euclidean distance defines a conditionally negative definite kernel (c.n.d.):

\[ \forall \alpha \in \mathbb{R}^n, \forall (x_i)_{1 \leq i \leq n} \in (\mathbb{R}^n)^n : \text{if } \sum_{i=1}^{n} \alpha_i = 0, \text{ then } \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \|x_i - x_j\|^2 \leq 0. \]  

(1.2)

At this stage, one may wonder: to what extent are these properties restrictive?

Given a task to solve and a discrete/symbolic dataset, there often exists an inherent geometry underlying the task: be it semantic similarity in natural language processing or image classification, proximity between users and items in recommender systems, or more broadly: the geometry that characterizes nearness in the space of labels for any classification or regression problem.

From a given task and dataset, such geometries can be defined by ideal similarity or discrepancy metrics. But, what guarantees that these metrics will share the above listed properties of Euclidean Geometry? Indeed, there exist other embedding geometries possessing properties contradictory to those listed above. For instance, as we will see in greater details later on, the angles of a triangle sum below (resp. above) \( \pi \) in a hyperbolic (resp. spherical) space. Another example: in a b-ary tree, the number of \( k \)-hops neighbors to the root grows exponentially with \( k \); similarly, in a hyperbolic space, the volume of a ball grows exponentially with its radius, which is not the case for Euclidean and spherical spaces. Finally, a Gram matrix need not be positive definite nor c.n.d. to represent a valid similarity or discrepancy kernel.

This immediately raises the question of how can we assess in advance which geometry is best to embed a dataset, relative to a given task? We discuss this matter in the next section.
PREVIEW. In practice, additional considerations must be taken into account, than such a geometrical match on a mathematical level: (i) computational efficiency, of course, but also (ii) hardness of optimization. Indeed, the vast majority of embedding methods rely on gradient-based optimization. A non-Euclidean embedding metric could theoretically represent a better choice if one were able to find a good optimum, but may make this optimization problem much harder. Developing new non-Euclidean machine learning methods requires to carefully consider these matters.

1.3 BEYOND EUCLIDEAN GEOMETRY

1.3.1 Characterizing the Geometry of Discrete Data

As mentionned above, how can we know which embedding geometry would theoretically best suit a given dataset/task? Although this remains an open problem, certain quantities can be computed to get an idea.

METRIC DISTORSION. Given a weighted graph or a discrete metric space, how closely can we embed it into a “continuous” space, so that distances in the latter reflect those in the former? A natural measurement of this distorsion is given by taking the sum, over the finite metric space, of squared differences between the target metric $d_{ij}$ and corresponding embeddings $x_i, x_j$:

$$\min_{x} \sum_{i,j} (d(x_i, x_j) - d_{ij})^2. \quad (1.3)$$

Note that finding a family $x$ of embeddings minimizing distorsion can be intractable in practice. Empirical upper bounds can be obtained either via combinatorial or gradient-based methods (e.g. see [De +18a] in hyperbolic spaces), and for simple cases theoretical lower bounds can be computed by leveraging the metric properties (e.g. see Section 1.3.2 of [Gan19]).

$\delta$-HYPERBOLICITY. This notion, originally proposed by Gromov [Gro87], characterizes how tree-like a space is, by looking at how close is each geodesic segment of a triangle to its barycenter: in a tree, e.g. for the 3-star graph made of one root node and four leaf nodes, this
distance is zero; in a Euclidean space, this distance in unbounded; in a hyperbolic space, it is upper bounded by a factor going to zero when curvature goes to $-\infty$. As it is defined as a worst case measure over all triangles in a metric space, more recently Borassi et al. [BCC15] proposed an averaged version yielding more robust and efficiently computable statistics for finite graphs. We provide further definitions and explanations about original and averaged $\delta$-hyperbolicities, as well as numerical computations, in section 3.2.6.

**Discrete Ricci-Ollivier Curvature.** Originally introduced by Y. Ollivier [Oll09], this notion provides a discrete analogue to the celebrated Ricci curvature from Riemannian Geometry. It captures the same intuition: namely, it compares the distance between small balls (w.r.t to optimal transportation) to the distance between their centers: a ratio above (resp. below) one characterizes a positive (resp. negative) curvature. On a graph, a “ball” and its center are replaced by a (transition) probability distribution over neighbors, allowing to define the curvature of Markov Chains over the graph. It was computed mathematically in expectation for Erdős-Renyi random graphs [LLY11] and empirically for certain graphs representing the internet topology [Ni+15].

Other recent variants exist such as Discrete Foreman [For03] and sectional [Gu+19a] curvatures, although less is known about the latter.

### 1.3.2 Non-Euclidean Properties of Data Embedded in a Euclidean Space

In machine learning and neuroscience, certain computational structures and algorithms are known to yield disentangled representations without us understanding why, the most striking examples being perhaps convolutional neural networks and the ventral stream of the visual cortex in humans and primates. As for the latter, it was conjectured that representations may be disentangled by being flattened progressively and at a local scale [DC07]. An attempt at a formalization of the role of invariance in learning representations was made recently, being referred to as I-theory [Ans+13b]. In this framework and using the language of differential geometry, we will show that pooling over a group of transformations of the input contracts the metric and reduces its curvature, and provide quantitative bounds, in the aim of moving towards a theoretical understanding on how to disentangle representations.
1.3.3 Hyperbolic Representations

Interestingly, [De +18a] shows that arbitrary tree structures cannot be embedded with arbitrary low distortion (i.e. almost preserving their metric) in the Euclidean space with unbounded number of dimensions, but this task becomes surprisingly easy in the hyperbolic space with only 2 dimensions where the exponential growth of distances matches the exponential growth of nodes with the tree depth.

But how should one generalize deep neural models to non-Euclidean domains? Hyperbolic spaces constitute an interesting non-Euclidean domain to explore for two essential reasons: (i) important mathematical quantities such as distances, exponential map, gradients and geodesics are known in efficiently computable closed-form formulas and (ii) they are known to be better suited than the Euclidean space to represent trees, and hence potentially tree-like graphs as well.

Indeed, the tree-likeness properties of hyperbolic spaces have been extensively studied [Gro87]; [Ham17]; [Ung08] and used to visualize large taxonomies [LRP95] or to embed heterogeneous complex networks [Kri+10]. In machine learning, recently, hyperbolic representations greatly outperformed Euclidean embeddings for hierarchical, taxonomic or entailment data [De +18a]; [GBH18b]; [NK17a]. Disjoint subtrees from the latent hierarchical structure surprisingly disentangle and cluster in the embedding space as a simple reflection of the space’s negative curvature. However, appropriate deep learning tools are needed to embed feature data in this space and use it in downstream tasks. For example, implicitly hierarchical sequence data (e.g. textual entailment data, phylogenetic trees of DNA sequences or hierarchial captions of images) would benefit from suitable hyperbolic Recurrent Neural Network (RNN)s.

On the other hand, Euclidean word embeddings are ubiquitous nowadays as first layers in neural network and deep learning models for natural language processing. They are essential in order to move from the discrete word space to the continuous space where differentiable loss functions can be optimized. The popular models of Glove [PSM14], Word2Vec [Mik+13b] or FastText [Boj+16], provide efficient ways to learn word vectors fully unsupervised from raw text corpora, solely based on word co-occurrence statistics. These models are then successfully applied to word similarity and other downstream tasks and,
surprisingly (or not [Aro+16]), exhibit a linear algebraic structure that is also useful to solve word analogy.

However, unsupervised word embeddings still largely suffer from revealing antisymmetric word relations including the latent hierarchical structure of words. This is currently one of the key limitations in automatic text understanding, e.g. for tasks such as textual entailment [Bow+15]. To address this issue, [MC18]; [VM15] propose to move from point embeddings to probability density functions, the simplest being Gaussian or Elliptical distributions. Their intuition is that the variance of such a distribution should encode the generality/specificity of the respective word. However, this method results in losing the arithmetic properties of point embeddings (e.g. for analogy reasoning) and becomes unclear how to properly use them in downstream tasks.

To this end, we propose to take the best from both worlds: we embed words as points in a Cartesian product of hyperbolic spaces and, additionally, explain how they are bijectively mapped to Gaussian embeddings with diagonal covariance matrices, where the hyperbolic distance between two points becomes the Fisher distance between the corresponding probability distribution functions. This allows us to derive a novel principled is-a score on top of word embeddings that can be leveraged for hypernymy detection. We learn these word embeddings unsupervised from raw text by generalizing the Glove method. Moreover, the linear arithmetic property used for solving word analogy has a mathematical grounded correspondence in this new space based on the established notion of parallel transport in Riemannian manifolds.

1.3.4 Riemannian Optimization

Developing powerful stochastic gradient-based optimization algorithms is of major importance for a variety of application domains. In particular, for computational efficiency, it is common to opt for a first order method, when the number of parameters to be optimized is great enough. Such cases have recently become ubiquitous in engineering and computational sciences, from the optimization of deep neural networks to learning embeddings over large vocabularies.

This new need resulted in the development of empirically very successful first order methods such as Adagrad [DHS11], Adadelta [Zei12], Adam [KB15] or its recent update Amsgrad [RKK18].
Note that these algorithms are designed to optimize parameters living in a Euclidean space $\mathbb{R}^n$, which has often been considered as the default geometry to be used for continuous variables. However, a recent line of work has been concerned with the optimization of parameters lying on a Riemannian manifold, a more general setting allowing non-Euclidean geometries. This family of algorithms has already found numerous applications, including for instance solving Lyapunov equations [VV10], matrix factorization [Tan+14], geometric programming [SH15], dictionary learning [CS17] or hyperbolic taxonomy embedding [De+18b]; [GBH18b]; [NK17a]; [NK18].

A few first order stochastic methods have already been generalized to this setting (see section 4.5), the seminal one being Riemannian Stochastic Gradient Descent (RSGD) [Bon13], along with new methods for their convergence analysis in the geodesically convex case [ZS16]. However, the above mentioned empirically successful adaptive methods, together with their convergence analysis, remain to find their respective Riemannian counterparts.

Indeed, the adaptivity of these algorithms can be thought of as assigning one learning rate per coordinate of the parameter vector. However, on a Riemannian manifold, one is generally not given an intrinsic coordinate system, rendering meaningless the notions sparsity or coordinate-wise update.

### 1.3.5 Graph-like Data

#### 1.3.5.1 On the Success of Euclidean GCNs

Recently, there has been considerable interest in developing learning algorithms for structured data such as graphs. For example, molecular property prediction has many applications in chemistry and drug discovery [Vam+19]; [Yan+19].

Historically, graphs were systematically decomposed into features such as molecular fingerprints, turned into non-parametric graph kernels [She+11]; [Vis+10], or, more recently, learned representations via GNN [DBV16a]; [Duv+15]; [KW17b].

Indeed, the success of convolutional networks and deep learning for image data has inspired generalizations for graphs for which sharing parameters is consistent with the graph geometry. [Bru+14]; [HBL15] are the pioneers of spectral GCN in the graph Fourier space using lo-
ocalized spectral filters on graphs. However, in order to reduce the graph-dependency on the Laplacian eigenmodes, [DBV16b] approximate the convolutional filters using Chebyshev polynomials leveraging a result of [HVG11]. The resulting method is computationally efficient and superior in terms of accuracy and complexity. Further, [KW17a] simplify this approach by considering first-order approximations obtaining high scalability. The proposed GCN locally aggregates node embeddings via a symmetrically normalized adjacency matrix, while this weight sharing can be understood as an efficient diffusion-like regularizer. Recent works extend GCNs to achieve state of the art results for link prediction [ZC18], graph classification [HYL17]; [Xu+18] and node classification [KBG19]; [Vel+18].

1.3.5.2 Extending GCNs Beyond the Euclidean Domain

In spite of this success, on the one hand, certain types of data (e.g. hierarchical, scale-free or spherical data) have been shown to be better represented by non-Euclidean geometries [Bro+17]; [Def+19]; [Gu+19b]; [NK17a], leading in particular to the rich theories of manifold learning [RS00]; [TDL00] and information geometry [AN07]. The mathematical framework in vigor to manipulate non-Euclidean geometries is known as Riemannian geometry [Spi79]. On the other hand, GNN are also often underutilized in whole graph prediction tasks such as molecule property prediction. Specifically, while GNNs produce node embeddings for each atom in the molecule, these are typically aggregated via simple operations such as a sum or average, turning the molecule into a single vector prior to classification or regression. As a result, some of the information naturally extracted by node embeddings may be lost.

These observations led us to (i) propose a first variant of the GCN architecture, called $\kappa$-GCN, which internally represents data in a family of chosen non-Euclidean domains, and to propose (ii) another variant, called Optimal Transport Graph Neural Network (OTGNN) dispensing completely with the final node (sum/average) aggregation yielding the final Euclidean graph embedding, thus preserving the graph representation as a point-cloud of node embeddings considered w.r.t. Wasserstein (Optimal Transport) Geometry.
κ-GCN. An interesting trade-off between the general framework of Riemannian Geometry, which often yields computationally intractable geometric quantities, and the well-known Euclidean space, is given by manifolds of constant sectional curvature. They define together what are called hyperbolic (negative curvature), elliptic/spherical (positive curvature) and Euclidean (zero curvature) geometries. Benefits from representing certain types of data into hyperbolic [GBH18b]; [Gu+19b]; [NK17a]; [NK18] or spherical [Dav+18]; [Def+19]; [Gra+18]; [Gu+19b]; [Mat13]; [Wil+14]; [XD18] spaces have recently been shown. As mentioned earlier, this mathematical framework has the benefit of yielding efficiently computable geometric quantities of interest, such as distances, gradients, exponential map and geodesics. We propose a unified architecture for spaces of both (constant) positive and negative sectional curvature by extending the mathematical framework of Gyrovector spaces to positive curvature via Euler’s formula and complex analysis, thus smoothly interpolating between geometries of constant curvatures irrespective of their signs. This is possible when working with the Poincaré ball and stereographic spherical projection models of respectively hyperbolic and spherical spaces.

OTGNN. As mentioned above, we also seeked to address the potential loss of information incurred by the final node aggregation step present in nearly all GNN architectures to yield the final graph embedding. As it turns out, recent work by Togninalli et al. [Tog+19] already proposed to dispense with the aggregation step altogether and instead derive a kernel function over graphs by directly comparing node embeddings as point clouds through optimal transport (Wasserstein distance). Their non-parametric model yields better empirical performance over popular graph kernels, but has not been so far extended to the more challenging parametric case. Motivated by this observation and drawing inspiration from prototypical networks [SSZ17], we introduce a new class of GNN where the key representational step consists of comparing each input graph to a set of abstract prototypes, as illustrated in Figure 5.8. These prototypes play the role of dictionary items or basis functions in the comparison; they are also stored as point clouds as if they were encoded from actual real graphs. Each input graph is first encoded into a set of node embeddings using a GNN. We then compare this resulting embedding point cloud to those corresponding to the prototypes. Formally, the distance between two point clouds is measured by appeal
to optimal transport Wasserstein distances. The prototypes as abstract basis functions can be understood as keys that highlight property values associated with different structural features. In contrast to kernel methods, the prototypes are learned together with the GNN parameters in an end-to-end manner. Our model improves upon traditional aggregation by explicitly tapping into the full set of node embeddings without collapsing them first to a single vector. We theoretically prove that, unlike standard GNN aggregation, our model defines a class of set functions that is a universal approximator. Most importantly, introducing point clouds as free parameters creates a challenging optimization problem. Indeed, as the models are trained end-to-end, the primary signal is initially available only in aggregate form. If trained as is, the prototypes often collapse to single points, reducing the Wasserstein distance between point clouds into Euclidean comparisons of their means. To counter this effect, we introduce a contrastive regularizer which effectively prevents the model from collapsing (Section 5.4.2). We demonstrate empirically that it both improves model performance and generates richer prototypes.

Our contributions are summarized in the following section.

1.4 THEESIS CONTRIBUTIONS

This thesis is organized into four chapters, subsequently taking deep dives into manifold representations, methods in hyperbolic spaces, Riemannian adaptive optimization and finally graph-like data.

Chapter 1. We start by taking interest into the effect of pooling on the geometry of data representations. We prove two theorems about orbit contraction (Theorem 2.1) and flattening (Theorem 2.2), and illustrate the phenomenon of interest via numerical simulations and visualizations.

Chapter 2. Then, we explore representing data in hyperbolic spaces. We start by generalizing the word embedding method GLoVe, and describe how to evaluate similarities and analogies. In particular, leveraging a connection with Fisher information, we develop a new method (Algorithm 1) to assess hypernymy, i.e. entailment relationships between concepts. We motivate the use of hyperbolic geometry by computing averaged delta-hyperbolicities on a weighted graph obtained from co-
occurrence counts, upon which GloVe is based. We then take interest into generalizing neural architectures to hyperbolic spaces, adapting matrix multiplication, GRU and MLR, also called softmax. To understand how to perform bias translations, we connect gyro-addition to parallel transport, the latter being computed explicitly (Theorem 3.1) from the Levi-Civita connection and its Christoffel symbols. To adapt GRUs to hyperbolic spaces, we prove a so-called gyro-chain-rule (Lemma 3.5).

**Chapter 3.** Motivated by the lack of hyperbolic analogues to the celebrated Euclidean adaptive optimization algorithms (ADAM, ADAGRAD, AMSGRAD) – much needed for high performance of various embedding methods, including the training of GloVe – we study adapting those to certain Riemannian manifolds (Algorithm 4.1a). We obtain convergence guarantees (Theorems 4.1 and 4.2), recovering the known bounds in the particular case of Euclidean spaces. Our methods are validated on the task of embedding the WordNet hierarchy in the Poincaré Disk.

**Chapter 4.** Finally, we investigate leveraging non-Euclidean geometries to better represent graph-like data. We start by adapting GCN to hyperbolic and spherical spaces by leveraging Einstein midpoint operations (Definition 5.5) and explore certain of its mathematical properties (Theorems 5.3-5.6-5.7), showing good empirical performance on graph embeddings (w.r.t. distortion) and node classification. In a second part, we take interest into dispensing with the final aggregation step performed by most GNN architecture. We do so by computing Wasserstein distances (Optimal Transport) between point-clouds of node embeddings and so-called prototype point-clouds, i.e. free parameters. We study properties of universality (Theorem 5.11) and non-definiteness (Theorem 5.13) of the resulting kernel and introduce a new regularizer (Eq. (5.57)) inspired from Noise Contrastive Estimation (NCE), whose use we show is crucial in order to prevent the model from naturally collapsing back to the standard sum/average node aggregation. We illustrate the empirical efficacy of the method with both qualitative and quantitative experiments on molecular datasets.
What does disentangling representations mean? In machine learning and neurosciences, representations being tangled has two principal interpretations, and they are intimately connected with each other. The first one is geometrical: consider two sheets of paper of different colors, place one of the two on top of the other, and crumple them together in a paper ball; now, it may look difficult to separate the two sheets with a third one: they are tangled, one color sheet representing one class of a classification problem. The second one is analytical: consider a dataset being parametrized by a set of coordinates \( \{x_i\}_{i \in I} \), such as images parametrized by pixels, and a classification task between two classes of images. On the one hand, we cannot find a subset \( \{x_i\}_{i \in J} \) with \( J \subset I \) of this coordinate system such that a variation of these would not change the class of an element, while still spanning a reasonable amount of different images of this class. On the other hand, we are likely to be capable of finding a large amount of transformations preserving the class of any image of the dataset, without being expressible as linear transformations on this coordinate system, and this is another way to interpret representations or factors of variation as being tangled.

Why is disentangling representations important? On the physiological side, the brains of humans and primates alike have been observed to solve object recognition tasks by progressively disentangling their representations via the visual stream, from V1 to the IT cortex [DC07]; [DZR12]. On the side of deep learning, deep convolutional neural networks are also able to disentangle highly tangled representations, since a softmax – which, geometrically, performs essentially a linear separation – computed on the representation of their last hidden layer can yield very good accuracy [KSH12]. Conversely, disentangling representations might be sufficient to pre-solve practically any task relevant to the observed data [Ben13].

How can we design algorithms in order to move towards more disentangled representations? Although it was conjectured that the visual stream might disentangle representations by flattening them locally,
Thus inducing a decrease in the curvature globally [DC07], the mechanisms underlying such a disentanglement, whether it be for the brain or deep learning architectures, remain very poorly understood [Ben13]; [DC07]. However, it is now of common belief that computing representations that are invariant with respect to irrelevant transformations of the input data can help. Indeed, on the one hand, deep convolutional networks have been noticed to naturally learn more invariant features with deeper layers [Goo+09]; [LV15]; [TM16]. On the other hand, the V1 part of the brain similarly achieves invariance to translations and rotations via a “pinwheels” structure, which can be seen as a principal fiber bundle [Pet03]; [Pog+12]. Conversely, enforcing a higher degree of invariance with respect to not only translations, but also rotations, flips, and other groups of transformation has been shown to achieve state-of-the-art results in various machine learning tasks [BM13]; [CW16]; [CW17]; [DDK16]; [GD14]; [OM15], and is believed to help in linearizing small diffeomorphisms [Mal16]. To the best of our knowledge, the main theoretical efforts in this direction include the theory of scattering operators [Mal12]; [WB15] as well as I-theory [Ans+13a]; [Ans+13b]; [AP14]; [ARP16]. In particular, I-theory permits to use the whole apparatus of kernel theory to build invariant features [MVP15]; [Raj+17].

Our work builds a bridge between the idea that disentangling is a result of (i) a local decrease in the curvature of the representations, and (ii) building representations that are invariant to nuisance deformations, by proving that pooling over such groups of transformations results in a local decreasing of the curvature.

We start by providing some background material, after which we introduce our formal framework and theorems, which we then discuss in the case of the non-commutative group generated by translations and rotations.

Research presented in this chapter was conducted at the very beginning of the author’s PhD, and was only published as an ArXiv preprint [Béc17].

2.1 Background Material

2.1.1 Groups, Lie Algebras & Haar Measures

A group is a set $G$ together with a map $\cdot : G \times G \rightarrow G$ such that:
2.1 Background Material

1. \( \forall g, g', g'' \in G, \ g \cdot (g' \cdot g'') = (g \cdot g') \cdot g'' \),

2. \( \exists e \in G, \ \forall g \in G, \ g \cdot e = e \cdot g = g \),

3. \( \forall g \in G, \ \exists g^{-1} \in G : \ g \cdot g^{-1} = g^{-1} \cdot g = e \),

where \( e \) is called the identity element. We write \( gg' \) instead of \( g \cdot g' \) for simplicity. If, moreover, \( gg' = g'g \) for all \( g, g' \in G \), then \( G \) is said to be commutative or abelian.

A subgroup of \( G \) is a set \( H \subset G \) such that for all \( h, h' \in H, hh' \in H \) and \( h^{-1} \in H \). A subgroup \( H \) of a group \( G \) is said to be normal in \( G \) if for all \( g \in G, gh = Hg \), or equivalently, for all \( g \in G \) and \( h \in H \), \( ghg^{-1} \in H \). If \( G \) is abelian, then all of its subgroups are normal in \( G \).

A Lie group is a group which is also a smooth manifold, and such that its product law and inverse map are smooth with respect to its manifold structure. A Lie group is said to be locally compact if each of its element possesses a compact neighborhood. On every locally compact Lie group, one can define a Haar measure, which is a left-invariant, non-trivial Lebesgue measure on its Borel algebra, and is uniquely defined up to a positive scaling constant. If this Haar measure is also right-invariant, then the group is said to be unimodular. This Haar measure is always finite on compact sets, and strictly positive on non-empty open sets. Examples of unimodular Lie groups include in particular all abelian groups, compact groups, semi-simple Lie groups and connected nilpotent Lie groups.

A group \( G \) is said to be acting on a set \( X \) if we have a map \( \cdot : G \times X \to X \) such that for all \( g, g' \in G \), for all \( x \in X \), \( g \cdot (g' \cdot x) = (gg') \cdot x \) and \( e \cdot x = x \). If this map is also smooth, then we say that \( G \) is smoothly acting on \( X \). We write \( gx \) instead of \( g \cdot x \) for simplicity. Then, the group orbit of \( x \in X \) under the action of \( G \) is defined by \( G \cdot x = \{ gx \mid g \in G \} \), and the stabilizer of \( x \) by \( G_x = \{ g \in G \mid gx = x \} \). Note that \( G_x \) is always a subgroup of \( G \), and that for all \( x, y \in X \), we have either \( (G \cdot x) \cap (G \cdot y) = \emptyset \), or \( G \cdot x = G \cdot y \). Hence, we can write \( X \) as the disjoint union of its group orbits, i.e. there exists a minimal subset \( \tilde{X} \subset X \) such that \( X = \bigsqcup_{x \in \tilde{X}} G \cdot x \). The set of orbits of \( X \) under the action of \( G \) is written \( X/G \), and is in one-to-one correspondence with \( \tilde{X} \). Moreover, note that if \( H \) is a subgroup of \( G \), then \( H \) is naturally
acting on $G$ via $(h, g) \in H \times G \mapsto hg \in G$; if we further assume that $H$ is normal in $G$, then one can define a canonical group structure on $G/H$, thus turning the canonical projection $g \in G \mapsto H \cdot g$ into a group morphism.

A diffeomorphism between two manifolds is a map that is smooth, bijective and has a smooth inverse. A group morphism between two groups $G$ and $G'$ is a map $\varphi : G \to G'$ such that for all $g_1, g_2 \in G$, $\varphi(g_1g_2) = \varphi(g_1)\varphi(g_2)$. A group isomorphism is a bijective group morphism, and a Lie group isomorphism is a group isomorphism that is also a diffeomorphism.

The Lie algebra $\mathfrak{g}$ of a Lie group $G$ is its tangent space at $e$, and is endorsed with a bilinear map $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called its Lie bracket, and such that for all $x, y, z \in \mathfrak{g}$, $[x, y] = -[y, x]$ and $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$. Moreover, there is a bijection between $\mathfrak{g}$ and left-invariant vector fields on $G$, defined by $\xi \in \mathfrak{g} \mapsto \{g \in G \mapsto d_eL_g(\xi)\}$, where $L_g(h) = gh$ is the left translation. Finally, the flow $t \mapsto \phi_t$ of such a left-invariant vector field $X_\xi$ is given by $\phi_t(g) = g \exp(t\xi)$, where $\exp : \mathfrak{g} \to G$ is the exponential map on $G$.

For more on Lie groups, Lie algebras, Lie brackets and group representations, see [Kir08], and for a rapid and clear presentation of the notions of sectional curvature and Riemannian curvature, see [AH10].

2.1.2 I-theory

I-theory aims at understanding how to compute a representation of an image $I$ that is both unique and invariant under some deformations of a group $G$, and how to build such representations in a hierarchical way [Ans+13a]; [Ans+13b]; [AP14]; [ARP16]; [Pog+12].

Suppose that we are given a Hilbert space $\mathcal{X}$, typically $L^2(\mathbb{R}^2)$, representing the space of images. Let $G$ be a locally compact group acting on $\mathcal{X}$. Then, note that the group orbit $G \cdot I$ constitutes such an invariant and unique representation of $I$, as $G \cdot I = G \cdot (gI)$, for all $g \in G$, and since two group orbits intersecting each other are equal.
But how can we compare such group orbits? For an image \( I \in \mathcal{X} \), define the map \( \Theta^I : g \in G \mapsto gI \in \mathcal{X} \) and the probability distribution \( P_I(A) = \mu_G(\Theta^{-1}_I(A)) \) for any borel set \( A \) of \( \mathcal{X} \), where \( \mu_G \) is the Haar measure on \( G \). For \( I, I' \in \mathcal{X} \), write \( I \sim I' \) if and only if \( P_I = P_{I'} \). Hence, we could compare \( G \cdot I \) and \( G \cdot I' \) by comparing \( P_I \) and \( P_{I'} \). However, computing \( P_I \) can be difficult, so one must be looking for ways to approximate \( P_I \). If \( t \in S(L^2(\mathbb{R}^2)) \), define \( P_{\langle I, t \rangle} \) to be the distribution associated with the random variable \( g \mapsto \langle gI, t \rangle \). One can then prove that \( P_I = P_{I'} \) if and only if \( P_{\langle I, t \rangle} = P_{\langle I', t \rangle} \) for all \( t \in S(L^2(\mathbb{R}^2)) \), and then provide a lower bound on the sufficient number \( K \) of such templates \( t_k, 1 \leq k \leq K \), drawn uniformly on \( S(L^2(\mathbb{R}^2)) \), in order to recover the information of \( P_I \) up to some error \( \varepsilon \) and with high probability \( 1 - \delta \).

Finally, each \( P_{\langle I, t_k \rangle} \) can be approximated by a histogram

\[
h^k_n(I) = \frac{1}{|G|} \sum_{g \in G} \eta_n(\langle gI, t_k \rangle),
\]

if \( G \) is finite or

\[
h^k_n(I) = \frac{1}{\mu_G(G)} \int_{g \in G} \eta_n(\langle gI, t_k \rangle) d\mu_G(g),
\]

if \( G \) is compact, where \( \eta_n \) are various non-negative and possibly non-linear functions, \( 1 \leq n \leq N \), such as sigmoid, ReLU, modulus, hyperbolic tangent or \( x \mapsto |x|^p \), among others.

In the case where the group \( G \) is only partially observable (for instance if \( G \) is only locally compact but not bounded), one can define instead a “partially invariant representation”, replacing each \( h^k_n(I) \) by

\[
\frac{1}{\mu_G(G_0)} \int_{g \in G_0} \eta_n(\langle gI, t_k \rangle) d\mu_G(g),
\]

where \( G_0 \) is a compact subset of \( G \) which can be observed in practice. Under some “localization condition” (see [Ans+13b]), it can be proved that this representation is invariant under deformations by elements of \( G_0 \). When this localization condition is not met, we do not have any exact invariance a priori, but one might expect that the variation in directions defined by the symmetries of \( G_0 \) is going to be reduced.
For instance, let $G$ be the group $\mathbb{R}^2$ of translations in the plane, $G_0 = [-a,a]^2$ for some $a > 0$, $\eta : x \mapsto (\sigma(x))^2$ where $\sigma$ is a point-wise non-linearity commonly used in neural networks, and $t_k \in S(L^2(\mathbb{R}^2))$ for $1 \leq k \leq K$. Then, note that the quantities

$$\sqrt{|G_0|h_k(I)} = \sqrt{\sum_{g \in G_0} \eta(\langle gI, t_k \rangle)}, \quad (2.4)$$

for $1 \leq k \leq K$ are actually computed by a 1-layer convolutional neural network with filters $(t_k)_{1 \leq k \leq K}$, non-linearity $\sigma$ and $L^2$-pooling. Moreover, the family $(\sqrt{|G_0|h_k(gI)})_{g \in G}$ is exactly the output of this convolutional layer, thus describing a direct correspondence between pooling and locally averaging over a group of transformations.

Another correspondence can be made between this framework and deep learning architectures. Indeed, assume that during learning, the set of filters of a layer of a convolutional neural network becomes stable under the action of some unknown group $G$ acting on the pixel space, and denote by $\sigma$ the point-wise non-linearity computed by the network. Moreover, suppose that the convolutional layer and point-wise non-linearity are followed by an $L^p$-pooling, defined by $\Pi^p_\phi(I)(x) = (\int_{y \in \mathbb{R}^2} |I(y) \mathbf{1}_{[0,a]^2}(x - y)|^p \, dy)^{1/p}$. Then, observe that the convolutional layer outputs the following feature maps:

$$\{\Pi^p_\phi(\sigma(I \ast t_k))\}_{1 \leq k \leq K}. \quad (2.5)$$

Besides, if the group $G$ has a unitary representation, and if its action preserves $\mathbb{R}^2$, then for all $g \in G$ and $1 \leq k \leq K$, we have

$$\Pi^p_\phi(\sigma(gI \ast t_k)) = \Pi^p_\phi(\sigma(g(I \ast g^{-1}t_k))) = g \Pi^p_\phi(\sigma(I \ast g^{-1}t_k)). \quad (2.6)$$

Then, the following layer of the convolutional network is going to compute the sum across channels $k$ of these different signals. However, if our set of filters $t_k$ can be written as $G_0 \cdot t$ for some filter $t$ and a subpart $G_0$ of $G$, then this sum will be closely related to a histogram as in $I$-theory:

$$\sum_{g \in G_0} \Pi^p_\phi(\sigma(I \ast gt)) = \sum_{g \in G_0} g \Pi^p_\phi(\sigma(g^{-1}I \ast t_k)). \quad (2.7)$$

In other words, (local) group invariances are free to appear during learning among filters of a convolutional neural network, and will naturally be pooled over by the next layer. For more on this, see [BSL13];
Finally, let’s mention that this implicit pooling over symmetries can also be computed explicitly, and such group invariances across filters enforced, if we know the group in advance, as in G-CNNs and steerable CNNs [CW16]; [CW17].

2.2 MAIN RESULTS: FORMAL FRAMEWORK AND THEOREMS

Let $G$ be a finite-dimensional, locally compact and unimodular Lie group smoothly acting on $\mathbb{R}^2$. This defines an action $(L_g f)(x) = f(g^{-1}x)$ on $L^2(\mathbb{R}^2)$. Let $G_0$ be a compact neighborhood of the identity element $e$ in $G$, and assume that there exists $\lambda > 0$ such that for all $g_0 \in G_0$, $\sup_{x \in \mathbb{R}^2} |J_{g_0}(x)| \leq \lambda$, where $J_g$ is the determinant of the Jacobian matrix of $g$ seen as a diffeomorphism of $\mathbb{R}^2$. We define $\Phi : L^2(\mathbb{R}^2) \rightarrow L^2(\mathbb{R}^2)$, the averaging operator on $G_0$, by

$$
\Phi(f) = \frac{1}{\mu_G(G_0)} \int_{g \in G_0} L_g f \, d\mu_G(g).
$$

DISCLAIMER. Although the mathematics in this paragraph may initially appear overly complicated to the reader, they were necessary to formulate things rigorously, and most importantly in order to avoid writing something wrong, which happens so easily when manipulating these objects. However, the profane reader may safely skip the mathematical subtleties and remain able to understand the high-level intuition behind the point we try to make about orbit contraction and flattening.

2.2.1 Group Pooling Results in Orbit Contraction

Our first result describes how the euclidean distance in $L^2(\mathbb{R}^2)$ between a function $f$ and its translation by some $g \in G_0$ is contracted by this locally averaging operator.
**Theorem 2.1.** For all \( f \in L^2(\mathbb{R}^2) \), for all \( g \in G \),
\[
\| \Phi(L_g f) - \Phi(f) \|_2 \leq \sqrt{\lambda} \max \left( 1, \sqrt{\| f \|_\infty} \right) \frac{\mu_G((G_0 g) \Delta G_0)}{\mu_G(G_0)} \| f \|_2.
\]

(2.9)

**Proof.** We have
\[
\mu_G(G_0)^2 \| \Phi(L_g f) - \Phi(f) \|_2^2 =
\int_{x \in \mathbb{R}^2} \left( \int_{g' \in G_0} L_{g' g} f(x) - L_{g'} f(x) d\mu_G(g') \right)^2 dx,
\]
but
\[
\int_{g' \in G_0} (L_{g' g} f(x) - L_{g'} f(x)) d\mu_G(g') =
\int_{g'' \in C_0 g} L_{g''} f(x) d\mu_G(g'') - \int_{g' \in G_0} L_{g'} f(x) d\mu_G(g'),
\]
i.e. setting \( g'' = g' g \) and using the right-invariance of \( \mu_G \),
\[
\int_{g' \in G_0} (L_{g' g} f(x) - L_{g'} f(x)) d\mu_G(g') =
\int_{g'' \in C_0 g} L_{g''} f(x) d\mu_G(g'') - \int_{g' \in G_0 \setminus C_0 g} L_{g'} f(x) d\mu_G(g').
\]
And using \( \int_A h - \int_B h = (\int_A h + \int_{A \cap B} h) - (\int_B h + \int_{B \cap A} h) = \int_{A \setminus B} h - \int_{B \setminus A} h \), we have:
\[
\int_{g' \in G_0} (L_{g' g} f(x) - L_{g'} f(x)) d\mu_G(g') =
\int_{g'' \in C_0 g \setminus G_0} L_{g''} f(x) d\mu_G(g'') - \int_{g' \in G_0 \setminus C_0 g} L_{g'} f(x) d\mu_G(g').
\]
(2.13)

Plugging this in the first equation gives
\[
\mu_G(G_0) \| \Phi(L_g f) - \Phi(f) \|_2 =
\| \int_{g' \in G_0 \setminus G_0} (L_{g'} f)d\mu_G(g') - \int_{g' \in G_0 \setminus G_0} (L_{g'} f)d\mu_G(g') \|_2,
\]
(2.14)
i.e. using a triangle inequality

\[ \mu_G(G_0) \| \Phi(L_g f) - \Phi(f) \|_2 \leq \left\| \int_{g' \in G_0 \setminus G_0} (L_{g'} f) d\mu_G(g') \right\|_2 + \left\| \int_{g' \in G_0 \setminus G_0} (L_{g'} f) d\mu_G(g') \right\|_2. \]  

(2.15)

Now observe that by interverting the integrals using Fubini’s theorem,

\[ \left\| \int_{g' \in G_0 \setminus G_0} (L_{g'} f) d\mu_G(g') \right\|_2 = \sqrt{\int_{g_1 \in G_0 \setminus G_0} \int_{g_2 \in G_0 \setminus G_0} (\int_{x \in \mathbb{R}^2} (L_{g_1} f)(x) (L_{g_2} f)(x) dx) d\mu_G(g_1) d\mu_G(g_2)}, \]  

(2.16)

and using a Cauchy-Schwarz inequality,

\[ \left\| \int_{g' \in G_0 \setminus G_0} (L_{g'} f) d\mu_G(g') \right\|_2 \leq \sqrt{\int_{g_1 \in G_0 \setminus G_0} \int_{g_2 \in G_0 \setminus G_0} \|L_{g_1} f\|_2 \|L_{g_2} f\|_2 d\mu_G(g_1) d\mu_G(g_2)}. \]  

(2.17)

As for all \( g' \in G_0 \) we have \( \|L_{g'} f\|_2 = \|f\sqrt{|I_{g'}|}\|_2 \leq \sqrt{\lambda} \|f\|_2 \) with a change of variables, we have:

\[ \left\| \int_{g' \in G_0 \setminus G_0} (L_{g'} f) d\mu_G(g') \right\|_2 \leq \sqrt{\lambda} \mu_G(G_0 \setminus (G_0 G)). \]  

(2.18)

For the other term, note that by setting \( g'' = g' g^{-1} \), we have

\[ \left\| \int_{g'' \in G_0 \setminus G_0} (L_{g''} f) d\mu_G(g'') \right\|_2 = \left\| \int_{g'' \in G_0 \setminus G_0} (L_{g''} g f) d\mu_G(g'') \right\|_2 \]  

(2.19)

\[ = \left\| \int_{g'' \in G_0 \setminus G_0} (L_{g''} L_g f) d\mu_G(g'') \right\|_2, \]  

(2.20)

and then similarly,

\[ \left\| \int_{g'' \in G_0 \setminus G_0} (L_{g''} L_g f) d\mu_G(g'') \right\|_2 = \sqrt{\int_{g_1 \in G_0 \setminus G_0} \int_{g_2 \in G_0 \setminus G_0} \|L_{g_1} L_g f\|_2 \|L_{g_2} L_g f\|_2 d\mu_G(g_1) d\mu_G(g_2)}. \]  

(2.21)
As for all $g' \in G_0$ we have $\|L_{g'} L_g f\|_2 = |f \sqrt{|J_{g'}|} | \leq \sqrt{\lambda} \|J_g\|_\infty \|f\|_2$, we have

$$\|\int_{g' \in G_0 \setminus G} (L_{g'} f) d\mu_G(g')\|_2 \leq \sqrt{\lambda} \|J_g\|_\infty \mu_G(G_0 \setminus (G_0 g^{-1})) \|f\|_2. \quad (2.22)$$

Therefore

$$\mu_G(G_0) \|\Phi(L_g f) - \Phi(f)\|_2 \leq \sqrt{\lambda} \|J_g\|_\infty \mu_G(G_0 \setminus (G_0 g^{-1})) \|f\|_2 + \sqrt{\lambda} \mu_G(G_0 \setminus (G_0 g)) \|f\|_2, \quad (2.23)$$

and the following fact concludes the proof:

$$\mu_G(G_0 \setminus (G_0 g^{-1})) + \mu_G(G_0 \setminus (G_0 g)) = \mu_G((G_0 g) \setminus G_0) + \mu_G(G_0 \setminus (G_0 g))$$

$$= \mu_G((G_0 g) \Delta G_0). \quad (2.24)$$

$$= \mu_G((G_0 g) \Delta G_0). \quad (2.25)$$

The symbol $\Delta$ above is defined $A\Delta B = (A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A)$. Note that, as one could have expected, this result doesn’t depend on the scaling constant of the Haar measure. Intuitively, this result formalizes the idea that locally averaging with respect to some factors of variation, or coordinates, will reduce the variation with respect to those coordinates. The following drawings illustrate the intuition behind Theorem 2.1, where we pass from left to right by applying $\Phi$.

Note that the quantity $\frac{\mu_G((G_0 g) \Delta G_0)}{\mu_G(G_0)}$, depending on the geometry of the group, is likely to decrease when we increase the size of $G_0$: if $G = \mathbb{R}^2$ is the translation group, $G_0 = [0, a]^2$ for some $a > 0$, and $g_\varepsilon$ is the translation by the vector $(\varepsilon, \varepsilon)$, then $\mu_G$ is just the usual Lebesgue measure in $\mathbb{R}^2$ and

$$\frac{\mu_G((G_0 g_\varepsilon) \Delta G_0)}{\mu_G(G_0)} \sim_{\varepsilon \to 0} \frac{2ae}{a^2} = \frac{4\varepsilon}{\sqrt{\mu_G(G_0)}}. \quad (2.26)$$

Indeed, locally averaging over a wider area will decrease the variation even more.
2.2.2 Group Pooling Results in Orbit Flattening

As images are handily represented by functions from the space of pixels $\mathbb{R}^2$ to either $\mathbb{R}$ or $\mathbb{C}$, let us define our dataset $\mathcal{X}$ to be a finite-dimensional manifold embedded in a bigger space of functions $\mathcal{Y}$. As for technical reasons we will need our functions to be $L^2$, smooth, and with a gradient having a fast decay at infinity, we choose $\mathcal{Y}$ to be the set of functions $f \in L^2(\mathbb{R}^2) \cap C^\infty(\mathbb{R}^2)$ such that $|\langle \nabla f(x), x \rangle| = O_{x \to \infty}(\frac{1}{\|x\|^2})$, for some fixed small $\varepsilon > 0$. Note that in practice, images are only non-zero on a compact domain, therefore these assumptions are not restrictive.

Further assume that for all $f \in \mathcal{X}$, for all $g \in G$, $L_g f \in \mathcal{X}$. Intuitively, $\mathcal{X}$ is our manifold of images, and $G$ corresponds to the group of transformations that are not relevant to the task at hand. Recall that from $I$-theory, the orbit of an image $f$ under $G$ constitutes a good unique and invariant representation. Here, we are interested in comparing $G \cdot f$ and $\Phi(G \cdot f)$, i.e. before and after locally averaging.
But how can we compute a bound on the curvature of $\Phi(G \cdot f)$? It is well known that in a Lie group endorsed with a bi-invariant pseudo-Riemannian metric $\langle \cdot , \cdot \rangle$, the Riemann curvature tensor is given by

$$R(X, Y, Z, W) = -\frac{1}{4} \langle [X, Y], [Z, W] \rangle,$$  \hspace{1cm} (2.27)

where $X, Y, Z, W$ are left-invariant vector-fields, and hence if $(X, Y)$ forms an orthonormal basis of the plane they span, then the sectional curvature is given by

$$\kappa(X \wedge Y) = R(X, Y, Y, X) = \frac{1}{4} \langle [X, Y], [X, Y] \rangle.$$  \hspace{1cm} (2.28)

Therefore, would we be able to define a Lie group structure and a bi-invariant pseudo-Riemannian metric on $\Phi(G \cdot f)$, we could use this formula to compute its curvature. First, we are going to define a Lie group structure on $G \cdot f$, which we will then transport on $\Phi(G \cdot f)$. As a Lie group structure is made of a smooth manifold structure and a compatible group structure, we need to construct both. In order to obtain the group structure on the orbit, let’s assume that the stabilizer $G_f$ is normal; a condition that is met for instance if $G$ is abelian, or if this subgroup is trivial, meaning that $f$ does not have internal symmetries corresponding to those of $G$, which is only a technical condition, as it can be enforced in practice by slightly deforming $f$, by breaking the relevant symmetries with a small noise. Besides, in order to obtain a smooth manifold structure on the orbits, we need to assume that $G_f$ is an embedded Lie subgroup of $G$, which, from proposition 2.3, is met automatically when this group admits a finite-dimensional representation.

Then, from proposition 2.4, there is one and only one manifold structure on the topological quotient space $G/G_f$ turning the canonical projection $\pi : G \rightarrow G/G_f$ into a smooth submersion; moreover, the action of $G$ on $G/G_f$ is smooth, $G/G_f$ is a Lie group, $\pi$ is a Lie group morphism, the Lie algebra $\mathfrak{g}_f$ of $G_f$ is an ideal of the Lie algebra $\mathfrak{g}$ of $G$ and the linear map from $T_eG/T_eG_f$ to $T_{eG_f}(G/G_f)$ induced by $T_e\pi$ is a Lie algebra isomorphism from $\mathfrak{g}/\mathfrak{g}_f$ to the Lie algebra of $G/G_f$.

Finally, we need a geometrical assumption on the orbits, insuring that $G$ is warped on $G \cdot f$ in a way that is not “fractal”, i.e. that this orbit can
be given a smooth manifold structure: assume that $G \cdot f$ is locally closed in $\mathcal{X}$. Using this assumption and proposition 2.5, the canonical map $\Theta_f : G/G_f \to \mathcal{X}$ defined by $\Theta_f(gG_f) = L_g f$ is a one-to-one immersion, whose image is the orbit $G \cdot f$, which is a submanifold of $\mathcal{X}$; moreover, $\Theta_f$ is a diffeomorphism from $G/G_f$ to $G \cdot f$. Further notice that $\Theta_f$ is $G$-equivariant, i.e. for all $g, g' \in G$,

$$\Theta_f(g(g'G_f)) = L_g g' = L_g L_g f = L_g \Theta_f(g'G_f).$$

Moreover, we can define on $G \cdot f$ a group law by

$$(L_{g_1} f) \cdot (L_{g_2} f) := L_{g_1 g_2} f,$$

for $g_1, g_2 \in G$. Indeed, let’s prove that this definition doesn’t depend on the choice of $g_1, g_2$. Assume that $g_i = a_i b_i$ for $a_i \in G$ and $b_i \in G_f$, $i \in \{1, 2\}$. Then, as $G_f$ is normal in $G$, there exists $b'_1 \in G_f$ such that $b_1 a_2 = a_2 b'_1$. Then $g_1 g_2 = a_1 a_2 b'_1 b_2$ and hence $L_{g_1 g_2} f = L_{a_1 a_2} f$, and this group law is well-defined. Now that $G \cdot f$ is a group, observe that $\Theta_f$ is a group isomorphism from $G/G_f$ to $G \cdot f$. Indeed, it is bijective since it is a diffeomorphism, and it is a group morphism as

$$\Theta_f((gG_f)(g'G_f)) = \Theta_f((gg')G_f)$$

$$= L_{gg'} f = (L_g f) \cdot (L_{g'} f)$$

$$= \Theta_f(gG_f) \cdot \Theta_f(g'G_f).$$

Hence, $G \cdot f$ is also a Lie group, since $G/G_f$ is a Lie group and $\Theta_f : G/G_f \to G \cdot f$ is a diffeomorphism. Moreover, $\text{Lie}(G \cdot f)$ is isomorphic to $g/g_f$ as a Lie algebra, since they are isomorphic as vector spaces ($\Theta_f$ being an immersion), and by the fact that the pushforward of a diffeomorphism always preserves the Lie bracket.

Now that we have defined a Lie group structure on $G \cdot f$, how can we obtain one on $\Phi(G \cdot f)$? Suppose that $\Phi$ is injective on $G \cdot f$ and on $\text{Lie}(G \cdot f)$. We can thus define a group law on $\Phi(G \cdot f)$ by:

$$\forall g_1, g_2 \in G/G_f, \Phi(L_{g_1} f) \cdot \Phi(L_{g_2} f) := \Phi(L_{g_1 g_2} f).$$

As the inverse function theorem tells us that $\Phi$ is a diffeomorphism from $G \cdot f$ onto its image, $\Phi(G \cdot f)$ is now endorsed with a Lie group structure. However, in order to carry out the relevant calculations, we still need to define left-invariant vector-fields on our Lie group orbits.
For all $\zeta \in \mathfrak{g}$, define the following left-invariant vector-fields respectively on $G \cdot f$ and $\Phi(G \cdot f)$:

$$X_\zeta : L_g f \mapsto \frac{d}{dt}|_{t=0} (L_g L_{\exp(t\zeta)} f), \quad (2.34)$$

$$\tilde{X}_\zeta : \Phi(L_g f) \mapsto \frac{d}{dt}|_{t=0} \Phi(L_g L_{\exp(t\zeta)} f). \quad (2.35)$$

We can now state the following theorem:

**Theorem 2.2.** For all $f \in \mathcal{X}$, for all $\zeta, \zeta' \in \mathfrak{g}$,

$$\| [\tilde{X}_\zeta, \tilde{X}_{\zeta'}] \Phi(f) \|_2^2 \leq \lambda \left[ \frac{d}{ds}|_{s=0} \frac{\mu_G((G_0 \exp(s[\zeta, \zeta']))(\Delta G_0))}{\mu_G(G_0)} \right]^2 \| f \|_2^2. \quad (2.36)$$

**Proof.** Let us start by proving the following lemma:

**Lemma 2.1.** For all $f \in \mathcal{X}$ and $\zeta \in \mathfrak{g}$,

$$\frac{d}{dt}|_{t=0} \Phi(L_{\exp(t\zeta)} f) = \Phi\left( \frac{d}{dt}|_{t=0} (L_{\exp(t\zeta)} f) \right).$$

**Proof.** For all $x \in \mathbb{R}^2$,

$$\left( \frac{d}{dt}|_{t=0} \Phi(L_{\exp(t\zeta)} f) \right)(x) =$$

$$\frac{1}{\mu_G(G_0)} \int_{g' \in G_0} (L_{g'} L_{\exp(t\zeta)} f)(x) d\mu_{G_0}(g')|_{t=0}$$

$$= \frac{1}{\mu_G(G_0)} \int_{g' \in G_0} \frac{d}{dt}|_{t=0} (f(\exp(-t\zeta)g'x)) d\mu_{G_0}(g')$$

$$= \frac{1}{\mu_G(G_0)} \int_{g' \in G_0} d(g'^{-1}x)f(-\zeta(g'^{-1}x)) d\mu_{G_0}(g')$$

$$= \Phi\left( d.f(-\zeta(\cdot)) \right)(x)$$

$$= \Phi\left( \frac{d}{dt}|_{t=0} (L_{\exp(t\zeta)} f) \right)(x). \quad (2.37)$$

$\square$
As \( \Phi \) realizes a diffeomorphism from \( G \cdot f \) onto its image, and as \( \Phi \) equals its differential from Lemma 2.1, we have that for all vector field \( X \) on \( G \cdot f \), \( \Phi_*(X)(\Phi(f)) = (d\Phi)_f(X(f)) = \Phi(X(f)) \). Hence
\[
[X_{\xi}, X_{\zeta}]_{\Phi(f)} = [\Phi(X_{\xi}), \Phi(X_{\zeta})]_f = [\Phi(X_{\xi}) \circ \Phi^{-1}, \Phi(X_{\zeta}) \circ \Phi^{-1}]_{\Phi(f)} = [\Phi_*(X_{\xi}), \Phi_*(X_{\zeta})]_{\Phi(f)} = \Phi_*([X_{\xi}, X_{\zeta}])_{\Phi(f)} = \Phi([X_{\xi}, X_{\zeta}])_{\Phi(f)}. \tag{2.38}
\]

Recall that the Lie bracket of left-invariant vector fields is given by the opposite of the Lie bracket of their corresponding generators, hence in our case:
\[
[X_{\xi}, X_{\zeta}] = X_{-[\xi, \zeta]} = -X_{[\xi, \zeta]}. \tag{2.43}
\]

Therefore,
\[
||[X_{\xi}, X_{\zeta}]_{\Phi(f)}||_2 = ||\Phi([X_{\xi}, X_{\zeta}])_{f}||_2 \tag{2.44}
= ||\Phi(X_{[\xi, \zeta]})_{f}||_2 \tag{2.45}
= ||\Phi(\lim_{t \to 0} \frac{1}{t}(L_{\exp(t[\xi, \zeta]})f - f))||_2 \tag{2.46}
= ||\lim_{t \to 0} \frac{1}{t} (\Phi(L_{\exp(t[\xi, \zeta]})f) - \Phi(f))||_2. \tag{2.47}
\]

From Theorem 2.1, we have
\[
||\Phi(L_{\exp(t[\xi, \zeta]})f) - \Phi(f)||_2 \leq \sqrt{\lambda \max(1, \sqrt{|J_{\exp(t[\xi, \zeta])}|}_\infty)} \times \frac{\mu_G((G_0 \exp(t[\xi, \zeta]))\Delta G_0)}{\mu_G(G_0)} \cdot ||f||_2. \tag{2.48}
\]

As \( \exp(t[\xi, \zeta]) \to e \) when \( t \to 0 \), its Jacobian goes to 1. Moreover, as \( f \) has a gradient with fast decay, we can take the limit out of the \( L^2 \)-norm, which concludes the proof. \( \square \)

As \( \mathcal{X} \) is a manifold embedded in \( L^2(\mathbb{R}^2) \), it inherits a Riemannian metric by projection of the usual inner-product of \( L^2(\mathbb{R}^2) \) on the tangent bundle of \( \mathcal{X} \). Moreover, if we further assume that for all \( g \in G \), \( |J_g| = 1 \), then this Riemannian metric is bi-invariant, and we can finally use the above formula on the Riemannian curvature, together with the previous inequality, to compute a bound on the curvature in a Lie
group endorsed with an bi-invariant metric:

**Corollary.** For all \( f \in \mathcal{X} \), for all \( \xi, \xi' \in \mathfrak{g} \),

\[
0 \leq R_{\Phi(f)}(\tilde{X}_\xi, \tilde{X}_{\xi'}, \tilde{X}_\xi, \tilde{X}_{\xi'}) \leq \left[ \frac{1}{2} \frac{d}{ds}_{|s=0} \mu_G((G_0 \exp(s[\xi, \xi'])) \Delta G_0) \right]^2 \|f\|^2.
\]

(2.49)

And if \((\tilde{X}_\xi, \tilde{X}_{\xi'})\) forms an orthonormal basis of the plane they span in \(\text{Lie}(\Phi(G \cdot f)) = \Phi(\text{Lie}(G \cdot f))\), then:

\[
0 \leq \kappa_{\Phi(f)}(\tilde{X}_\xi \wedge \tilde{X}_{\xi'}) \leq \left[ \frac{1}{2} \frac{d}{ds}_{|s=0} \mu_G((G_0 \exp(s[\xi, \xi'])) \Delta G_0) \right]^2 \|f\|^2.
\]

(2.50)

**Remark.** The sectional curvature of the basis \((\tilde{X}_\xi, \tilde{X}_{\xi'})\) at \(\Phi(f)\) is also the Gaussian curvature of the two-dimensional surface swept out by small geodesics induced by linear combinations of \(\tilde{X}_\xi(\Phi(f))\) and \(\tilde{X}_{\xi'}(\Phi(f))\).

Among well-known finite-dimensional, locally compact and unimodular Lie group smoothly acting on \(\mathbb{R}^2\), there are the group \(\mathbb{R}^2\) of translations, the compact groups \(O(2)\) and \(SO(2)\), the euclidean group \(E(2)\), as well as transvections, or shears. Moreover, another class of suitable unimodular Lie groups is given by the one-dimensional flows of Hamiltonian systems, which, as deformations of images, could be interpreted as the smooth evolutions of the screen in a video over time, provided that these evolutions can be expressed as group actions on the pixel space.

### 2.2.3 Orbit Flattening of \(\mathbb{R}^2 \ltimes SO(2)\)

Finally, let’s see what Theorem 2.2 gives us in the case \(G = \mathbb{R}^2 \ltimes SO(2)\). Note that this group is not commutative, and its curvature form is not identically zero. Let \(\theta \in (-\pi, \pi), a > 0\), and \(G_0 = [-\theta, \theta] \times [0, a]^2\). A representation of this group is given by matrices of the form

\[
g(\theta, x, y) = \begin{pmatrix}
cos(\theta) & -\sin(\theta) & x \\
\sin(\theta) & \cos(\theta) & y \\
0 & 0 & 1
\end{pmatrix},
\]

(2.51)
and a representation of its Lie algebra is given by
\[
\zeta(\xi, x, y) = \begin{pmatrix} 0 & -\xi & x \\ \xi & 0 & y \\ 0 & 0 & 0 \end{pmatrix}.
\]

(2.52)

The Lie bracket is then given by
\[
[\zeta(\xi, x, y), \zeta(\xi', x', y')] = \zeta(\xi, x, y)\zeta(\xi', x', y') - \zeta(\xi', x', y')\zeta(\xi, x, y)
\]
\[
= \zeta(0, \xi'y - \xi'x, \xi'x - \xi'x).
\]

(2.53)

(2.54)

As the exponential map on the group of translations is the identity map, and as the Haar measure on \(\mathbb{R}^2 \times SO(2)\) is just the product of the Haar measures on \(\mathbb{R}^2\) and \(SO(2)\), we have
\[
\mu_G((G_0 \exp(s[\zeta(\xi, x, y), \zeta(\xi', x', y')]) \Delta G_0) =
2\theta \times \mu_{\mathbb{R}^2}([[s(\xi'y - \xi'x), s(\xi'y - \xi'x) + a] \times
[s(\xi'x - \xi'x), s(\xi'x - \xi'x) + a]) \Delta [0, a]^2),
\]

(2.55)

and \(\mu_G(G_0) = 2\theta a^2\). Therefore, when \(s \to 0\), we have
\[
\mu_G((G_0 \exp(s[\zeta(\xi, x, y), \zeta(\xi', x', y')]) \Delta G_0) \sim
2\theta \times 2(as(\xi'y - \xi'x) + as(\xi'x - \xi'x)) =
4\theta as(\xi'x - \xi'y) - \zeta'(x - y)),
\]

(2.56)

from what we deduce that
\[
\left[ \frac{1}{2} \frac{d}{ds} \bigg|_{s=0} \mu_G((G_0 \exp(s[\zeta(\xi, x, y), \zeta(\xi', x', y')]) \Delta G_0) \right]^2 = \mu_G(G_0)
\]
\[
\frac{(\xi'(x' - y') - \zeta'(x - y))^2}{a^2}.
\]

(2.57)

As a consequence, if \(f \in \mathcal{X}\) is an image in our dataset, of \(L^2\)-norm equal to 1, and if we choose \(\zeta(\xi, x, y)\) and \(\zeta(\xi', x', y')\) such that the \(L^2\) functions \(\tilde{X}_{\zeta(\xi, x, y)}(\Phi(f))\) and \(\tilde{X}_{\zeta(\xi', x', y')}(\Phi(f))\) are orthogonal in \(L^2\) and have \(L^2\)-norm equal to 1, then the Gaussian curvature \(\kappa\) of the 2-dimensional surface swept out by these two vector fields around \(\Phi(f)\), in the Lie group \(\Phi(G \cdot f)\), is smaller than:
\[
\kappa \leq \frac{(\xi'(x' - y') - \zeta'(x - y))^2}{a^2}.
\]

(2.58)
2.3 Numerical Experiments: Shears and Translations on Images

2.3.1 Synthetic Orbit Contraction in Pixel Space

In this subsection, we present numerical simulations illustrating the theoretical results. Experiments were run twice, with the same parameters but on two different grayscale images of size $100 \times 100$: a handwritten ‘9’, which is a very simple image, and a grayscale version of ‘Lena’, being closer to a natural image. Note that each image can be seen as a point in $\mathbb{R}^{100^2}$, where each coordinate corresponds to a pixel value.

For each image $I$, we start by defining a set of transformations $\tilde{G}$ which is a discrete approximation of a connected subpart of the continuous Lie group $G$, and apply each of them to the image $I$, thus obtaining a discrete point-cloud $\tilde{G} \cdot I$ approximating the continuous group orbit $G \cdot I$. From this point-cloud, we build a second one, denoted by $\Phi(\tilde{G} \cdot I)$, which is obtained by replacing each point $gI$ in $\tilde{G} \cdot I$ by the average $\Phi(gI)$ of the set $\{\tilde{g}gI\}_{\tilde{g} \in \tilde{G}_0}$ where $\tilde{G}_0$ is a discrete set of transformations approximating the continuous neighbourhood $G_0$ of the identity element. Hence, from a point $I \in \mathbb{R}^{100^2}$, we build two point-clouds $\tilde{G} \cdot I \subset \mathbb{R}^{100^2}$ and $\Phi(\tilde{G} \cdot I) \subset \mathbb{R}^{100^2}$, where each point represents an image.

As the Lie groups we use in our experiments are manifolds of dimension either 1 or 2, our two point-clouds will be points sampled on 1- or 2-dimensional manifolds. Hence, even if these datasets are embedded in a euclidean space of very high dimension, they actually lie on much smaller-dimensional subspaces, meaning that using a good coordinate system, we could visualize them in a 3-dimensional space with very little loss of information. In order to find this coordinate system, we use a Multidimensional Scaling (MDS), which preserves as much as possible euclidean distances between points.

Note that MDS was applied to the union of the two datasets $\tilde{G} \cdot I$ and $\Phi(\tilde{G} \cdot I)$, so that we can visualize both of them in the same plot, as blue and red points respectively. On each of the six figures, the plot on the left-hand side (resp. right-hand side) corresponds to experiments obtained using the hand-written ‘9’ (resp. using ‘Lena’).
2.3 NUMERICAL EXPERIMENTS: SHEARS AND TRANSLATIONS ON IMAGES

These two images are shown in Figure 2.2 (a). Then, Figures 2.2 and 2.3 present the plots obtained by translating these images along the $x$-axis with 50 translations spaced uniformly in the range $[-20, 20]$.

In all other figures, the blue and the red point-clouds contain $50^2 = 2500$ images each, obtained by composing one of 50 shears, with one of 50 translations. The shears we used were always along the $x$-axis, i.e. of the form $(x, y) \in \mathbb{R}^2 \mapsto (x + \lambda y, y)$. The translations we used were either only along the $x$-axis or only along the $y$-axis.

Note that our shears commute with translations along the $x$-axis but not with those along the $y$-axis.

As expected, the red point-clouds, which represent the image of the blue ones under pooling, are contracted. In each figure from 3 to 6, one can see a green point: it is the origin of the space, i.e. the image with pixel value zero everywhere.

Note that as the blue point-cloud always has the same number of points as the red one, it is easy to see on these plots that the overall mass of the red point-clouds is closer to the origin than the blue mass is. As blue orbits get closer to zero after pooling, and see their variation reduced, their internal distances are contracted, which corresponds to a loss of information: as points get closer, they are progressively identified as being the same, thus loosing the information allowing to differentiate between them.

Identifying points in the same group orbit (w.r.t. $G$) as being the same corresponds to associating a progressively more and more ($G$-)invariant representation to each of these points.

Note that Figures 2.2 and 2.3 nicely illustrate what is described in Figure 2.1.
Figure 2.2: Illustration of orbits (first part).
2.3 NUMERICAL EXPERIMENTS: SHEARS AND TRANSLATIONS ON IMAGES

(a) Translations along $x$-axis and shears (other perspective).

(b) Translations along $y$-axis and shears.

(c) Translations along $y$-axis and shears (other perspective).

Figure 2.3: Illustration of orbits (second part).
In this subsection, we seek to qualitatively support the claim that the phenomenon of orbit contraction and flattening described in the previous sections correlates with what happens in practice with real data and standard architectures.

We train a 5-layers CNN for classification on the 10-classes image dataset CIFAR-10. Each of the six colors represents the translation orbit of an image of a different class, taken in the logit space, i.e. just before the softmax.

3D visualization was obtained via MDS. Note that the manifolds we plot are 2-dimensional, which supports the use of MDS despite the high dimension of the ambient space.

One can observe in Figure 2.4 that after only 10% of an epoch, classes start to disentangle (i.e. become more and more linearly separable). Moreover, translation orbits appear to be contracted (as can be seen on the scale) and locally flattened. Note that contraction in scale could in part also be due to weight regularization. Nonetheless, Figure 2.4 seems to support the claim that orbit contraction and flattening play a role in class disentanglement, and hence in solving classification.

2.4 CONCLUSIVE SUMMARY

Being able to disentangle highly tangled representations is a very important and challenging problem in machine learning. In deep learning in particular, there exist successful algorithms that may disentangle highly tangled representations in some situations, without us under-
standing why. Similarly, the ventral stream of the visual cortex in 
humans and primates seems to perform such a disentanglement of 
representations, but, again, the reasons behind this process are difficult 
to understand. It is believed that making representations invariant to 
some nuisance deformations, as well as locally flattening them, might 
help or even be an essential part of the disentangling process. As shown 
by our theorems, there is a connection between these two intuitions, in 
the sense that achieving a higher degree of invariance with respect to 
some group transformations will flatten the representations in direc-
tions of the tangent space corresponding to the Lie algebra generators 
of these transformations. We showed that in the case of the group of 
positive affine isometries, a precise bound on the sectional curvature 
can be computed, with respect to the pooling parameters. Numerical 
simulations supported the claim.

2.5 REMINDERS OF KNOWN RESULTS

The next three propositions are taken from the publicly available 
french textbook [Pau14], in which they’re respectively numbered as E.7, 
1.60, 1.62.

**Proposition 2.3.** Let $G$ be a Lie group and $\rho : G \to \text{GL}(V)$ a finite-
dimensional Lie group representation of $G$. Then for all $v \in V$, the map 
defined by $g \in G \mapsto \rho(g)v$ has constant rank, and the stabilizer $G_v$ is an 
embedded Lie subgroup of $G$.

**Proposition 2.4.** Let $G$ be a Lie group, $H$ be an embedded Lie subgroup of $G$, 
and $\pi : G \to G/H$ be the canonical projection. There exists one and only one 
smooth manifold structure on the topological quotient space $G/H$ turning $\pi$ 
into a smooth submersion. Moreover, the action of $G$ on $G/H$ is smooth, and 
if $H$ is normal in $G$, then $G/H$ is a Lie group, $\pi$ is a Lie group morphism, the 
Lie algebra $\mathfrak{h}$ of $H$ is an ideal of the Lie algebra $\mathfrak{g}$ of $G$ and the linear map from 
$T_eG/T_eH$ to $T_{eH}(G/H)$ induced by $T_e\pi$ is a Lie algebra isomorphism from 
$\mathfrak{g}/\mathfrak{h}$ to the Lie algebra of $G/H$.

**Proposition 2.5.** Let $M$ be a manifold together with a smooth action of a Lie 
group $G$, and $x \in M$; (i) the canonical map $\Theta_x : G/G_x \to M$ defined by
$\Theta_x(gG_x) = gx$ is a one-to-one immersion, whose image is the orbit $G \cdot x$; (ii) the orbit $G \cdot x$ is a submanifold of $M$ if and only if it is locally closed in $M$; (iii) if $G \cdot x$ is locally closed, then $\Theta_x$ is a diffeomorphism from $G / G_x$ to $G \cdot x$. 
Hyperbolic spaces have recently gained momentum in the context of machine learning due to their high capacity and tree-likeness properties. However, the representational power of hyperbolic geometry is not yet on par with Euclidean geometry, because of the absence of corresponding mathematical tools and statistical methods. This makes it hard to leverage hyperbolic geometry for data analysis and representation. In this chapter, we take a step towards bridging this gap in a principled manner by combining the formalism of Möbius gyrovector spaces with the Riemannian geometry of the Poincaré model of hyperbolic spaces. We support the choice of this geometry via the notions of Fisher Information and $\delta$-hyperbolicity. We start by generalizing the celebrated word embeddings method GloVe [PSM14] to metric spaces, specifying how to evaluate similarity, analogy and hypernymy of word embeddings in hyperbolic spaces. We then develop elementary operations in hyperbolic spaces akin to matrix multiplications and pointwise non-linearities, which enable us to generalize GRU [Chu+14]. Finally, we explain how to perform MLR in these spaces, also referred to as softmax layer, which is essential to a variety of neural architectures.

**Impact.** Some of our hyperbolic methods have been incorporated into the PyTorch package GeoOpt [KKK20]. Our contributions were successfully utilized by a broad variety of work, generalizing to hyperbolic spaces: graph neural networks [BBG20]; [Cha+19]; [LNK19]; [Zha+19], variational auto-encoders [Dai+20]; [Mat+19]; [Ovi19]; [SGB20], relational models [BAH19], normalizing flows [Bos+20] and others. The content of this chapter has in part been published at NeurIPS 2018 [GBH18c] and ICLR 2019 [TBG19], vulgarized in blogposts\(^1\) and was funded by the Max Planck ETH Center for Learning Systems while the author was at ETH Zürich.

\(^1\)http://hyperbolicdeeplearning.com
3.1 MATHEMATICAL PRELIMINARIES

We start by presenting the elementary notions of Riemannian geometry that are necessary to the exposition, before which we introduce the tools of hyperbolic geometry that we use in our models.

3.1.1 Reminders of Differential Geometry

For a deeper exposition, we refer the interested reader to [Spi79] and [HA10].

MANIFOLD. A manifold $M$ of dimension $n$ is a set that can be locally approximated by the Euclidean space $\mathbb{R}^n$. For instance, the sphere $S^2$ and the torus $T^2$ embedded in $\mathbb{R}^3$ are 2-dimensional manifolds, also called surfaces, as they can locally be approximated by $\mathbb{R}^2$. The notion of manifold is a generalization of the notion of surface.

TANGENT SPACE. For $x \in M$, the tangent space $T_xM$ of $M$ at $x$ is defined as the $n$-dimensional vector-space approximating $M$ around $x$ at a first order. It can be defined as the set of vectors $v \in \mathbb{R}^n$ that can be obtained as $v := c'(0)$, where $c : (-\epsilon, \epsilon) \to M$ is a smooth path in $M$ such that $c(0) = x$.

RIEMANNIAN METRIC. A Riemannian metric $g$ on $M$ is a collection $(g_x)_x$ of inner-products $g_x : T_xM \times T_xM \to \mathbb{R}$ on each tangent space $T_xM$, varying smoothly with $x$. Although it defines the geometry of $M$ locally, it induces a global distance function $d : M \times M \to \mathbb{R}_+$ by setting $d(x, y)$ to be the infimum of all lengths of smooth curves joining $x$ to $y$ in $M$, where the length $\ell$ of a curve $\gamma : [0, 1] \to M$ is defined by integrating the length of the speed vector living in the tangent space:

$$\ell(\gamma) = \int_0^1 \sqrt{g_{\gamma(t)}(\gamma'(t), \gamma'(t))} dt.$$  \hspace{1cm} (3.1)

RIEMANNIAN MANIFOLD. A smooth manifold $M$ equipped with a Riemannian metric $g$ is called a Riemannian manifold and is denoted by the pair $(M, g)$. Subsequently, due to their metric properties, we will only consider such manifolds.
3.1 Mathematical Preliminaries

Geodesics. A geodesic (straight line) between two points \( x, y \in \mathcal{M} \) is a smooth curve of minimal length joining \( x \) to \( y \) in \( \mathcal{M} \). Geodesics define shortest paths on the manifold. They are a generalization of lines in the Euclidean space.

Parallel Transport. In certain spaces, such as the hyperbolic space, there is a unique geodesic between two points, which allows to consider the parallel transport from \( x \) to \( y \) (implicitly taken along this unique geodesic) \( P_{x \rightarrow y} : T_x \mathcal{M} \to T_y \mathcal{M} \), which is a linear isometry between tangent spaces corresponding to moving tangent vectors along geodesics and defines a canonical way to connect tangent spaces.

Exponential Map. The exponential map \( \exp_x : T_x \mathcal{M} \to \mathcal{M} \) around \( x \), when well-defined, maps a small perturbation of \( x \) by a vector \( v \in T_x \mathcal{M} \) to a point \( \exp_x(v) \in \mathcal{M} \), such that \( t \in [0, 1] \mapsto \exp_x(tv) \) is a geodesic joining \( x \) to \( \exp_x(v) \). In Euclidean space, we simply have \( \exp_x(v) = x + v \). The exponential map is important, for instance, when performing gradient-descent over parameters lying in a manifold [Bon13]. For geodesically complete manifolds, such as the Poincaré ball considered in this work, \( \exp_x \) is well-defined on the full tangent space \( T_x \mathcal{M} \).

Conformality. A metric \( \tilde{g} \) on \( \mathcal{M} \) is said to be conformal to \( g \) if it defines the same angles, i.e. for all \( x \in \mathcal{M} \) and \( u, v \in T_x \mathcal{M} \setminus \{0\} \),

\[
\frac{\tilde{g}_x(u, v)}{\sqrt{\tilde{g}_x(u, u)} \sqrt{\tilde{g}_x(v, v)}} = \frac{g_x(u, v)}{\sqrt{g_x(u, u)} \sqrt{g_x(v, v)}}. \tag{3.2}
\]

This is equivalent to the existence of a smooth function \( \lambda : \mathcal{M} \to (0, \infty) \) such that \( \tilde{g}_x = \lambda^2 g_x \), which is called the conformal factor of \( \tilde{g} \) (w.r.t. \( g \)).

3.1.2 A Brief Introduction to Hyperbolic Geometry

The hyperbolic space of dimension \( n \geq 2 \) is a fundamental object in Riemannian geometry. It is (up to isometry) uniquely characterized as a complete, simply connected Riemannian manifold with constant negative curvature [Can+97]. The other two model spaces of constant sectional curvature are the flat Euclidean space \( \mathbb{R}^n \) (zero curvature) and the hyper-sphere \( \mathbb{S}^n \) (positive curvature).
The hyperbolic space has five models which are often insightful to work in. They are isometric to each other and conformal to the Euclidean space \([\text{Can}+97]; \text{Par}13\). We prefer to work in the Poincaré ball model \(D^n\) for the same reasons as \([\text{NK}17b]\).

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

\[
g^D_x = \lambda_x^2 g^E, \quad \text{where} \quad \lambda_x := \frac{2}{1 - \|x\|^2}, \quad (3.3)
\]

where \(g^E\) is the Euclidean metric tensor with components \(I_n\) of the standard space \(\mathbb{R}^n\), w.r.t. the usual Cartesian coordinates.

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 distorts distances.

**Induced distance and norm.** It is known \([\text{NK}17b]\) that the induced distance between 2 points \(x, y \in D^n\) is given by

\[
d_D(x, y) = \cosh^{-1} \left(1 + 2 \frac{\|x - y\|^2}{(1 - \|x\|^2) \cdot (1 - \|y\|^2)}\right). \quad (3.4)
\]

The Poincaré norm is then defined as:

\[
\|x\|_D := d_D(0, x) = 2 \tanh^{-1}(\|x\|). \quad (3.5)
\]

**Geodesics and exponential map.** Closed-form expression of the exponential map is obtained by first obtaining a closed form of unit-speed geodesics. We present these formulas below but refer the reader to \([\text{GBH}18b]\) for explicit computations. Let \(x \in D^n\) and \(v \in T_xD^n(= \mathbb{R}^n)\) such that \(g^D_x(v, v) = 1\). The unit-speed geodesic \(\gamma_{x, v} : \mathbb{R}_+ \rightarrow D^n\) with \(\gamma_{x, v}(0) = x\) and \(\dot{\gamma}_{x, v}(0) = v\) is given by

\[
\gamma_{x, v}(t) = \frac{(\lambda_x \cosh(t) + \lambda_x^2(x, v) \sinh(t)) x + \lambda_x \sinh(t)v}{1 + (\lambda_x - 1) \cosh(t) + \lambda_x^2(x, v) \sinh(t)}. \quad (3.6)
\]
As a consequence, one can then derive an explicit formula for the exponential map in the Poincaré ball. The exponential map at a point $x \in \mathbb{D}^n$, namely $\exp_x : T_x \mathbb{D}^n \to \mathbb{D}^n$, is given by

$$\exp_x(v) = \frac{\lambda_x \left( \cosh(\lambda_x \|v\|) + \langle x, \frac{v}{\|v\|} \rangle \sinh(\lambda_x \|v\|) \right)}{1 + (\lambda_x - 1) \cosh(\lambda_x \|v\|) + \lambda_x \langle x, \frac{v}{\|v\|} \rangle \sinh(\lambda_x \|v\|)} x$$

$$+ \frac{1}{\|v\|} \sinh(\lambda_x \|v\|) \cosh(\lambda_x \|v\|) + \lambda_x \langle x, \frac{v}{\|v\|} \rangle \sinh(\lambda_x \|v\|) v.$$

**Products.** We will also embed words in products of hyperbolic spaces, and explain why later on. A product of $p$ balls $(\mathbb{D}^n)^p$, with the induced product geometry, is known to have distance function

$$d_{(\mathbb{D}^n)^p}(x, y)^2 = \sum_{i=1}^p d_{\mathbb{D}^n}(x_i, y_i)^2.$$

**Half-plane.** Finally, another model of interest for us is the Poincaré half-plane $\mathbb{H}^2 = \mathbb{R} \times \mathbb{R}_+^*$ illustrated in fig. 3.1, with distance function

$$d_{\mathbb{H}^2}(x, y) = \cosh^{-1}\left(1 + \frac{\|x - y\|^2}{2x_2y_2}\right).$$

Figure 3.1 shows an isometry $\varphi$ from $\mathbb{D}^2$ to $\mathbb{H}^2$ mapping the vertical segment $\{0\} \times (-1, 1)$ to $\{0\} \times \mathbb{R}_+$, where $0 \in \mathbb{D}^2$ becomes $(0, 1) \in \mathbb{H}^2$.

![Isometric deformation \(\varphi\) of \(\mathbb{D}^2\) (left end) into \(\mathbb{H}^2\) (right end).](image)

Figure 3.1: Isometric deformation $\varphi$ of $\mathbb{D}^2$ (left end) into $\mathbb{H}^2$ (right end).

### 3.1.3 Riemannian Geometry of Gyrovector Spaces

In 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 [Albo08]; [Ung01]; [Ung08].

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.

We will make extensive use of these operations in our definitions of hyperbolic neural networks.

For \(c \geq 0\), denote \(\mathbb{D}^n_c := \{ x \in \mathbb{R}^n \mid c \|x\|^2 < 1 \}\). Note that if \(c = 0\), then \(\mathbb{D}^n_c = \mathbb{R}^n\); if \(c > 0\), then \(\mathbb{D}^n_c\) is the open ball of radius \(1/\sqrt{c}\). If \(c = 1\) then we recover the usual ball \(\mathbb{D}^n\). Note that for \(c, c' > 0\), \(\mathbb{D}^n_c\) and \(\mathbb{D}^n_{c'}\) are isometric.

**Möbius addition.** The Möbius addition of \(x\) and \(y\) in \(\mathbb{D}^n_c\) is defined as

\[
x \oplus_c y := \frac{(1 + 2c\langle x, y \rangle + c\|y\|^2)x + (1 - c\|x\|^2)y}{1 + 2c\langle x, y \rangle + c^2\|x\|^2\|y\|^2}.
\] (3.10)

In particular, when \(c = 0\), one recovers the Euclidean addition of two vectors in \(\mathbb{R}^n\). Note that without loss of generality, the case \(c > 0\) can be reduced to \(c = 1\). Unless stated otherwise, we will use \(\oplus\) as \(\oplus_1\) to simplify notations. For general \(c > 0\), this operation is not commutative nor associative. However, it satisfies \(x \oplus_c 0 = 0 \oplus_c x = x\). Moreover, for any \(x, y \in \mathbb{D}^n_c\), we have \((-x) \oplus_c x = x \oplus_c (-x) = 0\) and \((-x) \oplus_c (x \oplus_c y) = y\) (left-cancellation law). The Möbius substraction is then defined by the use of the following notation: \(x \ominus_c y := x \oplus_c (-y)\). See [Ver05, section 2.1] for a geometric interpretation of the Möbius addition.

**Möbius scalar multiplication.** For \(c > 0\), the Möbius scalar multiplication of \(x \in \mathbb{D}^n_c \setminus \{0\}\) by \(r \in \mathbb{R}\) is defined as

\[
r \otimes_c x := (1/\sqrt{c}) \tanh(r \tanh^{-1}(\sqrt{c}\|x\|)) \frac{x}{\|x\|},
\] (3.11)

and \(r \otimes_c 0 := 0\). Note that similarly as for the Möbius addition, one recovers the Euclidean scalar multiplication when \(c\) goes to zero: \(\lim_{c \to 0} r \otimes_c x = rx\). This operation satisfies desirable properties such

\^\text{2}\text{We take different notations as in [Ung01] where the author uses } s = 1/\sqrt{c}.\text{\text{\footnotesize\textsuperscript{2}}}
as \( n \otimes_c x = x \oplus_c \cdots \oplus_c x \) (\( n \) additions), \((r + r') \otimes_c x = r \otimes_c x \oplus_c r' \otimes_c x\) (scalar distributivity\(^3\)), \((rr') \otimes_c x = r \otimes_c (r' \otimes_c x)\) (scalar associativity) and \(|r| \otimes_c x/\|r \otimes_c x\| = x/\|x\|\) (scaling property).

**Distance.** If one defines the generalized hyperbolic metric tensor \( g^c \) as the metric conformal to the Euclidean one, with conformal factor \( \lambda^c_x = 2/(1-c\|x\|^2) \), then the induced distance function on \((D^n_c, g^c)\) is given by\(^4\)

\[
d_c(x, y) = (2/\sqrt{c}) \tanh^{-1} \left( \sqrt{c} \| x \oplus_c y \| \right).
\]

Again, observe that \( \lim_{c \to 0} d_c(x, y) = 2\|x - y\| \), \( \text{i.e.} \) we recover Euclidean geometry in the limit\(^5\). Moreover, for \( c = 1 \) we recover \( d_D \) of Eq. (3.4).

We now present how geodesics in the Poincaré ball model are usually described with Möbius operations, and push one step further the existing connection between gyrovector spaces and the Poincaré ball by finding new identities involving the exponential map, and parallel transport.

In particular, these findings provide us with a simpler formulation of Möbius scalar multiplication, yielding a natural definition of matrix-vector multiplication in the Poincaré ball.

**Riemannian gyroline element.** The Riemannian gyroline element is defined for an infinitesimal \( dx \) as \( ds := (x + dx) \ominus_c x \), and its size is given by [Ungo8, section 3.7]:

\[
\|ds\| = \|(x + dx) \ominus_c x\| = \|dx\|/(1 - c\|x\|^2).
\]

What is remarkable is that it turns out to be identical, up to a scaling factor of 2, to the usual line element \( 2\|dx\|/(1 - c\|x\|^2) \) of the Riemannian manifold \((D^n_c, g^c)\).

---

\(^3\)\( \otimes_c \) has priority over \( \oplus_c \) in the sense that \( a \otimes_c b \oplus c c := (a \otimes_c b) \oplus_c c \) and \( a \oplus_c (b \otimes_c c) \).

\(^4\)The notation \(-x \oplus_c y\) should always be read as \((−x) \oplus_c y\) and not \(- (x \oplus_c y)\).

\(^5\)The factor 2 comes from the conformal factor \( \lambda_x = 2/(1-\|x\|^2) \), which is a convention setting the curvature to \(-1\).
The geodesic connecting points \( x, y \in \mathbb{D}_c^n \) is shown in [Albo8]; [Ungo8] to be given by:

\[
\gamma_{x\rightarrow y}(t) := x \oplus_c ( - x \oplus_c y ) \otimes_c t,
\]

with \( \gamma_{x\rightarrow y} : \mathbb{R} \rightarrow \mathbb{D}_c^n \) s.t. \( \gamma_{x\rightarrow y}(0) = x \) and \( \gamma_{x\rightarrow y}(1) = y \). (3.14)

Note that when \( c \) goes to 0, geodesics become straight-lines, recovering Euclidean geometry. In the remainder of this subsection, we connect the gyrospace framework with Riemannian geometry.

**Lemma 3.1.** For any \( x \in \mathbb{D}_c^n \) and \( v \in T_x \mathbb{D}_c^n \) s.t. \( g_c^x(v,v) = 1 \), the unit-speed geodesic starting from \( x \) with direction \( v \) is given by:

\[
\gamma_{x,v}(t) = x \oplus_c \left( \tanh \left( \frac{\sqrt{c} t}{2} \frac{v}{\sqrt{c} \|v\|} \right) \right),
\]

where \( \gamma_{x,v} : \mathbb{R} \rightarrow \mathbb{D}_c^n \) s.t. \( \gamma_{x,v}(0) = x \) and \( \dot{\gamma}_{x,v}(0) = v \). (3.15)

**Proof.** One can use Eq. (3.14) and reparametrize it to unit-speed using Eq. (3.12). Alternatively, direct computation and identification with the formula in [GBH18b, Thm. 1] would give the same result. Using Eq. (3.12) and Eq. (3.15), one can sanity-check that \( d_c(\gamma(0), \gamma(t)) = t, \forall t \in [0,1] \). \( \square \)

**Exponential and Logarithmic Maps.** The following lemma gives the closed-form derivation of exponential and logarithmic maps.

**Lemma 3.2.** For any point \( x \in \mathbb{D}_c^n \), the exponential map \( \exp_c^x : T_x \mathbb{D}_c^n \rightarrow \mathbb{D}_c^n \) and the logarithmic map \( \log_c^x : \mathbb{D}_c^n \rightarrow T_x \mathbb{D}_c^n \) are given for \( v \neq 0 \) and \( y \neq x \) by:

\[
\exp_c^x(v) = x \oplus_c \left( \tanh \left( \frac{\sqrt{c} \lambda_c^x \|v\|}{2} \right) \frac{v}{\sqrt{c} \|v\|} \right),
\]

(3.16)

\[
\log_c^x(y) = \frac{2}{\sqrt{c} \lambda_c^x} \tanh^{-1} \left( \sqrt{c} \| - x \oplus_c y \| \right) \frac{-x \oplus_c y}{\| - x \oplus_c y \|}.
\]

(3.17)

**Proof.** Following the proof of [GBH18b, Cor. 1.1], one gets \( \exp_c^x(v) = \gamma_{x,v}(\sqrt{c} \lambda_c^x \|v\|) \). Using Eq. (3.15) gives the formula for \( \exp_c^x \). Algebraic check of the identity \( \log_c^x(\exp_c^x(v)) = v \) concludes. \( \square \)
The above maps have more appealing forms when $x = 0$, namely for $v \in T_0 \mathbb{D}_c^n \setminus \{0\}$, $y \in \mathbb{D}_c^n \setminus \{0\}$:

$$
\exp_c^0(v) = \tanh(\sqrt{c} \|v\|) \frac{v}{\sqrt{c} \|v\|}, \quad \log_c^0(y) = \tanh^{-1}(\sqrt{c} \|y\|) \frac{y}{\sqrt{c} \|y\|}.
$$

Moreover, we still recover Euclidean geometry in the limit $c \to 0$, as $\lim_{c \to 0} \exp_c^x(v) = x + v$ is the Euclidean exponential map, and $\lim_{c \to 0} \log_c^x(y) = y - x$ is the Euclidean logarithmic map.

**Möbius Scalar Multiplication Using Exponential and Logarithmic Maps.** We studied the exponential and logarithmic maps in order to gain a better understanding of the Möbius scalar multiplication (Eq. (3.11)). We found the following:

**Lemma 3.3.** The quantity $r \otimes x$ can actually be obtained by projecting $x$ in the tangent space at $0$ with the logarithmic map, multiplying this projection by the scalar $r$ in $T_0 \mathbb{D}_c^n$, and then projecting it back on the manifold with the exponential map:

$$
r \otimes_c x = \exp_c^0(r \log_c^0(x)), \quad \forall r \in \mathbb{R}, x \in \mathbb{D}_c^n.
$$

In addition, we recover the well-known relation between geodesics connecting two points and the exponential map:

$$
\gamma_{x \to y}(t) = x \oplus_c (-x \oplus_c y) \otimes_c t = \exp_c^x(t \log_c^x(y)), \quad t \in [0, 1].
$$

This last result enables us to generalize scalar multiplication in order to define matrix-vector multiplication between Poincaré balls, one of the essential building blocks of hyperbolic neural networks.

**Parallel Transport.** Finally, we connect parallel transport along the unique geodesic from $0$ to $x$ to gyrovector spaces with the following theorem.
In the manifold \((\mathbb{D}^n_c, g^c)\), the parallel transport w.r.t. the Levi-Civita connection of a vector \(v \in T_0 \mathbb{D}^n_c\) to another tangent space \(T_x \mathbb{D}^n_c\) is given by the following isometry:

\[
P^c_{0 \to x}(v) = \log^c_x (x \oplus_c \exp^c_0(v)) = \frac{\lambda^c_0}{\lambda^c_x} v. \tag{3.21}
\]

**Proof.** The geodesic in \(\mathbb{D}^n_c\) from \(0\) to \(x\) is given in Eq. (3.14) by \(\gamma(t) = x \otimes_c t\), for \(t \in [0, 1]\). Let \(v \in T_0 \mathbb{D}^n_c\). Then it is of common knowledge that there exists a unique parallel\(^6\) vector field \(X\) along \(\gamma\) (i.e. \(X(t) \in T_{\gamma(t)} \mathbb{D}^n_c\), \(\forall t \in [0, 1]\)) such that \(X(0) = v\). Let’s define:

\[
X : t \in [0, 1] \mapsto \log^c_{\gamma(t)} (\gamma(t) \oplus_c \exp^c_0(v)) \in T_{\gamma(t)} \mathbb{D}^n_c. \tag{3.22}
\]

Clearly, \(X\) is a vector field along \(\gamma\) such that \(X(0) = v\). Now define

\[
P^c_{0 \to x} : v \in T_0 \mathbb{D}^n_c \mapsto \log^c_x (x \oplus_c \exp^c_0(v)) \in T_x \mathbb{D}^n_c. \tag{3.23}
\]

From Eq. (3.17), it is easily seen that \(P^c_{0 \to x}(v) = \frac{\lambda^c_0}{\lambda^c_x} v\), hence \(P^c_{0 \to x}\) is a linear isometry from \(T_0 \mathbb{D}^n_c\) to \(T_x \mathbb{D}^n_c\). Since \(P^c_{0 \to x}(v) = X(1)\), it is enough to prove that \(X\) is parallel in order to guarantee that \(P^c_{0 \to x}\) is the parallel transport from \(T_0 \mathbb{D}^n_c\) to \(T_x \mathbb{D}^n_c\).

Since \(X\) is a vector field along \(\gamma\), its covariant derivative can be expressed with the Levi-Civita connection \(\nabla^c\) associated to \(g^c\):

\[
\frac{DX}{dt} = \nabla^c_{\dot{\gamma}(t)} X. \tag{3.24}
\]

Let’s compute the Levi-Civita connection from its Christoffel symbols. In a local coordinate system, they can be written as

\[
\Gamma^i_{jk} = \frac{1}{2} (g^c)^{il} (\partial_j g^c_{lk} + \partial_k g^c_{lj} - \partial_l g^c_{jk}), \tag{3.25}
\]

where superscripts denote the inverse metric tensor and using Einstein’s notations. As \(g^c_{ij} = (\lambda^c)^2 \delta_{ij}\), at \(\gamma(t) \in \mathbb{D}^n_c\) this yields:

\[
\Gamma^i_{jk} = c \lambda^c_{\gamma(t)} (\delta_{ik} \gamma(t)_j + \delta_{ij} \gamma(t)_k - \delta_{jk} \gamma(t)_i). \tag{3.26}
\]

\(^6\)i.e. that \(\frac{DX}{dt} = 0\) for \(t \in [0, 1]\), where \(\frac{D}{dt}\) denotes the covariant derivative.
On the other hand, since $X(t) = (\lambda_o^c / \lambda_{\gamma(t)}^c) u$, we have

$$\nabla_{\dot{\gamma}(t)}^c X = \dot{\gamma}(t)^i \nabla_i^c X = \dot{\gamma}(t)^i \nabla_i^c \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} v \right) = \nu^j \dot{\gamma}(t)^j \nabla_i^c \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} e_j \right).$$

(3.27)

Since $\gamma(t) = (1/\sqrt{c}) \tanh(t \tanh^{-1} (\sqrt{c} \|x\|)) \frac{x}{\|x\|}$, it is easily seen that $\dot{\gamma}(t)$ is colinear to $\gamma(t)$. Hence there exists $K_t^x \in \mathbb{R}$ such that $\dot{\gamma}(t) = K_t^x \gamma(t)$. Moreover, we have the following Leibniz rule:

$$\nabla_i^c \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} e_j \right) = \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} \nabla_i^c e_j + \frac{\partial}{\partial \gamma(t)_i} \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} \right) e_j.$$  

(3.28)

Combining these yields

$$\frac{DX}{dt} = K_t^x \nu^j \gamma(t)^i \nabla_i^c \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} \right) e_j + \frac{\partial}{\partial \gamma(t)_i} \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} \right) e_j.$$ 

(3.29)

Replacing with the Christoffel symbols of $\nabla^c$ at $\gamma(t)$ gives

$$\nabla_i^c e_j = \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} \Gamma_{ij}^k e_k = 2c[\delta_j^k \gamma(t)_i + \delta_i^k \gamma(t)_j - \delta_{ij} \gamma(t)^k] e_k.$$ 

(3.30)

Moreover,

$$\frac{\partial}{\partial \gamma(t)_i} \left( \frac{\lambda_o^c}{\lambda_{\gamma(t)}^c} \right) e_j = \frac{\partial}{\partial \gamma(t)_i} \left( -c \| \dot{\gamma}(t) \|^2 \right) e_j = -2c \dot{\gamma}(t)_i e_j.$$ 

(3.31)

Putting together everything, we obtain

$$\frac{DX}{dt} = K_t^x \nu^j \gamma(t)^i \left( 2c[\delta_j^k \gamma(t)_i + \delta_i^k \gamma(t)_j - \delta_{ij} \gamma(t)^k] e_k - 2c \dot{\gamma}(t)_i e_j \right)$$

(3.32)

$$= 2c K_t^x \nu^j \gamma(t)^i \left( \gamma(t)_j e_i - \delta_{ij} \gamma(t)^k e_k \right)$$

(3.33)

$$= 2c K_t^x \nu^j \left( \gamma(t)_j \gamma(t)^i e_i - \gamma(t)^i \delta_{ij} \gamma(t)^k e_k \right)$$

(3.34)

$$= 2c K_t^x \nu^j \left( \gamma(t)_j \gamma(t)^i e_i - \gamma(t)_j \gamma(t)^k e_k \right)$$

(3.35)

$$= 0,$$

(3.36)

which concludes the proof. \qed
As we’ll see later, this result is crucial in order to define and optimize parameters shared between different tangent spaces, such as biases in hyperbolic neural layers or parameters of hyperbolic \textit{MLR}.

Finally, another operator of interest is the so-called \textit{gyro} operator (see Eq.(1.27) of [Ungo8]), defined by:

\[\text{gyr}[u,v]w = \ominus (u \oplus v) \oplus \{u \oplus (v \oplus w)\} = w + 2\frac{Au + Bv}{D}, \quad (3.37)\]

where

\[A = -\langle u, w \rangle \|v\|^2 + \langle v, w \rangle + 2\langle u, v \rangle \cdot \langle v, w \rangle, \quad (3.38)\]
\[B = -\langle v, w \rangle \|u\|^2 - \langle u, w \rangle, \quad (3.39)\]
\[D = 1 + 2\langle u, v \rangle + \|u\|^2 \|v\|^2. \quad (3.40)\]

## 3.2 Word Embeddings

### 3.2.1 Adapting GloVe to Metric Spaces

\textbf{Euclidean GloVe.} The GloVe [PSM14] algorithm is an unsupervised method for learning word representations in the Euclidean space from statistics of word co-occurrences in a text corpus, with the aim to geometrically capture the words’ meaning and relations.

We use the notations: \(X_{ij}\) is the number of times word \(j\) occurs in the same window context as word \(i\); \(X_i = \sum_k X_{ik}\) is the word frequency; the embedding of a target word \(i\) is written \(\mathbf{w}_i\), while the embedding of a context word \(k\) is written \(\mathbf{w}_k\).

The initial formulation of the GloVe [PSM14] model suggests to learn embeddings as to math the word log-co-occurrence counts using vector dot-product together with biases:

\[\mathbf{w}_i^T \mathbf{w}_k + b_i + \tilde{b}_k = \log(X_{ik}). \quad (3.41)\]

The authors suggest to enforce this equality by optimizing a weighted least-square loss:

\[J = \sum_{i,j=1}^{V} f(X_{ij}) \left(\mathbf{w}_i^T \mathbf{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2, \quad (3.42)\]

where \(V\) is the size of the vocabulary and \(f\) down-weights the signal coming from frequent words (it is typically chosen to be \(f(x) = \min\{1, (x/x_m)^\alpha\}\), with \(\alpha = 3/4\) and \(x_m = 100\)).
How should one generalize GLOVE to non-Euclidean spaces such as the hyperbolic space? Note that there is no clear correspondence of the Euclidean inner-product in a hyperbolic space. However, we are provided with a distance function. Further notice that one could rewrite eq. (3.41) with the Euclidean distance as

$$-\frac{1}{2} \| \mathbf{w}_i - \tilde{\mathbf{w}}_k \|^2 + b_i + \tilde{b}_k = \log(X_{ik})$$

(3.43)

where we absorbed the squared norms of the embeddings into the biases. We thus replace the GLOVE loss by:

$$J = \sum_{i,j=1}^{V} f(X_{ij}) \left(-h(d(\mathbf{w}_i, \tilde{\mathbf{w}}_j)) + b_i + \tilde{b}_j - \log X_{ij} \right)^2,$$

(3.44)

where $h$ is a function to be chosen as a hyperparameter of the model, and $d$ can be any differentiable distance function. Although the most direct correspondence with GLOVE would suggest $h(x) = x^2/2$, we sometimes obtained better results with other functions, such as $h = \cosh^2$ (see section 3.2.6). Note that [De +18] also apply $\cosh$ to their distance matrix for hyperbolic MDS before applying Principal Component Analysis (PCA). Understanding why $h = \cosh^2$ is a good choice would be interesting future work.

3.2.2 Fisher Information and Hyperbolic Products

In order to endow Euclidean word embeddings with richer information, [VM15] proposed to represent words as Gaussians, i.e. with a mean vector and a covariance matrix\(^7\), expecting the variance parameters to capture how generic/specific a word is, and, hopefully, entailment relations. On the other hand, [NK17b] proposed to embed words of the WordNet hierarchy [Mil+90] in hyperbolic space, because this space is mathematically known to be better suited to embed tree-like graphs. It is hence natural to wonder: is there a connection between the two?

\(^7\text{diagonal or even spherical, for simplicity.}\)
the Fisher distance between two distributions relates to the hyperbolic
distance in $\mathbb{H}^2$:

$$d_F (N(\mu, \sigma^2), N(\mu', \sigma'^2)) = \sqrt{2} d_{\mathbb{H}^2} \left( \left( \mu / \sqrt{2}, \sigma \right), \left( \mu' / \sqrt{2}, \sigma' \right) \right). \quad (3.45)$$

For $n$-dimensional Gaussians with diagonal covariance matrices written
$\Sigma = \text{diag}(\sigma)^2$, it becomes:

$$d_F (N(\mu, \Sigma), N(\mu', \Sigma')) = \sqrt{\sum_{i=1}^{n} 2 d_{\mathbb{H}^2} \left( \left( \mu_i / \sqrt{2}, \sigma_i \right), \left( \mu'_i / \sqrt{2}, \sigma'_i \right) \right)^2}. \quad (3.46)$$

Hence there is a direct correspondence between diagonal Gaussians
and the product space $(\mathbb{H}^2)^n$.

This connection allows us to mathematically ground

- **word analogy** computations for Gaussian embeddings using hyper-
bolic geometry – section 3.2.4.2.

- **hypernymy detection** for hyperbolic embeddings using Gaussian
distributions – section 3.2.5.

**Fisher distance, KL & Gaussian embeddings.** The above para-
graph lets us relate the word2gauss algorithm [VM15] to hyperbolic
word embeddings. Although one could object that word2gauss is
trained using a KL divergence, while hyperbolic embeddings relate to
Gaussian distributions via the Fisher distance $d_F$, let us remind that KL
and $d_F$ define the same local geometry. Indeed, the KL is known to be
related to $d_F$, as its local approximation [Jef46].

**Riemannian optimization.** A benefit of representing words in
(products of) hyperbolic spaces, as opposed to (diagonal) Gaussian dis-
tributions, is that one can use recent Riemannian adaptive optimization
tools such as Radagrad [BG19]. Note that without this connection, it
would be unclear how to define a variant of Adamgrad [DHS11] intrinsic
to the statistical manifold of Gaussians. Empirically, we indeed noticed
better results using Radagrad, rather than simply Riemannian SGD
[Bon13]. Similarly, note that Glove trains with Adagrad.
3.2.3 Training Details

Experimental setup. We trained all models on a corpus provided by [LG14], [LGD15] used in other word embeddings related work. Corpus preprocessing is explained in the above references. The dataset has been obtained from an English Wikipedia dump and contains 1.4 billion tokens. Words appearing less than one hundred times in the corpus have been discarded, leaving 189,533 unique tokens. The co-occurrence matrix contains approximately 700 millions non-zero entries, for a symmetric window size of 10. All models were trained for 50 epochs, and unless stated otherwise, on the full corpus of 189,533 word types. For certain experiments, we also trained the model on a restricted vocabulary of the 50,000 most frequent words, which we specify by appending either “(190k)” or “(50k)” to the experiment’s name in the table of results.

Poincaré models, Euclidean baselines. We report results for both 100D embeddings trained in a 100D Poincaré ball, and for 50x2D embeddings, which were trained in the Cartesian product of 50 2D Poincaré balls. Note that in the case of both models, one word will be represented by exactly 100 parameters. For the Poincaré models we employ both $h(x) = x^2$ and $h(x) = \cosh^2(x)$. All hyperbolic models were optimized with RADAGRAD [BG19] as explained in section 3.2.2. On the tasks of similarity and analogy, we compare against a 100D Euclidean GloVe model which was trained using the hyperparameters suggested in the original GloVe paper [PSM14]. The vanilla GloVe model was optimized using ADAGRAD [DHS11]. For the Euclidean baseline as well as for models with $h(x) = x^2$ we used a learning rate of 0.05. For Poincaré models with $h(x) = \cosh^2(x)$ we used a learning rate of 0.01.

The initialization trick. We obtained improvement in the majority of the metrics when initializing our Poincaré model with pre-trained parameters. These were obtained by first training the same model on the restricted (50k) vocabulary, and then using this model as an initialization for the full (190K) vocabulary. This will be referred to as the “initialization trick”. For fairness, we also trained the vanilla (Euclidean) GloVe model in the same fashion.
3.2.4 Similarity & Analogy

3.2.4.1 Similarity

**Quantitative results.** Word similarity is assessed using a number of well established benchmarks shown in table 3.1. We summarize here our main results, but more extensive experiments (including in lower dimensions) are shown in [TBG19]. We note that, with a single exception, our 100D and 50x2D models outperform the vanilla Glove baselines in all settings.

**Qualitative results.** Nearest neighbors w.r.t the Poincaré distance are shown in Table 3.2.

Table 3.1: Word similarity results for 100-dimensional models. Highlighted: the best and the 2nd best. Implementation of these experiments was done in collaboration with co-authors in [TBG19].

<table>
<thead>
<tr>
<th>Experiment name</th>
<th>Rare Word SimLex SimVerb MC RG</th>
</tr>
</thead>
<tbody>
<tr>
<td>100D Vanilla GloVe</td>
<td>0.3798 0.2963 0.1675 0.6524 0.6894</td>
</tr>
<tr>
<td>100D Vanilla GloVe w/ init trick</td>
<td>0.3787 0.2964 0.1639 0.6562 0.6757</td>
</tr>
<tr>
<td>100D Poincaré GloVe ( h(x) = \cosh^2(x) ), w/ init trick</td>
<td>0.4187 0.3208 0.1915 0.7833 0.7578</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe ( h(x) = \cosh^2(x) ), w/ init trick</td>
<td>0.4276 0.3181 0.189 0.8046 0.7597</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe ( h(x) = x^2 ), w/ init trick</td>
<td>0.4104 0.3022 0.1685 0.7655 0.728</td>
</tr>
</tbody>
</table>

3.2.4.2 Analogy

A common task used to evaluate word embeddings, called analogy, consists in finding which word \( d \) is to the word \( c \), what the word \( b \) is to the word \( a \). For instance, queen is to woman what king is to man. In
Table 3.2: Nearest neighbors (in terms of Poincaré distance) for some words using our 100D hyperbolic embedding model.

<table>
<thead>
<tr>
<th>Word</th>
<th>Nearest neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>sixties</td>
<td>seventies, eighties, nineties, 60s, 70s, 1960s, 80s, 90s, 1980s, 1970s</td>
</tr>
<tr>
<td>dance</td>
<td>dancing, dances, music, singing, musical, performing, hip-hop, pop, folk, dancers</td>
</tr>
<tr>
<td>daughter</td>
<td>wife, married, mother, cousin, son, niece, granddaughter, husband, sister, eldest</td>
</tr>
<tr>
<td>vapor</td>
<td>vapour, refrigerant, liquid, condenses, supercooled, fluid, gaseous, gases, droplet</td>
</tr>
<tr>
<td>ronaldo</td>
<td>cristiano, ronaldinho, rivaldo, messi, zidane, romario, pele, zinedine, xavi, roboinho</td>
</tr>
<tr>
<td>mechanic</td>
<td>electrician, fireman, machinist, welder, technician, builder, janitor, trainer, brakeman</td>
</tr>
<tr>
<td>algebra</td>
<td>algebras, homological, heyting, geometry, subalgebra, quaternion, calculus, mathematics, unital, algebraic</td>
</tr>
</tbody>
</table>

the Euclidean embedding space, the solution to this problem is usually taken geometrically as $d = c + (b - a) = b + (c - a)$. Note that this implies that the same $d$ is also to $b$, what $c$ is to $a$.

How should one intrinsically define “analogy parallelograms” in a space of Gaussian distributions? Note that [VM15] do not evaluate their Gaussian embeddings on the analogy task, and that it would be unclear how to do so. However, since we can go back and forth between (diagonal) Gaussians and (products of) hyperbolic spaces as explained in section 3.2.2, we can use the fact that parallelograms are naturally defined in the Poincaré ball, by the notion of gyro-translation [Ung12, section 4]. In the Poincaré ball, the two solutions $d_1 = c + (b - a)$ and $d_2 = b + (c - a)$ are respectively generalized to

$$d_1 = c \oplus gyr[c, \ominus a](\ominus a \oplus b), \quad \text{and} \quad d_2 = b \oplus gyr[b, \ominus a](\ominus a \oplus c).$$

(3.47)

where the $gyr$ operator as well as the closed-form formulas for these operations are described in Eq. (3.37), being easy to implement. The fact that $d_1$ and $d_2$ differ is due to the curvature of the space. For evaluation, we chose a point $m_{d_1d_2}^t := d_1 \oplus ((-d_1 \oplus d_2) \otimes t)$ located
on the geodesic between \( d_1 \) and \( d_2 \) for some \( t \in [0, 1] \); if \( t = 1/2 \), this is called the gyro-midpoint and then \( m_{d_1 d_2}^{0.5} = m_{d_2 d_1}^{0.5} \), which is at equal hyperbolic distance from \( d_1 \) as from \( d_2 \). We select \( t \) based on 2-fold cross-validation, as explained in [TBG19].

Note that continuously deforming the Poincaré ball to the Euclidean space (by sending its radius to infinity) lets these analogy computations recover their Euclidean counterparts, which is a nice sanity check. Indeed, one can rewrite eq. (3.47) with tools from differential geometry as

\[
c \oplus \text{gyr}[c, \ominus a](\ominus a \oplus b) = \exp_c(P_{a \rightarrow c}(\log a(b))),
\]

where \( P_{x \rightarrow y} = (\lambda_x / \lambda_y)\text{gyr}[y, \ominus x] \) denotes the parallel transport along the unique geodesic from \( x \) to \( y \). The exp and log maps of Riemannian geometry are related to the theory of gyrovector spaces as mentioned in section 3.1.3. We also mention again that, when continuously deforming the hyperbolic space \( \mathbb{D}^n \) into the Euclidean space \( \mathbb{R}^n \), sending its curvature \( \kappa \) from \(-1\) to 0 (i.e. the radius of \( \mathbb{D}^n \) from 1 to \( \infty \)), the Möbius operations \( \oplus^\kappa, \ominus^\kappa, \otimes^\kappa, \text{gyr}^\kappa \) recover their respective Euclidean counterparts \(+, -, \cdot, Id\). Hence, the analogy solutions \( d_1, d_2, m_{d_1 d_2}^t \) of eq. (3.47) would then all recover the Euclidean formulation \( d = c + b - a \).

**Evaluation.** For word analogy, we evaluate on the Google benchmark [Mik+13a] and its two splits that contain semantic and syntactic analogy queries. We also use a benchmark by MSR that is also commonly employed in other word embedding works. For the Euclidean baselines we use 3COSADD [LGD15]. For our models, the solution \( d \) to the problem “which \( d \) is to \( c \), what \( b \) is to \( a \)” is selected as \( m_{d_1 d_2}^t \) as described in section 3.2.4.2.

In order to select the best \( t \) without overfitting on the benchmark dataset, we used the same 2-fold cross-validation method used by [LGD15, section 5.1], which resulted in selecting \( t = 0.3 \). We report our main results in table 3.3. More extensive experiments in various settings (including in lower dimensions) are shown in [TBG19]. We note that the vast majority of our models outperform the vanilla Glove baselines, with the 100D hyperbolic embeddings being the absolute best.
Table 3.3: Word analogy results for 100-dimensional models on the Google and MSR datasets. Highlighted: the best and the 2nd best. Implementation of these experiments was done in collaboration with co-authors in [TBG19].

<table>
<thead>
<tr>
<th>Experiment name</th>
<th>Method</th>
<th>SemG</th>
<th>SynG</th>
<th>G</th>
<th>MSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>100D Vanilla GloVe</td>
<td>3COSADD</td>
<td>0.6005</td>
<td>0.5869</td>
<td>0.5931</td>
<td>0.4868</td>
</tr>
<tr>
<td>100D Vanilla GloVe w/ init trick</td>
<td>3COSADD</td>
<td>0.6427</td>
<td>0.595</td>
<td>0.6167</td>
<td>0.4826</td>
</tr>
<tr>
<td>100D Poincaré GloVe h(x) = cosh²(x) w/ init. trick</td>
<td>Cosine dist</td>
<td>0.6641</td>
<td>0.6088</td>
<td>0.6339</td>
<td>0.4971</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe h(x) = x² w/ init. trick</td>
<td>Poincaré dist</td>
<td>0.6582</td>
<td>0.6066</td>
<td>0.6300</td>
<td>0.4672</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe h(x) = cosh²(x) w/ init. trick</td>
<td>Poincaré dist</td>
<td>0.6048</td>
<td>0.6042</td>
<td>0.6045</td>
<td>0.4849</td>
</tr>
</tbody>
</table>

3.2.5 Hypernymy: A New Way of Assessing Entailment

We are interested in using our word embeddings to address another task, namely hypernymy detection, i.e. to predict relations of type is-a(v,w) such as is-a(dog, animal).

As a strong first baseline, we focus on the method of [NK17b] that uses an heuristic entailment score in order to predict whether u is-a v, defined in terms of their embeddings u, v ∈ D^n as

\[
is-a(u, v) := -(1 + a(\|v\|_2 - \|u\|_2))d(u, v)\]  

(3.49)

This choice is based on the intuition that the Euclidean norm should encode generality/specificity of a concept/word. However, such a choice depends on the parametrization and origin of the hyperbolic space, which is problematic when the word embedding training loss involves only the distance function.

A second baseline is that of Gaussian word embeddings [VM15] in which words are modeled as normal probability distributions. The authors propose using the entailment score is-a(P, Q) := −KL(P∥Q) for continuous distributions P and Q representing two words.
argument is that a low $KL(P||Q)$ indicates that we can encode $Q$ easily as $P$, implying that $Q$ entails $P$. However, we would like to mitigate this statement. Indeed, if $P = \mathcal{N}(\mu, \sigma)$ and $Q = \mathcal{N}(\mu, \sigma')$ are two 1D Gaussian distributions with same mean, then $KL(P||Q) = z^2 - 1 - \log(z)$ where $z := \sigma / \sigma'$, which is not a monotonic function of $z$. This breaks the idea that the magnitude of the variance should encode the generality/specificity of the concept.

Our contribution consists in using the connection detailed in section 3.2.2 in order to introduce a novel principled score that can be applied on top of our (unsupervised) learned Poincaré Glove embeddings to address the task of hypernymy detection. This hypernymy score can leverage either fully unsupervised information (i.e. word frequencies) or weakly-supervised (i.e. WordNet tree levels) information.

**INVARIA NCE TO ISOMETRIC TRANSFORMATIONS OF DISTANCE BASED EMBEDDINGS.** The method of [NK17b] uses a heuristic entailment score in order to predict whether $u$ is-a $v$, defined for $u, v \in \mathbb{D}^n$ as

$$\text{is-a}(u, v) := -(1 + \alpha(\|v\|_2 - \|u\|_2))d(u, v) \tag{3.50}$$

This choice is based on the intuition that the Euclidean norm should encode generality/specificity of a concept/word. However, such a choice is not intrinsic to the hyperbolic space when the training loss involves only the distance function. We say that training is intrinsic to $\mathbb{D}^n$, i.e. invariant to applying any isometric transformation $\varphi : \mathbb{D}^n \rightarrow \mathbb{D}^n$ to all word embeddings (such as hyperbolic translation). But their “is-a” score is not intrinsic, i.e. depends on the parametrization. For this reason, we argue that an isometry has to be found and fixed before using the trained word embeddings in any non-intrinsic manner, e.g. to define hypernymy scores. To discover this transformation, we leverage the connection between hyperbolic and Gaussian embeddings as follows.

**HYPERBOLIC TO GAUSSIAN EMBEDDINGS VIA AN ISOMETRY.** We argue that generality of a concept as being embedded in a Poincaré ball is better captured by a direction in hyperbolic space, i.e. by a geodesic, rather than by the distance from the origin. Why? For a 1D Gaussian $\mathcal{N}(\mu, \sigma^2)$ representing a concept, generality should be naturally encoded in the magnitude of $\sigma$ [VM15]. As shown in section 3.2.2, the space of Gaussians endorsed with the Fisher distance is naturally mapped to the hyperbolic upper half-plane $\mathbb{H}^2$, where the variance
\[\sigma\] corresponds to the (positive) second coordinate in \(H^2 = \mathbb{R} \times \mathbb{R}_+^*\). Moreover, \(H^2\) can be isometrically mapped to \(D^2\), where the second coordinate \(\sigma \in \mathbb{R}_+^*\) corresponds to the open vertical segment \(\{0\} \times (-1,1)\) in \(D^2\). However, in \(D^2\), any (hyperbolic) translation or any rotation w.r.t. the origin is an isometry.\(^8\) Hence, in order to map a word \(x \in D^2\) to a Gaussian \(\mathcal{N}(\mu, \sigma^2)\) via \(H^2\), we first have to find this hidden alignment isometry. This transformation should align \(\{0\} \times (-1,1) \subset H^2\) with the (unknown) geodesic in \(D^2\) that encodes generality. For simplicity, we assume it is composed of a centering and a rotation operations in \(D^2\). As a consequence, we start by identifying two sets \(G\) and \(S\) of potentially generic and specific words, respectively. For the re-centering, we then compute the means \(g\) and \(s\) of \(G\) and \(S\) respectively, and \(m := (s + g) / 2\), and Möbius translate all words by the global mean with the operation \(w \mapsto \Theta m \oplus w\). For the rotation, we set \(u := (\Theta m \oplus g) / \| \Theta m \oplus g \|_2\), and rotate all words so that \(u\) is mapped to \((0,1)\). Figure 3.2 and Algorithm 1 illustrate these steps. In order to identify the two sets \(G\) and \(S\),

Figure 3.2: We show here one of the \(D^2\) spaces of 20D word embeddings embedded in \((D^2)^10\) with our unsupervised hyperbolic Glove algorithm. This illustrates the three steps of applying the isometry. From left to right: the trained embeddings, raw; then after centering; then after rotation; finally after isometrically mapping them to \(H^2\). The isometry was obtained with the weakly-supervised method WordNet 400 + 400. Legend: WordNet levels (root is 0). Model: \(h = (\cdot)^2\), full vocabulary of 190k words. More of these plots for other \(D^2\) spaces are shown in [TBG19].

we propose the following two methods.

- **Unsupervised 5K+5K**: a fully unsupervised method. We first define a restricted vocabulary of the 50k most frequent words among

\(^8\)See [http://bulatov.org/math/1001](http://bulatov.org/math/1001) for intuitive animations describing hyperbolic isometries.
the unrestricted one of 190k words, and rank them by frequency; we then define $G$ as the 5k most frequent ones, and $S$ as the 5k least frequent ones of the 50k vocabulary (to avoid extremely rare words which might have received less signal during training).

• **Weakly-supervised WN $x+x$:** a weakly-supervised method that uses words from the WordNet hierarchy. We define $G$ as the top $x$ words from the top 4 levels of the WordNet hierarchy, and $S$ as $x$ of the bottom words from the bottom 3 levels, randomly sampled in case of ties.

**Gaussian embeddings.** [VM15] propose using the entailment score $\text{is-a}(P, Q) := -KL(P \| Q)$ for distributions $P$ and $Q$ representing two words. Their argument for this choice is that a low $KL(P \| Q)$ indicates that we can encode $Q$ easily as $P$, implying that $Q$ entails $P$. However, we would like to mitigate this statement. Indeed, if $P = \mathcal{N}(\mu, \sigma)$ and $Q = \mathcal{N}(\mu, \sigma')$ are two 1D Gaussian distributions with same mean, then $KL(P \| Q) = z^2 - 1 - \log(z)$ where $z := \sigma / \sigma'$, which is not a monotonic function of $z$. *This breaks the idea that the magnitude of the variance should encode the generality/specificity of the concept.*

**A new entailment score for Gaussian embeddings.** What would constitute a good number for the variance’s magnitude of a $n$-dimensional Gaussian distribution $\mathcal{N}(\mu, \Sigma)$? It is known that 95% of its mass is contained within a hyper-ellipsoid of volume $V_{\Sigma} = V_n \sqrt{\det(\Sigma)}$, where $V_n$ denotes the volume of a ball of radius 1 in $\mathbb{R}^n$. For simplicity, we propose dropping the dependence in $\mu$ and define a simple score $\text{is-a}(\Sigma, \Sigma') := \log(V_{\Sigma'}) - \log(V_{\Sigma}) = \sum_{i=1}^{n} (\log(\sigma'_i) - \log(\sigma_i))$. Note that using difference of logarithms has the benefit of removing the scaling constant $V_n$, and makes the entailment score invariant to a rescaling of the covariance matrices: $\text{is-a}(r\Sigma, r\Sigma') = \text{is-a}(\Sigma, \Sigma'), \forall r > 0$. To compute this is-a score between two hyperbolic word embeddings, we first map all word embeddings to Gaussians as explained above and, subsequently, apply the above proposed is-a score. Algorithm 1 illustrates these steps. Results are shown in Figure 3.4 and Tables 3.6, 3.5.
Algorithm 1 is-a(v, w) hypernymy score using Poincaré embeddings

1: \textbf{procedure} \textsc{is-a-score}(v, w)
2: \hspace{1em} \textbf{Input}: \( v, w \in (\mathbb{D}^2)^p, v := [v_1, ..., v_p], w := [w_1, ..., w_p], v_i, w_i \in \mathbb{D}^2 \)
3: \hspace{1em} \textbf{Output}: is-a(v, w) lexical entailment score
4: \hspace{1em} \mathcal{G} \leftarrow \text{set of Poincaré embeddings of generic words}
5: \hspace{1em} \mathcal{S} \leftarrow \text{set of Poincaré embeddings of specific words}
6: \hspace{1em} \text{for } i \text{ from } 1 \text{ to } p \text{ do}
7: \hspace{2em} g_i \leftarrow \text{mean}\{x_i | x \in \mathcal{G}\} \quad \text{// Euclidean mean of generic words}
8: \hspace{2em} s_i \leftarrow \text{mean}\{y_i | y \in \mathcal{S}\} \quad \text{// Euclidean mean of specific words}
9: \hspace{2em} m_i \leftarrow (g_i + s_i)/2 \quad \text{// Total mean}
10: \hspace{2em} v'_i \leftarrow \ominus m_i \oplus v_i \quad \text{// Möbius translation by } m_i
11: \hspace{2em} w'_i \leftarrow \ominus m_i \oplus w_i \quad \text{// Möbius translation by } m_i
12: \hspace{2em} u_i \leftarrow (\ominus m_i \oplus g_i)/|| \ominus m_i \oplus g_i||_2 \quad \text{// Compute rotation vector}
13: \hspace{2em} v''_i \leftarrow \text{rotate}(v'_i, u_i) \quad \text{// rotate } v'_i
14: \hspace{2em} w''_i \leftarrow \text{rotate}(w'_i, u_i) \quad \text{// rotate } w'_i
15: \hspace{2em} \tilde{v}_i \leftarrow \text{poincare2halfplane}(v''_i) \quad \text{// Convert from Poincaré disk coordinates to half-plane coordinates.}
16: \hspace{2em} \tilde{w}_i \leftarrow \text{poincare2halfplane}(w''_i)
17: \hspace{2em} \mu^p_i \leftarrow \tilde{v}_{i1}/\sqrt{2}; \sigma^p_i \leftarrow \tilde{v}_{i2}
18: \hspace{2em} \mu^w_i \leftarrow \tilde{w}_{i1}/\sqrt{2}; \sigma^w_i \leftarrow \tilde{w}_{i2}
\text{return } \sum_{i=0}^p (\log(\sigma^p_i) - \log(\sigma^w_i))
**Hyperbolic spaces: embeddings & classification**

Figure 3.3: Different hierarchies captured by a 10x2D model with $h(x) = x^2$, in some selected 2D half-planes. The $y$ coordinate encodes the magnitude of the variance of the corresponding Gaussian embeddings, representing word generality/specificity. Thus, this type of Nx2D models offer an amount of interpretability.

**Hypernymy evaluation.** For hypernymy evaluation we use the Hyperlex [Vul+17] and WBLess (subset of BLESS) [BL11] datasets. We classify all the methods in three categories depending on the supervision used for word embedding learning and for the hypernymy score, respectively. For Hyperlex we report results in table 3.6 and use the baseline scores reported in [NK17b]; [Vul+17]. For WBLess we report results in table 3.5 and use the baseline scores reported in [Ngu+17]. Implementation of these experiments was done in collaboration with co-authors in [TBG19].

Table 3.4: Some words selected from the 100 nearest neighbors and ordered according to the hypernymy score function for a 50x2D hyperbolic embedding model using $h(x) = x^2$.

| **reptile** | amphibians, carnivore, crocodilian, fish-like, dinosaur, alligator, triceratops |
| **algebra** | mathematics, geometry, topology, relational, invertible, endomorphisms, quaternions |
| **music**  | performance, composition, contemporary, rock, jazz, electroacoustic, trio |
| **feeling** | sense, perception, thoughts, impression, emotion, fear, shame, sorrow, joy |
3.2 Word Embeddings

Hypernymy results discussion. We first note that our fully unsupervised 50x2D, \( h(x) = x^2 \) model outperforms all its corresponding baselines setting a new state-of-the-art on unsupervised WBLESS accuracy and matching the previous state-of-the-art on unsupervised HyperLex Spearman correlation.

Second, once a small amount of weakly supervision is used for the hypernymy score, we obtain significant improvements as shown in the same tables. We note that this weak supervision is only as a post-processing step (after word embeddings are trained). Moreover, it does not consist of hypernymy pairs, but only of 400 or 800 generic and specific sets of words from WordNet. Even so, our unsupervised learned embeddings are remarkably able to outperform all (except WN-Poincaré) supervised embedding learning baselines on HyperLex which have the great advantage of using the hypernymy pairs to train the word embeddings.

Figure 3.4: Left: Gaussian variances of our learned hyperbolic embeddings (trained unsupervised on co-occurrence statistics, isometry found with “Unsupervised 1k+1k”) correlate with WordNet levels; Right: performance of our embeddings on hypernymy (HyperLex dataset) evolve when we increase the amount of supervision \( x \) used to find the correct isometry in the model WN \( x + x \). As can be seen, a very small amount of supervision (e.g. 20 words from WordNet) can significantly boost performance compared to fully unsupervised methods.
Table 3.5: WBLESS results in terms of accuracy for 3 different model types ordered according to their difficulty.

<table>
<thead>
<tr>
<th>Method</th>
<th>Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Wee+14]</td>
<td>0.75</td>
</tr>
<tr>
<td>WN-Poincaré from [NK17b]</td>
<td>0.86</td>
</tr>
<tr>
<td>[Ngu+17]</td>
<td>0.87</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>i) Supervised embedding learning, Unsupervised hypernym score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order Embeddings [Ven+15]</td>
<td>0.191</td>
</tr>
<tr>
<td>PARAGRAM + CF</td>
<td>0.320</td>
</tr>
<tr>
<td>WN-Basic</td>
<td>0.240</td>
</tr>
<tr>
<td>WN-WuP</td>
<td>0.214</td>
</tr>
<tr>
<td>WN-LCh</td>
<td>0.214</td>
</tr>
<tr>
<td>WN-Eucl from [NK17b]</td>
<td>0.389</td>
</tr>
<tr>
<td>WN-Poincaré from [NK17b]</td>
<td>0.512</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>ii) Unsupervised embedding learning, Weakly-supervised hypernym score</th>
</tr>
</thead>
<tbody>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = \cosh^2(x)$, init trick (190k)</td>
<td>0.749</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = x^2$, init trick (190k)</td>
<td>0.790</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>iii) Unsupervised embedding learning, Unsupervised hypernym score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGNS from [Ngu+17]</td>
<td>0.48</td>
</tr>
<tr>
<td>[Wee+14]</td>
<td>0.58</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = \cosh^2(x)$, init trick (190k)</td>
<td>0.575</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = x^2$, init trick (190k)</td>
<td>0.652</td>
</tr>
</tbody>
</table>

Table 3.6: Hyperlex results in terms of Spearman correlation for 3 different model types ordered according to their difficulty.

<table>
<thead>
<tr>
<th>Method</th>
<th>$\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>i) Supervised embedding learning, Unsupervised hypernym score</td>
<td></td>
</tr>
<tr>
<td>Order Embeddings [Ven+15]</td>
<td>0.191</td>
</tr>
<tr>
<td>PARAGRAM + CF</td>
<td>0.320</td>
</tr>
<tr>
<td>WN-Basic</td>
<td>0.240</td>
</tr>
<tr>
<td>WN-WuP</td>
<td>0.214</td>
</tr>
<tr>
<td>WN-LCh</td>
<td>0.214</td>
</tr>
<tr>
<td>WN-Eucl from [NK17b]</td>
<td>0.389</td>
</tr>
<tr>
<td>WN-Poincaré from [NK17b]</td>
<td>0.512</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>ii) Unsupervised embedding learning, Weakly-supervised hypernym score</th>
</tr>
</thead>
<tbody>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = \cosh^2(x)$, init trick (190k)</td>
<td>0.402</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = x^2$, init trick (190k)</td>
<td>0.421</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>iii) Unsupervised embedding learning, Unsupervised hypernym score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word2Gauss-DistPos</td>
<td>0.206</td>
</tr>
<tr>
<td>SGNS-Deps</td>
<td>0.205</td>
</tr>
<tr>
<td>Frequency</td>
<td>0.279</td>
</tr>
<tr>
<td>SLQS-Slim</td>
<td>0.229</td>
</tr>
<tr>
<td>Vis-ID</td>
<td>0.253</td>
</tr>
<tr>
<td>DIVE-WA$S$ [Cha+18]</td>
<td>0.333</td>
</tr>
<tr>
<td>SBOW-PPMI-C$\Delta$S from [Cha+18]</td>
<td>0.345</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe $h(x) = \cosh^2(x)$ init trick (190k)</td>
<td>0.284</td>
</tr>
<tr>
<td>50x2D Poincaré GloVe, $h(x) = x^2$, init trick (190k)</td>
<td>0.341</td>
</tr>
</tbody>
</table>
3.2.6 Motivating Hyperbolic GloVe via $\delta$-hyperbolicity

Why would we embed words in a hyperbolic space? Given some symbolic data, such as a vocabulary along with similarity measures between words — in our case, co-occurrence counts $X_{ij}$ — can we understand in a principled manner which geometry would represent it best? Choosing the right metric space to embed words can be understood as selecting the right inductive bias — an essential step.

$\delta$-hyperbolicity. A particular quantity of interest describing qualitative aspects of metric spaces is the $\delta$-hyperbolicity. This metric introduced by Gromov [Gro87] quantifies the tree-likeness of a metric space. Formally, the hyperbolicity $\delta(x, y, z, t)$ of a 4-tuple $(x, y, z, t)$ is defined as half the difference between the biggest two of the following sums: $d(x, y) + d(z, t)$, $d(x, z) + d(y, t)$, $d(x, t) + d(y, z)$. The $\delta$-hyperbolicity of a metric space is defined as the supremum of these numbers over all 4-tuples. Following [ADM14], we will denote this number by $\delta_{\text{worst}}$ (it is a worst-case measure), and by $\delta_{\text{avg}}$ the average of these over all 4-tuples, when the space is a finite set. Intuitively, a low $\delta_{\text{avg}}$ of a finite metric space characterizes that this space has an underlying hyperbolic geometry, i.e. an approximate tree-like structure, and that the hyperbolic space would be well suited to isometrically embed it. We also report the ratio $2 \ast \delta_{\text{avg}} / d_{\text{avg}}$ (invariant to metric scaling), where $d_{\text{avg}}$ is the average distance in the finite space, as suggested by [BCC15], whose low value also characterizes the “hyperbolicness” of the space. An equivalent and more intuitive definition holds for geodesic spaces, i.e. when we can define triangles: its $\delta$-hyperbolicity ($\delta_{\text{worst}}$) is the smallest $\delta > 0$ such that for any triangle $\Delta xyz$, there exists a point at distance at most $\delta$ from each side of the triangle. [Che+13] and [BCC15] analyzed $\delta_{\text{worst}}$ and $\delta_{\text{avg}}$ for specific graphs, respectively. A low hyperbolicity of a graph indicates that it has an underlying hyperbolic geometry, i.e. that it is approximately tree-like, or at least that there exists a taxonomy of nodes. Conversely, a high hyperbolicity of a graph suggests that it possesses long cycles, or could not be embedded in a low dimensional hyperbolic space without distortion. For instance, the Euclidean space $\mathbb{R}^n$ is not $\delta$-hyperbolic for any $\delta > 0$, and is hence described as $\infty$-hyperbolic, while the Poincaré disk $\mathbb{D}^2$ is known to have a $\delta$-hyperbolicity of $\log(1 + \sqrt{2}) \simeq 0.88$. On the other-hand, a product $\mathbb{D}^2 \times \mathbb{D}^2$ is $\infty$-hyperbolic, because a 2D plane $\mathbb{R}^2$ could be
isometrically embedded in it using for instance the first coordinates of each $\mathbb{D}^2$. However, if $\mathbb{D}^2$ would constitute a good choice to embed some given symbolic data, then most likely $\mathbb{D}^2 \times \mathbb{D}^2$ would as well. This stems from the fact that $\delta$-hyperbolicity ($\delta_{\text{worst}}$) is a worst case measure which does not reflect what one could call the “hyperbolic capacity” of the space. Furthermore, note that computing $\delta_{\text{worst}}$ requires $O(n^4)$ for a graph of size $n$, while $\delta_{\text{avg}}$ can be approximated via sampling. Finally, $\delta_{\text{avg}}$ is robust to adding/removing a node from the graph, while $\delta_{\text{worst}}$ is not. For all these reasons, we choose $\delta_{\text{avg}}$ as a measure of hyperbolicity.

**Computing $\delta_{\text{avg}}$.** Since our methods are trained on a weighted graph of co-occurrences, it makes sense to look for the corresponding hyperbolicity $\delta_{\text{avg}}$ of this symbolic data. The lower this value, the more hyperbolic is the underlying nature of the graph, thus indicating that the hyperbolic space should be preferred over the Euclidean space for embedding words. However, in order to do so, one needs to be provided with a distance $d(i, j)$ for each pair of words $(i, j)$, while our symbolic data is only made of similarity measures. Inspired by eq. (3.44), we associate to words $i, j$ the distance $^9$ $h(d(i, j)) := -\log(X_{ij}) + b_i + b_j \geq 0$ with the choice $b_i := \log(X_{ii})$, i.e.

\[
d(i, j) := h^{-1}(\log((X_jX_j)/X_{ij})). \tag{3.51}
\]

Table 3.7 shows values for different choices of $h$. The discrete metric spaces we obtained for our symbolic data of co-occurrences appear to have a very low hyperbolicity, i.e. to be very much “hyperbolic”, which suggests to embed words in (products of) hyperbolic spaces.

\[^9\text{One can replace } \log(x) \text{ with } \log(1 + x) \text{ to avoid computing the logarithm of zero.}\]
3.2 Word Embeddings

<table>
<thead>
<tr>
<th>( h(x) )</th>
<th>( \log(x) )</th>
<th>( x )</th>
<th>( x^2 )</th>
<th>( \cosh(x) )</th>
<th>( \cosh^2(x) )</th>
<th>( \cosh^4(x) )</th>
<th>( \cosh^{10}(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d_{avg} )</td>
<td>18950.4</td>
<td>18.9505</td>
<td>4.3465</td>
<td>3.68</td>
<td>2.3596</td>
<td>1.7918</td>
<td>1.4947</td>
</tr>
<tr>
<td>( \delta_{avg} )</td>
<td>8498.6</td>
<td>0.7685</td>
<td>0.088</td>
<td>0.0384</td>
<td>0.0167</td>
<td>0.0072</td>
<td>0.0026</td>
</tr>
<tr>
<td>( \frac{2\delta_{avg}}{d_{avg}} )</td>
<td>0.8969</td>
<td>0.0811</td>
<td>0.0405</td>
<td>0.0209</td>
<td>0.0142</td>
<td>0.0081</td>
<td>0.0034</td>
</tr>
</tbody>
</table>

Table 3.7: Average distances, \( \delta \)-hyperbolicities and ratios computed via sampling for the metrics induced by different \( h \) functions, as defined in eq. (3.51).

Table 3.8: Similarity results on the unrestricted (190k) vocabulary for various \( h \) functions. This table should be read together with Table 3.7.

<table>
<thead>
<tr>
<th>Experiment’s name</th>
<th>Rare Word</th>
<th>WordSim</th>
<th>SimLex</th>
<th>SimVerb</th>
</tr>
</thead>
<tbody>
<tr>
<td>100D Vanilla</td>
<td>0.3840</td>
<td>0.5849</td>
<td>0.3020</td>
<td>0.1674</td>
</tr>
<tr>
<td>100D Poincaré, ( \cosh )</td>
<td>0.3353</td>
<td>0.5841</td>
<td>0.2607</td>
<td>0.1394</td>
</tr>
<tr>
<td>100D Poincaré, ( \cosh^2 )</td>
<td>0.3981</td>
<td>0.6509</td>
<td>0.3131</td>
<td>0.1757</td>
</tr>
<tr>
<td>100D Poincaré, ( \cosh^3 )</td>
<td>0.4170</td>
<td>0.6314</td>
<td>0.3155</td>
<td>0.1825</td>
</tr>
<tr>
<td>100D Poincaré, ( \cosh^4 )</td>
<td><strong>0.4272</strong></td>
<td>0.6294</td>
<td><strong>0.3198</strong></td>
<td><strong>0.1845</strong></td>
</tr>
</tbody>
</table>

**More experiments.** As explained above, we computed hyperbolicities of the metric space induced by different \( h \) functions, on the matrix of co-occurrence counts, as reported in Table 3.7. We also conducted similarity experiments, reported in Table 3.8. Apart from WordSim, results improved for higher powers of \( \cosh \), corresponding to more hyperbolic spaces. However, also note that higher powers will tend to result in words embedded much closer to each other, *i.e.* with smaller distances. In order to know whether this benefit comes from contracting distances or making the space more “hyperbolic”, it would be interesting to learn (or cross-validate) the curvature \( c \) of the Poincaré ball (or equivalently, its radius) jointly with the \( h \) function. Finally, in order to explain why WordSim behaved differently compared to other benchmarks, we investigated different properties of these, as reported.
in Table 3.9. The geometry of the words appearing in WordSim do not seem to have a different hyperbolicity compared to other benchmarks; however, WordSim seems to contain much more frequent words. Since hyperbolicities are computed with the assumption that $b_i = \log(X_i)$ (see Eq. (3.51)), it would be interesting to explore whether learned biases indeed take these values. We left this as future work.

Table 3.9: Various properties of similarity benchmark datasets. The frequency index indicates the rank of a word in the vocabulary in terms of its frequency: a low index describes a frequent word. The median of indexes seems to best discriminate WordSim from SimLex and SimVerb.

<table>
<thead>
<tr>
<th>Property</th>
<th>WordSim</th>
<th>SimLex</th>
<th>SimVerb</th>
</tr>
</thead>
<tbody>
<tr>
<td># of test instances</td>
<td>353</td>
<td>999</td>
<td>3,500</td>
</tr>
<tr>
<td># of different words</td>
<td>419</td>
<td>1,027</td>
<td>822</td>
</tr>
<tr>
<td>min. index (frequency)</td>
<td>57</td>
<td>38</td>
<td>21</td>
</tr>
<tr>
<td>max. index (frequency)</td>
<td>58,286</td>
<td>128,143</td>
<td>180,417</td>
</tr>
<tr>
<td>median of indexes</td>
<td>2,723</td>
<td>4,463</td>
<td>9,338</td>
</tr>
<tr>
<td>$\delta_{\text{avg}} \cdot (\cdot)^2$</td>
<td>0.0738</td>
<td>0.0759</td>
<td>0.0799</td>
</tr>
<tr>
<td>$\delta_{\text{avg}} \cdot \cosh^2$</td>
<td>0.0154</td>
<td>0.0156</td>
<td>0.0164</td>
</tr>
<tr>
<td>$2\delta_{\text{avg}}/d_{\text{avg}} \cdot (\cdot)^2$</td>
<td>0.0381</td>
<td>0.0384</td>
<td>0.0399</td>
</tr>
<tr>
<td>$2\delta_{\text{avg}}/d_{\text{avg}} \cdot \cosh^2$</td>
<td>0.0136</td>
<td>0.0137</td>
<td>0.0143</td>
</tr>
</tbody>
</table>

### 3.3 Classification & Regression

#### 3.3.1 Matrix Multiplications & Pointwise non-Linearities

In order to define hyperbolic neural networks, it is crucial to define a canonically simple parametric family of transformations, playing the role of linear mappings in usual Euclidean neural networks, and to know how to apply pointwise non-linearities. Inspiring ourselves from our reformulation of Möbius scalar multiplication in eq. (3.19), we define:
3.3 Classification & Regression

**Definition 3.2 (Möbius version).** For \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m \), we define the **Möbius version** of \( f \) as the map from \( D^c_n \) to \( D^c_m \) by:

\[
f^\otimes_c(x) := \exp_0^c(f(\log_0^c(x))),
\]

where \( \exp_0^c : T_0 D^c_n \rightarrow D^c_n \) and \( \log_0^c : D^c_n \rightarrow T_0 D^c_n \).

Note that similarly as for other Möbius operations, we recover the Euclidean mapping in the limit \( c \rightarrow 0 \) if \( f \) is continuous, as

\[
\lim_{c \rightarrow 0} f^\otimes_c(x) = f(x).
\]

This definition satisfies a few desirable properties too, such as:

\[
(f \circ g)^\otimes_c = f^\otimes_c \circ g^\otimes_c \quad \text{for} \quad f : \mathbb{R}^m \rightarrow \mathbb{R}^l \quad \text{and} \quad g : \mathbb{R}^n \rightarrow \mathbb{R}^m \quad \text{(morphism property)},
\]

\[
f^\otimes_c(x)/\|f^\otimes_c(x)\| = f(x)/\|f(x)\| \quad \text{for} \quad f(x) \neq 0 \quad \text{(direction preserving)}.
\]

It is then straightforward to prove the following result:

**Lemma 3.4 (Möbius matrix-vector multiplication).** If \( M : \mathbb{R}^n \rightarrow \mathbb{R}^m \) is a linear map, which we identify with its matrix representation \( M \in M_{m,n}(\mathbb{R}) \), then \( \forall x \in D^c_n \), if \( Mx \neq 0 \) we have

\[
M^\otimes_c(x) = (1/\sqrt{c}) \tanh \left( \frac{\|Mx\|}{\|x\|} \tanh^{-1}(\sqrt{c}\|x\|) \right) \frac{Mx}{\|Mx\|},
\]

and \( M^\otimes_c(x) = 0 \) if \( Mx = 0 \). Moreover, if we define the Möbius matrix-vector multiplication of \( M \) with \( x \) by \( M \otimes_c x := M^\otimes_c(x) \), then we have:

- \( (MM') \otimes_c x = M \otimes_c (M' \otimes_c x) \) for any \( M \in M_{l,m}(\mathbb{R}) \) and \( M' \in M_{m,n}(\mathbb{R}) \) (matrix associativity)
- \( (rM) \otimes_c x = r \otimes_c (M \otimes_c x) \) for \( r \in \mathbb{R} \) and \( M \in M_{m,n}(\mathbb{R}) \) (scalar-matrix associativity)
- \( M \otimes_c x = Mx \) for all \( M \in O_n(\mathbb{R}) \) (rotations are preserved)

**Pointwise non-linearity.** If \( \varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a pointwise non-linearity, then its Möbius version \( \varphi^\otimes_c \) can be applied to elements of the Poincaré ball.

**Bias translation.** The generalization of a translation in the Poincaré ball is naturally given by moving along geodesics. But should we use the Möbius sum \( x \oplus_c b \) with a hyperbolic bias \( b \in D^c_n \) or the exponential map \( \exp_x^c(b') \) with a Euclidean bias \( b' \in \mathbb{R}^n \)? These views are
unified with parallel transport (see theorem 3.1). Möbius translation of a point \( x \in D_{c}^{n} \) by a bias \( b \in D_{c}^{n} \) is given by

\[
 x \leftarrow x \oplus c b = \exp_c (P_{0 \to x} (\log_c (b))) = \exp_x \left( \frac{\lambda_c}{\lambda_x} \log_0 (b) \right). \quad (3.54)
\]

We recover Euclidean translations in the limit \( c \to 0 \). Note that bias translations play a particular role in this model. Indeed, consider multiple layers of the form \( f_k(x) = \varphi_k(M_k x) \), each of which having Möbius version \( f_k^c(x) = \varphi_k^c(M_k \otimes_c x) \). Then their composition can be re-written \( f_k^c \circ \cdots \circ f_1^c = \exp_0^c \circ f_k \circ \cdots \circ f_1 \circ \log_0^c \). This means that these operations can essentially be performed in Euclidean space. Therefore, it is the interposition between those with the bias translation of eq. (3.54) which differentiates this model from its Euclidean counterpart.

### Concatenation of Multiple Input Vectors

If a vector \( x \in \mathbb{R}^{n+p} \) is the (vertical) concatenation of two vectors \( x_1 \in \mathbb{R}^n \), \( x_2 \in \mathbb{R}^p \), and \( M \in \mathcal{M}_{m,n+p}(\mathbb{R}) \) can be written as the (horizontal) concatenation of two matrices \( M_1 \in \mathcal{M}_{m,n}(\mathbb{R}) \) and \( M_2 \in \mathcal{M}_{m,p}(\mathbb{R}) \), then \( Mx = M_1 x_1 + M_2 x_2 \).

We generalize this to hyperbolic spaces: if we are given \( x_1 \in D_{c}^{n} \), \( x_2 \in D_{c}^{p} \), \( x = (x_1 x_2)^T \in D_{c}^{n} \times D_{c}^{p} \), and \( M, M_1, M_2 \) as before, then we define

\[
 M \otimes_c x := M_1 \otimes_c x_1 \oplus_c M_2 \otimes_c x_2 \quad (3.55)
\]

Note that, when \( c \) goes to zero, we recover the Euclidean formulation, as

\[
 \lim_{c \to 0} M \otimes_c x = \lim_{c \to 0} M_1 \otimes_c x_1 \oplus_c M_2 \otimes_c x_2 = M_1 x_1 + M_2 x_2 = Mx \quad (3.56)
\]

Moreover, hyperbolic vectors \( x \in D_{c}^{n} \) can also be “concatenated” with real features \( y \in \mathbb{R} \) via \( M \otimes_c x \oplus_c y \otimes_c b \) with learnable bias \( b \in D_{c}^{n} \) and weight matrices \( M \in \mathcal{M}_{m,n}(\mathbb{R}) \).

### 3.3.2 Recurrent Networks & Gated Recurrent Units

**Naive RNN.** A simple RNN is defined by

\[
 h_{t+1} = \varphi(W h_t + U x_t + b) \quad (3.57)
\]
where \( \varphi \) is a pointwise non-linearity, typically tanh, sigmoid, ReLU, etc. This formula can be naturally generalized to the hyperbolic space as follows. For parameters \( \mathbf{W} \in \mathcal{M}_{m,n}(\mathbb{R}), \mathbf{U} \in \mathcal{M}_{m,d}(\mathbb{R}), \mathbf{b} \in \mathbb{D}_c^m \), we define:

\[
\mathbf{h}_{t+1} = \varphi \odot (\mathbf{W} \odot \mathbf{h}_t \odot \mathbf{x}_t \odot \mathbf{b}), \quad \mathbf{h}_t \in \mathbb{D}_c^n, \ \mathbf{x}_t \in \mathbb{D}_c^d.
\]

(3.58)

Note that if inputs \( \mathbf{x}_t \)'s are Euclidean, one can write \( \mathbf{x}_t := \exp_c^c(\mathbf{x}_t) \) and use the above formula because of the following relation

\[
\exp_{W \odot \mathbf{h}_t}^c (P_{0 \to W \odot \mathbf{h}_t}(\mathbf{U} \mathbf{x}_t)) = \mathbf{W} \odot \mathbf{h}_t \odot \mathbf{U} \odot \mathbf{x}_t \mathbf{b}^c
\]

(3.59)

**GRU Architecture.** One can also adapt the GRU architecture:

\[
\begin{align*}
\mathbf{r}_t &= \sigma(\mathbf{W}^r \mathbf{h}_{t-1} + \mathbf{U}^r \mathbf{x}_t + \mathbf{b}^r) \\
\mathbf{z}_t &= \sigma(\mathbf{W}^z \mathbf{h}_{t-1} + \mathbf{U}^z \mathbf{x}_t + \mathbf{b}^z) \\
\tilde{\mathbf{h}}_t &= \varphi(\mathbf{W}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{U} \mathbf{x}_t + \mathbf{b}) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t,
\end{align*}
\]

(3.60)

where \( \odot \) denotes pointwise product. First, how should we adapt the pointwise multiplication by a scaling gate? Note that the definition of the Möbius version (see eq. (3.52)) can be naturally extended to maps \( f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m \) as

\[
f^\odot_c : \mathbb{D}_c^n \times \mathbb{D}_c^p \to \mathbb{D}_c^m,
\]

\[f^\odot_c (\mathbf{h}, \mathbf{h}') = \exp_c^c(f(\log_c^c(\mathbf{h}), \log_c^c(\mathbf{h}')))
\]

(3.61)

In particular, choosing \( f(\mathbf{h}, \mathbf{h}') := \sigma(\mathbf{h}) \odot \mathbf{h}' \) yields\(^{10}\)

\[f^\odot_c (\mathbf{h}, \mathbf{h}') = \exp_c^c(\sigma(\log_c^c(\mathbf{h})) \odot \log_c^c(\mathbf{h}')) = \text{diag}(\sigma(\log_c^c(\mathbf{h}))) \odot \mathbf{h}'
\]

(3.62)

Hence we adapt \( \mathbf{r}_t \odot \mathbf{h}_{t-1} \) to \( \text{diag}(\mathbf{r}_t) \odot_c \mathbf{h}_{t-1} \) and the reset gate \( \mathbf{r}_t \) to:

\[
\mathbf{r}_t = \sigma \log_c^c(\mathbf{W}^r \odot \mathbf{h}_{t-1} \odot \mathbf{U}^r \odot \mathbf{x}_t \odot \mathbf{b}^r),
\]

(3.63)

and similarly for the update gate \( \mathbf{z}_t \). Note that as the argument of \( \sigma \) in the above is unbounded, \( \mathbf{r}_t \) and \( \mathbf{z}_t \) can a priori take values onto the full range \((0, 1)\). Now the intermediate hidden state becomes:

\(^{10}\)If \( \mathbf{x} \) has \( n \) coordinates, then \( \text{diag}(\mathbf{x}) \) denotes the diagonal matrix of size \( n \) with \( \mathbf{x}_i \)'s on its diagonal.
hyperbolic spaces: embeddings & classification

\[ \mathbf{h}_t = \varphi^{ \otimes c} ( (\mathbf{W} \text{diag} (r_t)) \otimes_c \mathbf{h}_{t-1} \oplus_c \mathbf{U} \otimes_c \mathbf{x}_t \oplus \mathbf{b}), \] (3.64)

where Möbius matrix associativity simplifies \( \mathbf{W} \otimes_c (\text{diag}(r_t) \otimes_c \mathbf{h}_{t-1}) \) into \((\mathbf{W} \text{diag}(r_t)) \otimes_c \mathbf{h}_{t-1} \). Finally, we propose to adapt the update-gate equation as

\[ \mathbf{h}_t = \mathbf{h}_{t-1} \oplus_c \text{diag} (\mathbf{z}_t) \otimes_c (\mathbf{h}_{t-1} \oplus_c \mathbf{\tilde{h}}_t). \] (3.65)

Note that when \( c \) goes to zero, one recovers the usual GRU. Moreover, if \( \mathbf{z}_t = 0 \) or \( \mathbf{z}_t = 1 \), then \( \mathbf{h}_t \) becomes \( \mathbf{h}_{t-1} \) or \( \mathbf{\tilde{h}}_t \) respectively, similarly as in the usual GRU. This adaptation was obtained by adapting [TO18]: in this work, the authors re-derive the update-gate mechanism from a first principle called time-warping invariance. We adapted their derivation to the hyperbolic setting by using the notion of gyroderivative [BU01] and proving a gyro-chain-rule. The idea consists in the following.

**RECOVERING THE UPDATE-GATE FROM TIME-WARPING.** A naive RNN is given by the equation

\[ h(t + 1) = \varphi(\mathbf{W}h(t) + \mathbf{U}x(t) + b) \] (3.66)

Let’s drop the bias \( b \) to simplify notations. If \( h \) is seen as a differentiable function of time, then a first-order Taylor development gives \( h(t + \delta t) \approx h(t) + \delta t \frac{dh}{dt}(t) \) for small \( \delta t \). Combining this for \( \delta t = 1 \) with the naive RNN equation, one gets

\[ \frac{dh}{dt}(t) = \varphi(\mathbf{W}h(t) + \mathbf{U}x(t)) - h(t). \] (3.67)

As this is written for any \( t \), one can replace it by \( t \leftarrow \alpha(t) \) where \( \alpha \) is a (smooth) increasing function of \( t \) called the time-warping. Denoting by \( \mathbf{\tilde{h}}(t) := h(\alpha(t)) \) and \( \mathbf{\tilde{x}}(t) := x(\alpha(t)) \), using the chain rule \( \frac{dh}{dt}(t) = \frac{d\alpha}{dt}(t) \frac{d\mathbf{\tilde{h}}}{dt}(\alpha(t)) \), one gets

\[ \frac{d\mathbf{\tilde{h}}}{dt}(t) \approx \frac{d\alpha}{dt}(t) \varphi(\mathbf{W}\mathbf{\tilde{h}}(t) + \mathbf{U}\mathbf{\tilde{x}}(t)) - \frac{d\alpha}{dt}(t)\mathbf{\tilde{h}}(t). \] (3.68)

Removing the tildas to simplify notations, discretizing back with \( \frac{dh}{dt}(t) \approx h(t+1) - h(t) \) yields

\[ h(t+1) = \frac{d\alpha}{dt}(t) \varphi(\mathbf{W}h(t) + \mathbf{U}x(t)) + \left(1 - \frac{d\alpha}{dt}(t)\right) h(t). \] (3.69)
3.3 Classification & Regression

Requiring that our class of neural networks be invariant to time-warpings means that this class should contain RNNs defined by Eq. (3.69), i.e. that \( \frac{da}{dt}(t) \) can be learned. As this is a positive quantity, we can parametrize it as \( z(t) = \sigma(W^z h(t) + U^z x(t)) \), recovering the forget-gate equation:

\[
h(t + 1) = z(t) \varphi(W h(t) + U x(t)) + (1 - z(t)) h(t). \tag{3.70}
\]

Adapting this idea to hyperbolic RNNs. The gyroderivative [BÜ01] of a map \( h: \mathbb{R} \rightarrow \mathbb{D}^n_b \) is defined as

\[
\frac{dh}{dt}(t) = \lim_{\delta t \rightarrow 0} \frac{1}{\delta t} \bigotimes_c (-h(t) \oplus_c h(t + \delta t)). \tag{3.71}
\]

Using Möbius scalar associativity and the left-cancellation law leads us to

\[
h(t + \delta t) \approx h(t) \bigotimes_c \delta t \bigotimes_c \frac{dh}{dt}(t), \tag{3.72}
\]

for small \( \delta t \). Combining this with the equation of a simple hyperbolic RNN of Eq. (3.58) with \( \delta t = 1 \), one gets

\[
\frac{dh}{dt}(t) = -h(t) \bigotimes_c \varphi^{\otimes c}(W \bigotimes_c h(t) \bigotimes_c U \bigotimes_c x(t)). \tag{3.73}
\]

For the next step, we need the following lemma:

**Lemma 3.5 (Gyro-chain-rule).** For \( \alpha: \mathbb{R} \rightarrow \mathbb{R} \) differentiable and \( h: \mathbb{R} \rightarrow \mathbb{D}^n_b \) with a well-defined gyro-derivative, if \( \tilde{h} := h \circ \alpha \), then we have

\[
\frac{d\tilde{h}}{dt}(t) = \frac{d\alpha}{dt}(t) \bigotimes_c \frac{dh}{dt}(\alpha(t)), \tag{3.74}
\]

where \( \frac{d\alpha}{dt}(t) \) denotes the usual derivative.
Proof.

\[
\frac{dh}{dt}(t) = \lim_{\delta t \to 0} \frac{1}{\delta t} \otimes_c [-\tilde{h}(t) \oplus_c \tilde{h}(t + \delta t)]
\]

(3.75)

\[
= \lim_{\delta t \to 0} \frac{1}{\delta t} \otimes_c [-h(\alpha(t)) \oplus_c h(\alpha(t) + \delta t(\alpha'(t) + O(\delta t)))]
\]

(3.76)

\[
= \lim_{\delta t \to 0} \frac{\alpha'(t) + O(\delta t)}{\delta t(\alpha'(t) + O(\delta t))} \otimes_c [-h(\alpha(t)) \oplus_c h(\alpha(t) + \delta t(\alpha'(t) + O(\delta t)))]
\]

(3.77)

\[
= \lim_{u \to 0} \frac{\alpha'(t)}{u} \otimes_c [-h(\alpha(t)) \oplus_c h(\alpha(t) + u)]
\]

(3.78)

\[
= \frac{d\alpha}{dt}(t) \otimes_c \frac{dh}{dt}(\alpha(t)) \quad \text{(Möbius scalar associativity)}
\]

(3.80)

where we set \(u = \delta t(\alpha'(t) + O(\delta t))\), with \(u \to 0\) when \(\delta t \to 0\), which concludes.

Using lemma 3.5 and Eq. (3.73), with similar notations as in Eq. (3.68) we have

\[
\frac{dh}{dt}(t) = \frac{d\alpha}{dt}(t) \otimes_c (-\tilde{h}(t) \oplus_c \phi \otimes_c (W \otimes_c \tilde{h}(t) \oplus_c U \otimes_c \tilde{x}(t))).
\]

(3.81)

Finally, discretizing back with Eq. (3.72), using the left-cancellation law and dropping the tildas yields

\[
h(t + 1) = h(t) \oplus_c \frac{d\alpha}{dt}(t) \otimes_c (-h(t) \oplus_c \phi \otimes_c (W \otimes_c h(t) \oplus_c U \otimes_c x(t))).
\]

(3.82)

Since \(\alpha\) is a time-warping, by definition its derivative is positive and one can choose to parametrize it with an update-gate \(z_t\) (a scalar) defined with a sigmoid. Generalizing this scalar scaling by the Möbius version of the pointwise scaling \(\otimes\) yields the Möbius matrix scaling \(\text{diag}(z_t) \otimes_c \cdot\), leading to our proposed Eq. (3.65) for the hyperbolic GRU.

**Empirical Validation: Textual Entailment**

**SNLI task and dataset.** We evaluate our method on two tasks. The first is natural language inference, or textual entailment. Given two
classifica
tion & regression

sentences, a premise (e.g. “Little kids A. and B. are playing soccer.”) and a hypothesis (e.g. “Two children are playing outdoors.”), the binary classification task is to predict whether the second sentence can be inferred from the first one. This defines a partial order in the sentence space. We test hyperbolic networks on the biggest real dataset for this task, SNLI [Bow+15]. It consists of 570K training, 10K validation and 10K test sentence pairs. Following [Ven+15], we merge the “contradiction” and “neutral” classes into a single class of negative sentence pairs, while the “entailment” class gives the positive pairs.

prefix task and datasets. We conjecture that the improvements of hyperbolic neural networks are more significant when the underlying data structure is closer to a tree. To test this, we design a proof-of-concept task of detection of noisy prefixes, i.e. given two sentences, one has to decide if the second sentence is a noisy prefix of the first, or a random sentence. We thus build synthetic datasets PREFIX-Z% (for Z being 10, 30 or 50) as follows: for each random first sentence of random length at most 20 and one random prefix of it, a second positive sentence is generated by randomly replacing Z% of the words of the prefix, and a second negative sentence of same length is randomly generated. Word vocabulary size is 100, and we generate 500K training, 10K validation and 10K test pairs.

models architecture. Our neural network layers can be used in a plug-n-play manner exactly like standard Euclidean layers. They can also be combined with Euclidean layers. However, optimization w.r.t. hyperbolic parameters is different (see below) and based on Riemannian gradients which are just rescaled Euclidean gradients when working in the conformal Poincaré model [NK17b]. Thus, back-propagation can be applied in the standard way.

In our setting, we embed the two sentences using two distinct hyperbolic RNNs or GRUs. The sentence embeddings are then fed together with their squared distance (hyperbolic or Euclidean, depending on their geometry) to a Feed-forward Neural Network (FFNN) (Euclidean or hyperbolic, see section 3.3.1) which is further fed to an MLR (Euclidean or hyperbolic, see section 3.3.3) that gives probabilities of the two classes (entailment vs neutral). We use cross-entropy loss on top. Note that hyperbolic and Euclidean layers can be mixed, e.g. the full network can be hyperbolic and only the last layer be Euclidean, in
which case one has to use $\log_0$ and $\exp_0$ functions to move between the two manifolds in a correct manner as explained for eq. (3.52).

**Optimization.** Our models have both Euclidean (e.g. weight matrices in both Euclidean and hyperbolic FFNNs, RNNs or GRUs) and hyperbolic parameters (e.g. word embeddings or biases for the hyperbolic layers). We optimize the Euclidean parameters with Adam [KB15] (learning rate 0.001). Hyperbolic parameters cannot be updated with an equivalent method that keeps track of gradient history due to the absence of a Riemannian Adam. Thus, they are optimized using full RSGD [Bon13]. We also experiment with projected RSGD [NK17b], but optimization was sometimes less stable. We use a different constant learning rate for word embeddings (0.1) and other hyperbolic weights (0.01) because words are updated less frequently.

**Numerical errors.** Gradients of the basic operations defined above (e.g. $\oplus_c$, exponential map) are not defined when the hyperbolic argument vectors are on the ball border, i.e. $\sqrt{c \|x\|} = 1$. Thus, we always project results of these operations in the ball of radius $1 - \epsilon$, where $\epsilon = 10^{-5}$. Numerical errors also appear when hyperbolic vectors get closer to 0, thus we perturb them with an $\epsilon' = 10^{-15}$ before they are used in any of the above operations. Finally, arguments of the tanh function are clipped between $\pm 15$ to avoid numerical errors, while arguments of $\tanh^{-1}$ are clipped to at most $1 - 10^{-5}$.

**Hyperparameters.** For all methods, baselines and datasets, we use $c = 1$, word and hidden state embedding dimension of 5 (we focus on the low dimensional setting that was shown to already be effective [NK17b]), batch size of 64. We ran all methods for a fixed number of 30 epochs. For all models, we experiment with both identity (no non-linearity) or tanh non-linearity in the RNN/GRU cell, as well as identity or ReLU after the FFNN layer and before MLR. As expected, for the fully Euclidean models, tanh and ReLU respectively surpassed the identity variant by a large margin. We only report the best Euclidean results. Interestingly, for the hyperbolic models, using only identity for both non-linearities works slightly better and this is likely due to the fact that our hyperbolic layers already contain non-linearities by their nature.
## 3.3 Classification & Regression

<table>
<thead>
<tr>
<th></th>
<th>SNLI</th>
<th>PREFIX 10%</th>
<th>PREFIX 30%</th>
<th>PREFIX 50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eucl RNN+FFNN</td>
<td>79.34 %</td>
<td>89.62 %</td>
<td>81.71 %</td>
<td>72.10 %</td>
</tr>
<tr>
<td>Eucl MLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyp RNN+FFNN</td>
<td>79.18 %</td>
<td>96.36 %</td>
<td>87.83 %</td>
<td>76.50 %</td>
</tr>
<tr>
<td>Hyp MLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyp RNN+FFNN</td>
<td>78.21 %</td>
<td>96.91 %</td>
<td>87.25 %</td>
<td>62.94 %</td>
</tr>
<tr>
<td>Hyp MLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eucl GRU+FFNN</td>
<td>81.52 %</td>
<td>95.96 %</td>
<td>86.47 %</td>
<td>75.04 %</td>
</tr>
<tr>
<td>Eucl MLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyp GRU+FFNN</td>
<td>79.76 %</td>
<td>97.36 %</td>
<td>88.47 %</td>
<td>76.87 %</td>
</tr>
<tr>
<td>Hyp MLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hyp RNN+FFNN</td>
<td>81.19 %</td>
<td>97.14 %</td>
<td>88.26 %</td>
<td>76.44 %</td>
</tr>
<tr>
<td>Hyp MLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.5: Test accuracies for various models and four datasets. “Eucl” denotes Euclidean, “Hyp” denotes hyperbolic. All word and sentence embeddings have dimension 5. We highlight in **bold** the best method (or methods, if the difference is less than 0.5%). Implementation of these experiments was done by co-authors in [GBH18c].

For the results shown in fig. 3.5, we run each model (baseline or ours) exactly 3 times and report the test result corresponding to the best validation result from these 3 runs. We do this because the highly non-convex spectrum of hyperbolic neural networks sometimes results in convergence to poor local minima, suggesting that initialization is very important.

**Results.** Results are shown in fig. 3.5. Note that the fully Euclidean baseline models might have an advantage over hyperbolic baselines because more sophisticated optimization algorithms such as Adam do not have a hyperbolic analogue at the moment. We first observe
Figure 3.6: PREFIX-30% accuracy and first (premise) sentence norm plots for different runs of the same architecture: hyperbolic GRU followed by hyperbolic FFNN and hyperbolic/Euclidean (half-half) MLR. The X axis shows millions of training examples processed. Implementation of these experiments was done by co-authors in [GBH18c].

that all GRU models overpass their RNN variants. Hyperbolic RNNs and GRUs have the most significant improvement over their Euclidean variants when the underlying data structure is more tree-like, e.g. for PREFIX-10% — for which the tree relation between sentences and their prefixes is more prominent — we reduce the error by a factor of 3.35 for hyperbolic vs Euclidean RNN, and by a factor of 1.5 for hyperbolic vs Euclidean GRU. As soon as the underlying structure diverges more and more from a tree, the accuracy gap decreases — for example, for PREFIX-50% the noise heavily affects the representational power of hyperbolic networks. Also, note that on SNLI our methods perform similarly as with their Euclidean variants. Moreover, hyperbolic and Euclidean MLR are on par when used in conjunction with hyperbolic sentence embeddings, suggesting further empirical investigation is needed for this direction (see below).
3.3 Classification & Regression

![Figure 3.7](image)

(a) Test accuracy

(b) Norm of the first sentence. Averaged over all sentences in the test set.

Figure 3.7: PREFIX-30% accuracy and first (premise) sentence norm plots for different runs of the same architecture: Euclidean GRU followed by Euclidean FFNN and Euclidean MLR. The X axis shows millions of training examples processed. Implementation of these experiments was done by co-authors in [GBH18c].

Additional Experimental Results  We observed that, in the hyperbolic setting, accuracy is often much higher when sentence embeddings increase norms, tending to diverge towards the “infinity” border of the Poincaré ball. Moreover, the faster the two sentence norms go to 1, the more it’s likely that a good local minima was reached. See figs. 3.6 and 3.8.

We often observe that test accuracy starts increasing exactly when sentence embedding norms do. However, in the hyperbolic setting, the sentence embeddings norms remain close to 0 for a few epochs, which does not happen in the Euclidean case. See figs. 3.6 to 3.8. This surprising behavior was also exhibited in a similar way by [NK17b] which suggests that the model first has to adjust the angular layout in the almost Euclidean vicinity of 0 before increasing norms and fully exploiting hyperbolic geometry.
3.3.3 Multinomial Logistic Regression

In order to perform multi-class classification on the Poincaré ball, one needs to generalize MLR — also called softmax regression — to the Poincaré ball.

REFORMULATING EUCLIDEAN MLR. Let’s first reformulate Euclidean MLR from the perspective of distances to margin hyperplanes, as in [LL04, Section 5]. This will allow us to easily generalize it.

Given $K$ classes, one learns a margin hyperplane for each such class $k \in \{1, ..., K\}$ using softmax probabilities:

$$p(y = k|\mathbf{x}) \propto \exp \left( (\langle \mathbf{a}_k, \mathbf{x} \rangle - b_k) \right)$$  \ \ (3.83)$$

where $b_k \in \mathbb{R}$ is a class bias, $\mathbf{a}_k \in \mathbb{R}^n$ is a class weight vector and $\mathbf{x} \in \mathbb{R}^n$ is an embedding vector of the point to be classified. Note that any affine
3.3 Classification & Regression

A hyperplane in \( \mathbb{R}^n \) can be written with a normal vector \( \mathbf{a} \in \mathbb{R}^n \setminus \{0\} \) and a scalar shift \( b \in \mathbb{R} \):

\[
H_{\mathbf{a},b} = \{ \mathbf{x} \in \mathbb{R}^n : \langle \mathbf{a}, \mathbf{x} \rangle - b = 0 \} \tag{3.84}
\]

As in [LL04, Section 5], we note that

\[
\langle \mathbf{a}, \mathbf{x} \rangle - b = \text{sign}(\langle \mathbf{a}, \mathbf{x} \rangle - b) \| \mathbf{a} \| d(\mathbf{x}, H_{\mathbf{a},b}) \tag{3.85}
\]

Using eq. (3.83), we get for \( b_k \in \mathbb{R}, \mathbf{x}, \mathbf{a}_k \in \mathbb{R}^n \):

\[
p(y = k | \mathbf{x}) \propto \exp(\text{sign}(\langle \mathbf{a}_k, \mathbf{x} \rangle - b_k) \| \mathbf{a}_k \| d(\mathbf{x}, H_{\mathbf{a}_k,b_k})) \tag{3.86}
\]

As it is not immediately obvious how to generalize the Euclidean hyperplane of eq. (3.84) to other spaces such as the Poincaré ball, we reformulate it as follows:

\[
\tilde{H}_{\mathbf{a},p} = \{ \mathbf{x} \in \mathbb{R}^n : \langle -\mathbf{p} + \mathbf{x}, \mathbf{a} \rangle = 0 \} = \mathbf{p} + \{ \mathbf{a} \}^\perp \tag{3.87}
\]

where \( \mathbf{p} \in \mathbb{R}^n, \mathbf{a} \in \mathbb{R}^n \setminus \{0\} \). This new definition relates to the previous one as \( \tilde{H}_{\mathbf{a},p} = H_{\mathbf{a}}(\mathbf{a}, \mathbf{p}) \). Rewriting eq. (3.86) with \( b = \langle \mathbf{a}, \mathbf{p} \rangle \):

\[
p(y = k | \mathbf{x}) \propto \exp(\text{sign}(\langle -\mathbf{p}_k + \mathbf{x}, \mathbf{a}_k \rangle) \| \mathbf{a}_k \| d(\mathbf{x}, \tilde{H}_{\mathbf{a}_k,\mathbf{p}_k})) \tag{3.88}
\]

where \( \mathbf{p}_k, \mathbf{x}, \mathbf{a}_k \in \mathbb{R}^n \). It is now natural to adapt the previous definition to the hyperbolic setting by replacing \( + \) by \( \oplus_c \):

**Definition 3.3 (Poincaré hyperplanes).** For \( \mathbf{p} \in \mathbb{D}_c^n, \mathbf{a} \in T_\mathbf{p}\mathbb{D}_c^n \setminus \{0\} \), we define

\[
\{ \mathbf{a} \}^\perp := \{ \mathbf{z} \in T_\mathbf{p}\mathbb{D}_c^n : \mathcal{g}_\mathbf{p}(\mathbf{z}, \mathbf{a}) = 0 \} = \{ \mathbf{z} \in T_\mathbf{p}\mathbb{D}_c^n : \langle \mathbf{z}, \mathbf{a} \rangle = 0 \} \tag{3.89}
\]

Then, we define\(^a\) Poincaré hyperplanes as

\[
\tilde{H}_{\mathbf{a},p}^c := \{ \mathbf{x} \in \mathbb{D}_c^n : \langle \log_c^c(\mathbf{x}), \mathbf{a} \rangle_p = 0 \} = \exp_c^c(\{ \mathbf{a} \}^\perp) \tag{3.90}
\]

\(^a\)Where \( \langle \cdot, \cdot \rangle \) denotes the (Euclidean) inner-product of the ambient space.

An alternative definition of the Poincaré hyperplanes is

**Lemma 3.6.**

\[
\tilde{H}_{\mathbf{a},p}^c = \{ \mathbf{x} \in \mathbb{D}_c^n : \langle -\mathbf{p} \oplus_c \mathbf{x}, \mathbf{a} \rangle = 0 \}. \tag{3.91}
\]
Proof. Two steps proof:

i) \( \exp_c(\{a\}^\perp) \subseteq \{x \in D^n_c : \langle -p \oplus_c x, a \rangle = 0 \} \):

Let \( z \in \{a\}^\perp \). From eq. (3.17), we have that:
\[
\exp_c(z) = -p \oplus_c \beta z, \quad \text{for some } \beta \in \mathbb{R}.
\] (3.92)

This, together with the left-cancellation law in gyrospaces (see section 3.1.3), implies that
\[
\langle -p \oplus_c \exp_c(z), a \rangle = \langle \beta z, a \rangle = 0 \quad (3.93)
\]
which is what we wanted.

ii) \( \{x \in D^n_c : \langle -p \oplus_c x, a \rangle = 0 \} \subseteq \exp_c(\{a\}^\perp) \):

Let \( x \in D^n_c \) s.t. \( \langle -p \oplus_c x, a \rangle = 0 \). Then, using eq. (3.17), we derive
\[
\log_c(x) = \beta(-p \oplus_c x), \quad \text{for some } \beta \in \mathbb{R},
\] (3.94)
which is orthogonal to \( a \), by assumption. This implies \( \log_c(x) \in \{a\}^\perp \), hence \( x \in \exp_c(\{a\}^\perp) \).
\[\Box\]

Turning back, \( \tilde{H}^c_{a,p} \) can also be described as the union of images of all geodesics in \( D^n_c \) orthogonal to \( a \) and containing \( p \). Notice that our definition matches that of hypergyroplanes, see [Ung14, definition 5.8]. A 3D hyperplane example is depicted in fig. 3.9.

**Theorem 3.4.**

\[
d_c(x, \tilde{H}^c_{a,p}) := \inf_{w \in \tilde{H}^c_{a,p}} d_c(x, w)
\]
\[
= \frac{1}{\sqrt{c}} \sinh^{-1} \left( \frac{2\sqrt{c} |\langle -p \oplus_c x, a \rangle|}{(1 - c \| -p \oplus_c x \|^2) \| a \|} \right).
\] (3.95)

**Proof.** See [GBH18c]. \(\Box\)

**Final formula for MLR in the Poincaré Ball.** Putting together eq. (3.88) and theorem 3.4, we get the hyperbolic MLR formulation. Given \( K \) classes and \( k \in \{1, \ldots, K\} \), the separation hyperplane are determined by \( p_k \in D^n_c, \ a_k \in T_{p_k}D^n_c \setminus \{0\} \) and given for all \( x \in D^n_c \) by:
\[
p(y = k|x) \propto \exp(\text{sign}(\langle -p_k \oplus_c x, a_k \rangle) \sqrt{g^c_{p_k}(a_k, a_k)} d_c(x, \tilde{H}^c_{a_k, p_k}))
\] (3.96)
or, equivalently

\[
p(y = k|\mathbf{x}) \propto \exp\left(\frac{\lambda^c_\mathbf{p}_k \| \mathbf{a}_k \|}{\sqrt{c}} \sinh^{-1}\left(\frac{2\sqrt{c}(-\mathbf{p}_k \oplus_c \mathbf{x}, \mathbf{a}_k)}{(1 - c\| - \mathbf{p}_k \oplus_c \mathbf{x}\|^2)}\right)\right)
\] (3.97)

Notice that when \( c \) goes to zero, the above formula becomes

\[
p(y = k|\mathbf{x}) \propto \exp\left(4(-\mathbf{p}_k + \mathbf{x}, \mathbf{a}_k)\right) = \exp\left((\lambda^0_\mathbf{p}_k)^2(-\mathbf{p}_k + \mathbf{x}, \mathbf{a}_k)\right)
\] (3.98)

recovering the usual Euclidean softmax.

However, it is unclear how to perform optimization over \( \mathbf{a}_k \), since these vectors live in \( T_{\mathbf{p}_k} \mathbb{D}_c^n \) and, hence, depend on \( \mathbf{p}_k \). The solution is that one should write

\[
\mathbf{a}_k = P^c_{0 \rightarrow \mathbf{p}_k}(\mathbf{a}'_k) = \left(\frac{\lambda^c_0}{\lambda^c_{\mathbf{p}_k}}\right)\mathbf{a}'_k
\] (3.99)

where \( \mathbf{a}'_k \in T_0 \mathbb{D}_c^n = \mathbb{R}^n \), and optimize \( \mathbf{a}'_k \) as a Euclidean parameter.
Empirical Validation: Classification of Hierarchical Word Embeddings

**Setup Description.** For the sentence entailment classification task we did not see a clear advantage of using the hyperbolic MLR layer compared to its Euclidean variant. A possible reason is that, when trained end-to-end, the model might decide to place positive and negative embeddings in a manner that is already well separated with a classic MLR. As a consequence, we further investigate MLR for the task of subtree classification. Using an open source implementation\textsuperscript{11} of [NK\textsuperscript{17b}], we pre-trained Poincaré embeddings of the WordNet noun hierarchy (82,115 nodes). We then choose one node in this tree (see fig. 3.11) and classify all other nodes (solely based on their embeddings) as being part of the subtree rooted at this node. All nodes in such a subtree are divided into positive training nodes (80\%) and positive test nodes (20\%).

Figure 3.10: Hyperbolic (left) vs Direct Euclidean (right) binary MLR used to classify nodes as being part in the GROUP N.01 subtree of the WordNet noun hierarchy solely based on their Poincaré embeddings. The positive points (from the subtree) are in red, the negative points (the rest) are in yellow and the trained positive separation hyperplane is depicted in green.

\footnote{https://github.com/dalab/hyperbolic_cones}
The same splitting procedure is applied for the remaining WordNet nodes that are divided into a negative training and negative test sets.

**Training Details.** Three variants of MLR are then trained on top of pre-trained Poincaré embeddings\(^{[NK17b]}\) to solve this binary classification task: hyperbolic MLR, Euclidean MLR applied directly on the hyperbolic embeddings (even if mathematically this is not respecting the hyperbolic geometry) and Euclidean MLR applied after mapping all embeddings in the tangent space at 0 using the \(\log_0\) map.

We use different embedding dimensions: 2, 3, 5 and 10. For the hyperbolic MLR, we use full RSGD with a learning rate of 0.001. For the two Euclidean models we use ADAM optimizer and the same learning rate. During training, we always sample the same number of negative and positive nodes in each minibatch of size 16; thus positive nodes are frequently re-sampled. All methods are trained for 30 epochs and the final F\(_1\) score is reported (no hyperparameters to validate are used, thus we do not require a validation set). This procedure is repeated for four subtrees of different sizes.

**Results.** Quantitative results are presented in fig. 3.11. We can see that the hyperbolic MLR overpasses its Euclidean variants in almost all settings, sometimes by a large margin. Moreover, to provide further understanding, we plot the 2-dimensional embeddings and the trained separation hyperplanes (geodesics in this case) in fig. 3.10.

### 3.4 Additional Related Work

Recent *supervised methods* can be applied to embed any tree or directed acyclic graph in a low dimensional space with the aim of improving link prediction either by imposing a partial order in the embedding space \([AW18]; [Ven+15]; [Vil+18]\), by using hyperbolic geometry \([NK17b]; [NK18]\), or both.

To learn *word embeddings* that exhibit hypernymy or hierarchical information, supervised methods \([Ngu+17]; [VM18]\) leverage external information (e.g. WordNet) together with raw text corpora. However, the same goal is also targeted by more ambitious fully unsupervised models which move away from the “point” assumption and learn various probability densities for each word \([AW17]; [MC18]; [Sin+18]; [VM15]\).
### Hyperbolic Spaces: Embeddings & Classification

<table>
<thead>
<tr>
<th>WordNet subtree</th>
<th>Model</th>
<th>d = 2</th>
<th>d = 3</th>
<th>d = 5</th>
<th>d = 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>animal.n.01 3218 / 798</td>
<td>Hyp</td>
<td>$47.43 \pm 1.07$</td>
<td>$91.92 \pm 0.61$</td>
<td>$98.07 \pm 0.55$</td>
<td>$99.26 \pm 0.59$</td>
</tr>
<tr>
<td></td>
<td>Eucl</td>
<td>$41.69 \pm 0.19$</td>
<td>$68.43 \pm 3.90$</td>
<td>$95.59 \pm 1.18$</td>
<td>$99.36 \pm 0.18$</td>
</tr>
<tr>
<td></td>
<td>log$_0$</td>
<td>$38.89 \pm 0.01$</td>
<td>$62.57 \pm 0.61$</td>
<td>$89.21 \pm 1.34$</td>
<td>$98.27 \pm 0.70$</td>
</tr>
<tr>
<td>group.n.01 6649 / 1727</td>
<td>Hyp</td>
<td>$81.72 \pm 0.17$</td>
<td>$89.87 \pm 2.73$</td>
<td>$87.89 \pm 0.80$</td>
<td>$91.91 \pm 3.07$</td>
</tr>
<tr>
<td></td>
<td>Eucl</td>
<td>$61.13 \pm 0.42$</td>
<td>$63.56 \pm 1.22$</td>
<td>$67.82 \pm 0.81$</td>
<td>$91.38 \pm 1.19$</td>
</tr>
<tr>
<td></td>
<td>log$_0$</td>
<td>$60.75 \pm 0.24$</td>
<td>$61.98 \pm 0.57$</td>
<td>$67.92 \pm 0.74$</td>
<td>$91.41 \pm 0.18$</td>
</tr>
<tr>
<td>worker.n.01 861 / 254</td>
<td>Hyp</td>
<td>$12.68 \pm 0.82$</td>
<td>$24.09 \pm 1.49$</td>
<td>$55.46 \pm 5.49$</td>
<td>$66.83 \pm 11.38$</td>
</tr>
<tr>
<td></td>
<td>Eucl</td>
<td>$10.86 \pm 0.01$</td>
<td>$22.39 \pm 0.04$</td>
<td>$35.23 \pm 3.16$</td>
<td>$47.29 \pm 3.93$</td>
</tr>
<tr>
<td></td>
<td>log$_0$</td>
<td>$9.04 \pm 0.06$</td>
<td>$22.57 \pm 0.20$</td>
<td>$26.47 \pm 0.78$</td>
<td>$36.66 \pm 2.74$</td>
</tr>
<tr>
<td>mammal.n.01 953 / 228</td>
<td>Hyp</td>
<td>$32.01 \pm 17.14$</td>
<td>$87.54 \pm 4.55$</td>
<td>$88.73 \pm 3.22$</td>
<td>$91.37 \pm 6.09$</td>
</tr>
<tr>
<td></td>
<td>Eucl</td>
<td>$15.58 \pm 0.04$</td>
<td>$44.68 \pm 1.87$</td>
<td>$59.35 \pm 1.31$</td>
<td>$77.76 \pm 5.08$</td>
</tr>
<tr>
<td></td>
<td>log$_0$</td>
<td>$13.10 \pm 0.13$</td>
<td>$44.89 \pm 1.18$</td>
<td>$52.51 \pm 0.85$</td>
<td>$56.11 \pm 2.21$</td>
</tr>
</tbody>
</table>

Figure 3.11: Test F1 classification scores (%) for four different subtrees of WordNet noun tree. 95% confidence intervals for 3 different runs are shown for each method and each dimension. “Hyp” denotes our hyperbolic MLR, “Eucl” denotes directly applying Euclidean MLR to hyperbolic embeddings in their Euclidean parametrization, and log$_0$ denotes applying Euclidean MLR in the tangent space at 0, after projecting all hyperbolic embeddings there with log$_0$. Implementation of these experiments was done by co-authors in [GBH18].

There have been two very recent attempts at learning unsupervised word embeddings in the hyperbolic space [Dhi+18]; [LW18]. However, they suffer from either not being competitive on standard tasks in high dimensions, not showing the benefit of using hyperbolic spaces to model antisymmetric relations, or not being trained on realistically large corpora. We address these problems and, moreover, the connection with density based methods is made explicit and leveraged to improve hypernymy detection.
3.5 CONCLUSIVE SUMMARY

We propose to adapt the GloVe algorithm to hyperbolic spaces and to leverage a connection between statistical manifolds of Gaussian distributions and hyperbolic geometry, in order to better interpret entailment relations between hyperbolic embeddings. We justify the choice of products of hyperbolic spaces via this connection to Gaussian distributions and via computations of the hyperbolicity of the symbolic data upon which GloVe is based. Empirically we present the first model that can simultaneously obtain state-of-the-art results or close on the three tasks of word similarity, analogy and hypernymy detection. While there is no single model that outperforms all the baselines on all presented tasks, one can remark that the model $50 \times 2D, h(x) = x^2$, with the initialization trick obtains state-of-the-art results on hypernymy detection and is close to the best models for similarity and analogy (also Poincaré Glove models), but almost constantly outperforming the vanilla GloVe baseline on these. This is the first model that can achieve competitive results on all these three tasks, additionally offering interpretability via the connection to Gaussian word embeddings. Future work includes jointly learning the curvature of the model, together with the $h$ function defining the geometry from co-occurrence counts, as well as re-running experiments in the hyperboloid model, which has been reported to lead to lesser numerical instabilities. We showed how classic Euclidean deep learning tools such as MLR, FFNN, RNN or GRU can be generalized in a principled manner to all spaces of constant negative curvature combining Riemannian geometry with the elegant theory of gyrovector spaces. Empirically we found that our models outperform or are on par with corresponding Euclidean architectures on sequential data with implicit hierarchical structure. We hope to trigger exciting future research related to better understanding of the hyperbolic non-convexity spectrum and development of other non-Euclidean deep learning methods.
In this chapter, we discuss generalizing several celebrated adaptive algorithms from the Euclidean domain to certain Riemannian settings. We start by discussing the natural difficulties and obstacles in doing so, before defining a realistic setting for generalization. We provide new algorithms, together with convergence guarantees which recover their already known Euclidean analogue, in the particular case where the selected Riemannian space is the Euclidean space. We obtain faster convergence and to a lower train loss value for our Riemannian adaptive methods compared to their corresponding baselines on the realistic task of embedding the WordNet taxonomy into the Poincaré ball. More detailed context is provided in section 1.3.4.

**Our initial motivation.** The particular application that motivated us in developing Riemannian versions of AdaGrad and Adam was the learning of symbolic embeddings in non-Euclidean spaces. As an example, the GloVe algorithm [PSM14] — an unsupervised method for learning Euclidean word embeddings capturing semantic/syntactic relationships — benefits significantly from optimizing with AdaGrad compared to using Stochastic Gradient Descent (SGD), presumably because different words are sampled at different frequencies. Hence the absence of Riemannian adaptive algorithms could constitute a significant obstacle to the development of competitive optimization-based Riemannian embedding methods.

**Impact.** Our methods were published at ICLR 2019 [BG19], have been incorporated into the PyTorch GeoOpt package [KKK20], and contributed to obtain new state-of-the-art in a variety of tasks, from word embeddings [TBG19] to graph neural networks [LNK19].
4.1 Preliminaries and Notations

4.1.1 Elements of Geometry

We recall here some elementary notions of differential geometry. For more in-depth expositions, we refer the interested reader to [Spi79] and [RS11].

**Manifold, Tangent Space, Riemannian Metric.** A manifold $\mathcal{M}$ of dimension $n$ is a space that can locally be approximated by a Euclidean space $\mathbb{R}^n$, and which can be understood as a generalization to higher dimensions of the notion of surface. For instance, the sphere $S := \{ x \in \mathbb{R}^n | \|x\|_2 = 1 \}$ embedded in $\mathbb{R}^n$ is an $(n-1)$-dimensional manifold. In particular, $\mathbb{R}^n$ is a very simple $n$-dimensional manifold, with zero curvature. At each point $x \in \mathcal{M}$, one can define the tangent space $T_x\mathcal{M}$, which is an $n$-dimensional vector space and can be seen as a first order local approximation of $\mathcal{M}$ around $x$. A Riemannian metric $\rho$ is a collection $\rho := (\rho_x)_{x \in \mathcal{M}}$ of inner-products $\rho_x(\cdot, \cdot) : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ on $T_x\mathcal{M}$, varying smoothly with $x$. It defines the geometry locally on $\mathcal{M}$. For $x \in \mathcal{M}$ and $u \in T_x\mathcal{M}$, we also write $\|u\|_x := \sqrt{\rho_x(u, u)}$. A Riemannian manifold is a pair $(\mathcal{M}, \rho)$.

**Induced Distance Function, Geodesics.** Notice how a choice of a Riemannian metric $\rho$ induces a natural global distance function on $\mathcal{M}$. Indeed, for $x, y \in \mathcal{M}$, we can set $d(x, y)$ to be equal to the infimum of the lengths of smooth paths between $x$ and $y$ in $\mathcal{M}$, where the length $\ell(c)$ of a path $c$ is given by integrating the size of its speed vector $\dot{c}(t) \in T_{c(t)}\mathcal{M}$, in the corresponding tangent space: $\ell(c) := \int_{t=0}^{1} \|\dot{c}(t)\|_{c(t)} dt$. A geodesic $\gamma$ in $(\mathcal{M}, \rho)$ is a smooth curve $\gamma : (a, b) \to \mathcal{M}$ which locally has minimal length. In particular, a shortest path between two points in $\mathcal{M}$ is a geodesic.

**Exponential and Logarithmic Maps.** Under some assumptions, one can define at point $x \in \mathcal{M}$ the exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$. Intuitively, this map folds the tangent space on the manifold. Locally, if $v \in T_x\mathcal{M}$, then for small $t$, $\exp_x(tv)$ tells us how to move in $\mathcal{M}$ as to take a shortest path from $x$ with initial direction $v$. In $\mathbb{R}^n$, $\exp_x(v) = x + v$. In some cases, one can also define the logarithmic map $\log_x : \mathcal{M} \to T_x\mathcal{M}$ as the inverse of $\exp_x$. 

4.1 Preliminaries and Notations

Parallel Transport. In the Euclidean space, if one wants to transport a vector \( v \) from \( x \) to \( y \), one simply translates \( v \) along the straight-line from \( x \) to \( y \). In a Riemannian manifold, the resulting transported vector will depend on which path was taken from \( x \) to \( y \). The parallel transport \( P_x(v; w) \) of a vector \( v \) from a point \( x \) in the direction \( w \) and in a unit time, gives a canonical way to transport \( v \) with zero acceleration along a geodesic starting from \( x \), with initial velocity \( w \).

4.1.2 Riemannian Optimization

Consider performing an SGD update of the form

\[
    x_{t+1} \leftarrow x_t - \alpha g_t,
\]

where \( g_t \) denotes the gradient of objective \( f^t \) and \( \alpha > 0 \) is the step-size. In a Riemannian manifold \( (\mathcal{M}, \rho) \), for smooth \( f : \mathcal{M} \to \mathbb{R} \), [Bon13] defines Riemannian SGD by the following update:

\[
    x_{t+1} \leftarrow \exp_{x_t}(-\alpha g_t),
\]

where \( g_t \in T_{x_t} \mathcal{M} \) denotes the Riemannian gradient of \( f_t \) at \( x_t \). Note that when \( (\mathcal{M}, \rho) \) is the Euclidean space \( (\mathbb{R}^n, I_n) \), these two match, since we then have \( \exp_x(v) = x + v \).

Intuitively, applying the exponential map enables to perform an update along the shortest path in the relevant direction in unit time, while remaining in the manifold.

In practice, when \( \exp_x(v) \) is not known in closed-form, it is common to replace it by a retraction map \( R_x(v) \), most often chosen as \( R_x(v) = x + v \), which is a first-order approximation of \( \exp_x(v) \).

4.1.3 AMSGrad, Adam, Adagrad

Let’s recall here the main algorithms that we are taking interest in.

---

\(^1\)To be interpreted as the objective with the same parameters, evaluated at the minibatch taken at time \( t \).
ADAGRAD. Introduced by [DHS11], the standard form of its update step is defined as\(^2\)

\[
x_{t+1}^{i} \leftarrow x_t^i - \alpha g_t^i / \sqrt{\sum_{k=1}^{t} (g_k^i)^2}.
\]  \(4.3\)

Such updates rescaled coordinate-wise depending on the size of past gradients can yield huge improvements when gradients are sparse, or in deep networks where the size of a good update may depend on the layer. However, the accumulation of all past gradients can also slow down learning.

ADAM. Proposed by [KB15], the ADAM update rule is given by

\[
x_{t+1}^i \leftarrow x_t^i - \alpha m_t^i / \sqrt{v_t^i},
\]  \(4.4\)

where \(m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t\) can be seen as a momentum term and \(v_t = \beta_2 v_{t-1} + (1 - \beta_2) (g_t)^2\) is an adaptivity term. When \(\beta_1 = 0\), one essentially recovers the unpublished method RMSProp [TH12], the only difference to ADAGRAD being that the sum is replaced by an exponential moving average, hence past gradients are forgotten over time in the adaptivity term \(v_t\). This circumvents the issue of ADAGRAD that learning could stop too early when the sum of accumulated squared gradients is too significant. Let us also mention that the momentum term introduced by ADAM for \(\beta_1 \neq 0\) has been observed to often yield huge empirical improvements.

AMSGRAD. More recently, [RKK18] identified a mistake in the convergence proof of ADAM. To fix it, they proposed to either modify the ADAM algorithm with\(^3\)

\[
x_{t+1}^i \leftarrow x_t^i - \alpha m_t^i / \sqrt{\hat{v}_t^i}, \text{ where } \hat{v}_t^i = \max\{\hat{v}_{t-1}^i, v_t^i\},
\]  \(4.5\)

which they coin AMSGRAD, or to choose an increasing schedule for \(\beta_2\), making it time dependent, which they call ADAMNC (for non-constant).

---

\(^2\)A small \(\epsilon = 10^{-8}\) is often added in the square-root for numerical stability, omitted here for simplicity.

\(^3\)With \(m_t\) and \(v_t\) defined by the same equations as in ADAM (see above paragraph).
\section*{4.2 Adaptive Schemes in Riemannian Manifolds}

\subsection*{4.2.1 Obstacles in the General Setting}

**Intrinsic Updates.** It is easily understandable that writing any coordinate-wise update requires the choice of a coordinate system. However, on a Riemannian manifold \((\mathcal{M}, \rho)\), one is generally not provided with a canonical coordinate system. The formalism only allows to work with certain local coordinate systems, also called *charts*, and several different charts can be defined around each point \(x \in \mathcal{M}\). One usually says that a quantity defined using a chart is *intrinsic* to \(\mathcal{M}\) if its definition *does not depend* on which chart was used. For instance, it is known that the Riemannian gradient \(\text{grad} f\) of a smooth function \(f : \mathcal{M} \rightarrow \mathbb{R}\) can be defined intrinsically to \((\mathcal{M}, \rho)\), but its Hessian is only intrinsically defined at critical points\(^4\). It is easily seen that the RSGD update of Eq. (4.2) is intrinsic, since it only involves \(\exp\) and \(\text{grad}\), which are objects intrinsic to \((\mathcal{M}, \rho)\). However, it is unclear whether it is possible at all to express either of Eqs. (4.3,4,4.4,5) in a coordinate-free or intrinsic manner.

**A Tempting Solution.** Note that since an update is defined in a tangent space, one could be tempted to fix a canonical coordinate system \(e := (e^{(1)}, ..., e^{(n)})\) in the tangent space \(T_{x_0} \mathcal{M} \simeq \mathbb{R}^d\) at the initialization \(x_0 \in \mathcal{M}\), and parallel-transport \(e\) along the optimization trajectory, adapting Eq. (4.3) to:

\[
x_{t+1} \leftarrow \exp_{x_t}(\Delta_t), \quad e_{t+1} \leftarrow P_{x_t}(e_t; \Delta_t), \quad \text{with } \Delta_t := -\alpha g_t \odot \sqrt{\sum_{k=1}^{l} (g_k)^2},
\]

(4.6)

where \(\odot\) and \((\cdot)^2\) denote coordinate-wise division and square respectively, *these operations being taken relatively to coordinate system \(e_t\).* In the Euclidean space, parallel transport between two points \(x\) and \(y\) does not depend on the path it is taken along because the space has no curvature. However, in a general Riemannian manifold, not only does it depend on the chosen path but curvature will also give to parallel transport a rotational component\(^5\), which will almost surely break the

\(^4\)because the Poisson bracket cancels at critical points [Mil63, part 1.2].

\(^5\)The rotational component of parallel transport inherited from curvature is called the *holonomy.*
sparsity of the gradients and hence the benefit of adaptivity. Besides, the interpretation of adaptivity as optimizing different features (i.e. gradient coordinates) at different speeds is also completely lost here, since the coordinate system used to represent gradients depends on the optimization path. Finally, note that the techniques we used to prove our theorems would not apply to updates defined in the vein of Eq. (4.6).

4.2.2 Adaptivity Across Manifolds in a Cartesian Product

From now on, we assume additional structure on \((M, \rho)\), namely that it is the cartesian product of \(n\) Riemannian manifolds \((M_i, \rho^i)\), where \(\rho\) is the induced product metric:

\[
M := M_1 \times \cdots \times M_n, \quad \rho := \begin{bmatrix} \rho^1 & & \\ & \ddots & \\ & & \rho^n \end{bmatrix}.
\] (4.7)

**Product Notations.** The induced distance function \(d\) on \(M\) is known to be given by \(d(x, y)^2 = \sum_{i=1}^{n} d^i(x^i, y^i)^2\), where \(d^i\) is the distance in \(M_i\). The tangent space at \(x = (x^1, \ldots, x^n)\) is given by \(T_xM = T_{x^1}M_1 \oplus \cdots \oplus T_{x^n}M_n\), and the Riemannian gradient \(g\) of a smooth function \(f : M \to \mathbb{R}\) at point \(x \in M\) is simply the concatenation \(g = ((g^1)^T \cdots (g^n)^T)^T\) of the Riemannian gradients \(g^i \in T_{x^i}M_i\) of each partial map \(f^i : y \in M_i \mapsto f(x^1, \ldots, x^{i-1}, y, x^{i+1}, \ldots, x^n)\). Similarly, the exponential, log map and the parallel transport in \(M\) are the concatenations of those in each \(M_i\).

**Riemannian Adagrad.** We just saw in the above discussion that designing meaningful adaptive schemes — intuitively corresponding to one learning rate per coordinate — in a general Riemannian manifold was difficult, because of the absence of intrinsic coordinates. Here, we propose to see each component \(x^i \in M^i\) of \(x\) as a “coordinate”, yielding a simple adaptation of Eq. (4.3) as

\[
x_{i+1}^j \leftarrow \exp_{x_i^j}^i \left( -\alpha g_i^j / \sqrt{\sum_{k=1}^{l} \|g_k^i\|_{x_k}^2} \right).
\] (4.8)
ON THE ADAPTIVITY TERM. Note that we take (squared) Riemannian norms \( \|\tilde{g}_i^t\|^2_{x_i^t} = \rho^i_{x_i^t}(\tilde{g}_i^t, \tilde{g}_i^t) \) in the adaptivity term rescaling the gradient. In the Euclidean setting, this quantity is simply a scalar \((g_i^t)^2\), which is related to the size of an SGD update of the \(i\)th coordinate, rescaled by the learning rate (see Eq. (4.1)): \( |g_i^t| = |x_{t+1}^i - x_i^t|/\alpha \). By analogy, note that the size of an RSGD update in \(\mathcal{M}_i\) (see Eq. (4.2)) is given by \( d^i(x_{t+1}^i, x_i^t) = d^i(\exp^i_{x_i^t}(-\alpha g_i^t), x_i^t) = \| - \alpha^i g_i^t \|_{x_i^t} \), hence we also recover \( \|g_i^t\|_{x_i^t} = d^i(x_{t+1}^i, x_i^t)/\alpha \), which indeed suggests replacing the scalar \((g_i^t)^2\) by \( g_i^t\|_{x_i^t}^2 \) when transforming a coordinate-wise adaptive scheme into a manifold-wise adaptive one.

4.3 ALGORITHMS & CONVERGENCE GUARANTEES

In section 4.1, we briefly presented Adagrad, Adam and AMSGrad. Intuitively, Adam can be described as a combination of Adagrad with a momentum (of parameter \(\beta_1\)), with the slight modification that the sum of the past squared-gradients is replaced with an exponential moving average, for an exponent \(\beta_2\). Let’s also recall that AMSGrad implements a slight modification of Adam, allowing to correct its convergence proof. Finally, ADAMNC is simply Adam, but with a particular non-constant schedule for \(\beta_1\) and \(\beta_2\). On the other hand, what is interesting to note is that the schedule initially proposed by [RKK18] for \(\beta_2\) in ADAMNC, namely \(\beta_{2i} := 1 - 1/t\), lets \(v_t\) recover the sum of squared-gradients of Adagrad. Hence, AdagradNC without momentum (i.e. \(\beta_{1i} = 0\)) yields Adagrad.

ASSUMPTIONS AND NOTATIONS. For \(1 \leq i \leq n\), we assume \((\mathcal{M}_i, \rho^i)\) is a geodesically complete Riemannian manifold with sectional curvature lower bounded by \(k_i \leq 0\). As written in Eq. (4.7), let \((\mathcal{M}, \rho)\) be the product manifold of the \((\mathcal{M}_i, \rho^i)\)’s. For each \(i\), let \(\mathcal{X}_i \subset \mathcal{M}_i\) be a compact, geodesically convex set and define \(\mathcal{X} := \mathcal{X}_1 \times \cdots \times \mathcal{X}_n\), the set of feasible parameters. Define \(\Pi_{\mathcal{X}_i} : \mathcal{M}_i \to \mathcal{X}_i\) to be the projection operator, i.e. \(\Pi_{\mathcal{X}_i}(x)\) is the unique \(y \in \mathcal{X}_i\) minimizing \(d^i(y, x)\). Denote by \(P^i, \exp^i\) and \(\log^i\) the parallel transport, exponential and log maps in \((\mathcal{M}_i, \rho^i)\), respectively. For \(f : \mathcal{M} \to \mathbb{R}\), if \(g = \grad f(x)\) for \(x \in \mathcal{M}\), denote by \(x^i \in \mathcal{M}_i\) and by \(g^i \in T_{x^i}\mathcal{M}_i\) the corresponding components of \(x\) and \(g\). In the sequel, let \((f_i)\) be a family of differentiable, geodesically convex functions from \(\mathcal{M}\) to \(\mathbb{R}\). Assume that each
$\mathcal{X}_i \subset \mathcal{M}_i$ has a diameter bounded by $D_\infty$ and that for all $1 \leq i \leq n$, $t \in [T]$ and $x \in \mathcal{X}$, $\|\nabla f_i(x)^j\|_{x_j} \leq G_\infty$. Finally, our convergence guarantees will bound the regret, defined at the end of $T$ rounds as $R_T = \sum_{t=1}^T f_t(x_t) - \min_{x \in \mathcal{X}} \sum_{j=1}^T f_j(x)$, so that $R_T = o(T)$. Finally, $\phi_{x^i \rightarrow y^j}$ denotes any isometry from $T_{x^i} \mathcal{M}_i$ to $T_{y^j} \mathcal{M}_i$, for $x^i, y^j \in \mathcal{M}_i$.

Following the discussion in section 4.2.2 and especially Eq. (4.8), we present Riemannian AMSGRAD in Figure 4.1a. For comparison, we show next to it the standard AMSGRAD algorithm in Figure 4.1b.

Require: $x_1 \in \mathcal{X}$, $\{\alpha_t\}_{t=1}^T$, $\{\beta_t\}_{t=1}^T$, $\beta_2$

Set $m_0 = 0$, $\tau_0 = 0$, $v_0 = 0$ and $\delta_0 = 0$

for $t = 1$ to $T$ do (for all $1 \leq i \leq n$)

\begin{align*}
    g_t & = \nabla f_i(x_t) \\
    m_t^i & = \beta_{11} \tau_{t-1}^i + (1 - \beta_{11}) g_t^i \\
    v_t^i & = \beta_2 v_{t-1}^i + (1 - \beta_2) \|g_t^i\|_{x_t}^2 \\
    \delta_t^i & = \max\{\delta_{t-1}^i, v_t^i\} \\
    x_{t+1}^i & = \Pi_{\mathcal{X}_i}(\exp_{x_t^i}(\frac{-\alpha_t m_t^i}{\sqrt{\delta_t^i}})) \\
    \tau_t^i & = \phi_{x_t^i \rightarrow x_{t+1}^i}(m_t^i)
\end{align*}

end for

(a) Ramsgard in $\mathcal{M}_1 \times \cdots \times \mathcal{M}_n$.

Require: $x_1 \in \mathcal{X}$, $\{\alpha_t\}_{t=1}^T$, $\{\beta_t\}_{t=1}^T$, $\beta_2$

Set $m_0 = 0$, $v_0 = 0$ and $\delta_0 = 0$

for $t = 1$ to $T$ do (for all $1 \leq i \leq n$)

\begin{align*}
    g_t & = \nabla f_i(x_t) \\
    m_t^i & = \beta_{11} m_{t-1}^i + (1 - \beta_{11}) g_t^i \\
    v_t^i & = \beta_2 v_{t-1}^i + (1 - \beta_2) (g_t^i)^2 \\
    \delta_t^i & = \max\{\delta_{t-1}^i, v_t^i\} \\
    x_{t+1}^i & = \Pi_{\mathcal{X}_i}(x_t^i - \alpha_t m_t^i / \sqrt{\delta_t^i})
\end{align*}

end for

(b) AMSGRAD in $\mathbb{R}^n$.

Figure 4.1: Comparison of the Riemannian and Euclidean versions of AMSGRAD.

Write $h_t^i := -\alpha_t m_t^i / \sqrt{\delta_t^i}$. As a natural choice for $\phi^i$, one could first parallel-transport\footnote{The idea of parallel-transporting $m_t$ from $T_{x_t^i} \mathcal{M}$ to $T_{x_{t+1}^i} \mathcal{M}$ previously appeared in [CL17].} $m_t^i$ from $x_t^i$ to $\tilde{x}_{t+1}^i := \exp_{x_t^i}(h_t^i)$ using $P^i(\cdot; h_t^i)$, and then from $\tilde{x}_{t+1}^i$ to $x_{t+1}^i$ along a minimizing geodesic.

As can be seen, if $(\mathcal{M}_i, \rho_i) = \mathbb{R}$ for all $i$, RAMSGRAD and AMSGRAD coincide: we then have $x_i = 0$, $d^i(x^i, y^j) = |x^i - y^j|$, $\phi^i = Id$, $\exp_{x_i}^i(v^i) = x^i + v^i$, $\mathcal{M}_1 \times \cdots \times \mathcal{M}_n = \mathbb{R}^n$, $\|g_t^i\|_{x_t^i}^2 = (g_t^i)^2 \in \mathbb{R}$. From
these algorithms, RADAM and ADAM are obtained simply by removing the max operations, i.e. replacing \( \hat{v}_i^t = \max\{\hat{v}_{i-1}^t, v_i^t\} \) with \( \hat{v}_i^t = v_i^t \).

### 4.3.1 RAMSGRAD

The convergence guarantee that we obtain for RAMSGRAD is presented in Theorem 4.1, where the quantity \( \zeta \) is defined by [ZS16] as

\[
\zeta(\kappa, c) := \frac{c\sqrt{|\kappa|}}{\tanh(c\sqrt{|\kappa|})} = 1 + \frac{c}{3}|\kappa| + O_{\kappa \to 0}(\kappa^2). \tag{4.9}
\]

For comparison, we also show the convergence guarantee of the original AMSGRAD in Section 4.7. Note that when \((M_i, \rho_i) = \mathbb{R}\) for all \(i\), convergence guarantees between RAMSGRAD and AMSGRAD coincide as well. Indeed, the curvature dependent quantity \( (\zeta(\kappa_i, D_{\infty}) + 1)/2 \) in the Riemannian case then becomes equal to 1, recovering the convergence theorem of AMSGRAD. It is also interesting to understand at which speed does the regret bound worsen when the curvature is small but non-zero: by a multiplicative factor of approximately \( 1 + D_{\infty}|\kappa|/6 \) (see Eq.(4.9)). Similar remarks hold for RADAMNC, whose convergence guarantee is shown in Theorem 4.2. Finally, notice that \( \beta_1 := 0 \) in Theorem 4.2 yields a convergence proof for RADAGRAD, whose update rule we defined in Eq. (4.8).

**Theorem 4.1 (Convergence of RAMSGRAD).** Let \((x_i)\) and \((\hat{v}_i)\) be the sequences obtained from Algorithm 4.1a, \(\alpha_t = \alpha/\sqrt{t}, \beta_1 = \beta_{11}, \beta_{1t} \leq \beta_1\) for all \(t \in [T]\) and \(\gamma = \beta_1/\sqrt{\beta_2} < 1\). We then have:

\[
R_T \leq \frac{\sqrt{T}D_{\infty}^2}{2\alpha(1-\beta_1)} \sum_{i=1}^n \sqrt{\hat{v}_i^t} + \frac{D_{\infty}^2}{2(1-\beta_1)} \sum_{i=1}^n \beta_{1t} \sqrt{\hat{v}_i^t} + \frac{\alpha\sqrt{1 + \log T}}{(1-\beta_1)^2(1-\gamma)\sqrt{1-\beta_2}} \sum_{i=1}^n \frac{\zeta(\kappa_i, D_{\infty}) + 1}{2} \sqrt{\|g_i^t\|_{x_i^t}^2}. \tag{4.10}
\]

**Proof.** Denote by \( \bar{x}_{i+1}^i := \exp_{x_i^t}^{\alpha_i m_i^t / \sqrt{\hat{v}_i^t}} \) and consider the geodesic triangle defined by \( \bar{x}_{i+1}^i, x_i^t \) and \( x_i^* \). Now let \( a = d^i(\bar{x}_{i+1}^i, x_i^*), b = \)
Adaptive optimization in non-Euclidean geometry

d^i(\tilde{x}_{i+1}^i, x_i^i), c = d^i(x_i^i, x_i^*) and A = \angle \tilde{x}_{i+1}^i x_i^i x_i^*. Combining the following formula\(^7\):

\[
d^i(x_i^i, \tilde{x}_{i+1}^i) d^i(x_i^i, x_i^*) \cos(\tilde{x}_{i+1}^i x_i^i) = \langle -\alpha_t m_i^t / \sqrt{\Delta_i}, \log_{x_i^i}^{i}(x_i^*) \rangle_{x_i^i},
\]

with the following inequality (given by lemma 4.3):

\[
a^2 \leq \zeta(\kappa, c) b^2 + c^2 - 2bc \cos(A), \quad \text{with} \quad \zeta(\kappa, c) := \frac{\sqrt{|\kappa|} c}{\tanh(\sqrt{|\kappa|} c)},
\]

yields

\[
\langle -m_i^t, \log_{x_i^i}^{i}(x_i^*) \rangle_{x_i^i} \leq \frac{\sqrt{\Delta_i}}{2 \alpha_t} \left( d^i(x_i^i, x_i^*)^2 - d^i(\tilde{x}_{i+1}^i, x_i^*)^2 \right)
\]

\[
+ \zeta(\kappa, d^i(x_i^i, x_i^*)) \frac{\alpha_t}{2 \sqrt{\Delta_i}} \|m_i^t\|^2_{x_i^i}, \quad (4.13)
\]

where the use the notation \(\langle \cdot, \cdot \rangle_{x_i^i}\) for \(\rho_{x_i^i}^{i}(\cdot, \cdot)\) when it is clear which metric is used. By definition of \(\Pi_{x_i^i}\), we can safely replace \(\tilde{x}_{i+1}^i\) by \(x_{i+1}^i\) in the above inequality. Plugging \(m_i^t = \beta_{1t} q_{x_i^i \rightarrow x_i^t}^{i}(m_{i-1}^t) + (1 - \beta_{1t})g_i^t\) into Eq. (4.13) gives us

\[
\langle -g_i^t, \log_{x_i^i}^{i}(x_i^*) \rangle_{x_i^i} \leq \frac{\sqrt{\Delta_i}}{2 \alpha_t(1 - \beta_{1t})} \left( d^i(x_i^i, x_i^*)^2 - d^i(x_{i+1}^i, x_i^*)^2 \right)
\]

\[
+ \zeta(\kappa, d^i(x_i^i, x_i^*)) \frac{\alpha_t}{2(1 - \beta_{1t}) \sqrt{\Delta_i}} \|m_i^t\|^2_{x_i^i}
\]

\[
+ \frac{\beta_{1t}}{(1 - \beta_{1t})} \langle q_{x_i^i \rightarrow x_i^t}^{i}(m_{i-1}^t), \log_{x_i^i}^{i}(x_i^*) \rangle_{x_i^i}. \quad (4.14)
\]

\(^7\)Note that since each \(x_i^i\) is geodesically convex, logarithms are well-defined.
Now applying Cauchy-Schwarz' and Young’s inequalities to the last term yields

\[
\langle -g_t^i, \log x_t^i(x_t^i) \rangle_{x_t^i} \leq \frac{\sqrt{\hat{\sigma}_t^i}}{2\alpha_t(1-\beta_{1t})} \left( d^i(x_t^i, x_t^i)^2 - d^i(x_t^{i+1}, x_t^{i+1})^2 \right) \\
+ \zeta(\kappa_i, d^i(x_t^i, x_t^i)) \frac{\alpha_t}{2(1-\beta_{1t})} \|m_t^i\|_{x_t^i}^2 \\
+ \frac{\beta_{1t}}{2(1-\beta_{1t})} \frac{\alpha_t}{\sqrt{\hat{\sigma}_t^i}} \|m_{t-1}^i\|_{x_{t-1}^i}^2 + \frac{\beta_{1t}}{2(1-\beta_{1t})} \frac{\sqrt{\hat{\sigma}_t^i}}{\alpha_t} \|\log x_t^i(x_t^i)\|_{x_t^i}^2. \tag{4.15}
\]

From the geodesic convexity of \(f_t\) for \(1 \leq t \leq T\), we have

\[
\sum_{i=1}^{T} f_t(x_t) - f_t(x_t^*) \leq \sum_{i=1}^{T} \langle -g_t^i, \log x_t^i(x_t^i) \rangle_{x_t^i} = \sum_{i=1}^{n} \sum_{t=1}^{T} \langle -g_t^i, \log x_t^i(x_t^i) \rangle_{x_t^i}. \tag{4.16}
\]

Let’s look at the first term. Using \(\beta_{1t} \leq \beta_1\) and with a change of indices, we have

\[
\sum_{i=1}^{n} \sum_{t=1}^{T} \frac{\sqrt{\hat{\sigma}_t^i}}{2\alpha_t(1-\beta_{1t})} \left( d^i(x_t^i, x_t^i)^2 - d^i(x_t^{i+1}, x_t^{i+1})^2 \right) \tag{4.17}
\]

\[
\leq \frac{1}{2(1-\beta_1)} \left[ \sum_{i=1}^{n} \sum_{t=2}^{T} \left( \frac{\sqrt{\hat{\sigma}_t^i}}{\alpha_t} - \frac{\sqrt{\hat{\sigma}_{t-1}^i}}{\alpha_{t-1}} \right) d^i(x_t^i, x_t^i)^2 + \sum_{i=1}^{n} \frac{\sqrt{\hat{\sigma}_t^i}}{\alpha_1} d^i(x_t^i, x_t^i)^2 \right] \tag{4.18}
\]

\[
\leq \frac{1}{2(1-\beta_1)} \sum_{i=1}^{n} \sum_{t=2}^{T} \left( \frac{\sqrt{\hat{\sigma}_t^i}}{\alpha_t} - \frac{\sqrt{\hat{\sigma}_{t-1}^i}}{\alpha_{t-1}} \right) D_\infty^2 + \sum_{i=1}^{n} \frac{\sqrt{\hat{\sigma}_t^i}}{\alpha_1} D_\infty^2 \tag{4.19}
\]

\[
= \frac{D_\infty^2}{2\alpha_T(1-\beta_1)} \sum_{i=1}^{n} \frac{\sqrt{\hat{\sigma}_T^i}}{\alpha_T}, \tag{4.20}
\]

where the last equality comes from a standard telescopic summation. We now need the following lemma.

**Lemma 4.1.**

\[
\sum_{i=1}^{T} \frac{\alpha_t}{\sqrt{\hat{\sigma}_t^i}} \|m_t^i\|_{x_t^i}^2 \leq \frac{\alpha \sqrt{1 + \log T}}{(1-\beta_1)(1-\gamma) \sqrt{1-\beta_2}} \sqrt{T} \sum_{i=1}^{T} \|g_t^i\|_{x_t^i}^2 \tag{4.21}
\]
Proof. Let’s start by separating the last term, and removing the hat on \( v \).

\[
\sum_{t=1}^{T} \frac{\alpha_t}{\hat{\sigma}_t^i} \|m^i_t\|_x^2 \leq \sum_{t=1}^{T-1} \frac{\alpha_t}{\hat{\sigma}_t^i} \|m^i_t\|_x^2 + \frac{\alpha_T}{\hat{\sigma}_T^i} \|m^i_T\|_x^2 \tag{4.22}
\]

\[
\leq \sum_{t=1}^{T-1} \frac{\alpha_t}{\hat{\sigma}_t^i} \|m^i_t\|_x^2 + \frac{\alpha_T}{\hat{\sigma}_T^i} \|m^i_T\|_x^2 \tag{4.23}
\]

Let’s now have a closer look at the last term. We can reformulate \( m^i_T \) as:

\[
m^i_T = \sum_{j=1}^{T} (1 - \beta_{1j}) \left( \prod_{k=1}^{T-j} \beta_{1,(T-k+1)} \right) \varphi^i_{x^i_{j-1} \rightarrow x^i_T} \circ \cdots \circ \varphi^i_{x^i_j \rightarrow x^i_{j+1}} (g^i_j) \tag{4.24}
\]

Applying lemma 4.4, we get

\[
\|m^i_T\|_{x^i_T}^2 \leq \left( \sum_{j=1}^{T} (1 - \beta_{1j}) \left( \prod_{k=1}^{T-j} \beta_{1,(T-k+1)} \right) \right) \times \left( \sum_{j=1}^{T} (1 - \beta_{1j}) \left( \prod_{k=1}^{T-j} \beta_{1,(T-k+1)} \right) \|\varphi^i_{x^i_{j-1} \rightarrow x^i_T} \circ \cdots \circ \varphi^i_{x^i_j \rightarrow x^i_{j+1}} (g^i_j)\|_{x^i_T}^2 \right). \tag{4.25}
\]

Since \( \varphi^i \) is an isometry, we always have \( \|\varphi^i_{x^i \rightarrow y}(u)\|_y = \|u\|_x \), i.e.

\[
\|\varphi^i_{x^i_{j-1} \rightarrow x^i_T} \circ \cdots \circ \varphi^i_{x^i_j \rightarrow x^i_{j+1}} (g^i_j)\|_{x^i_T}^2 = \|g^i_j\|_{x^i_j}^2. \tag{4.26}
\]

Using that \( \beta_{1k} \leq \beta_1 \) for all \( k \in [T] \),

\[
\|m^i_T\|_{x^i_T}^2 \leq \left( \sum_{j=1}^{T} (1 - \beta_{1j}) \beta_1^{T-j} \right) \left( \sum_{j=1}^{T} (1 - \beta_{1j}) \beta_1^{T-j} \|g^i_j\|_{x^i_j}^2 \right). \tag{4.27}
\]

Finally, \((1 - \beta_{1j}) \leq 1\) and \( \sum_{j=1}^{T} \beta_1^{T-j} \leq 1/(1 - \beta_1) \) yield

\[
\|m^i_T\|_{x^i_T}^2 \leq \frac{1}{1 - \beta_1} \left( \sum_{j=1}^{T} \beta_1^{T-j} \|g^i_j\|_{x^i_j}^2 \right). \tag{4.28}
\]

Let’s now look at \( v^i_T \). It is given by

\[
v^i_T = (1 - \beta_2) \sum_{j=1}^{T} \beta_2^{T-j} \|g^i_j\|_{x^i_j}^2. \tag{4.29}
\]
Combining Eq. (4.28) and Eq. (4.29) allows us to bound the last term of Eq. (4.22):

\[
\frac{\alpha}{\sqrt{v_T}} \frac{\|m_T\|^2_{x'_i}}{\sum_{i=1}^{T} \beta_1^{T-j} \|g_j^i\|^2_{x'_i}} \leq \frac{\alpha}{(1 - \beta_1) \sqrt{T}} \frac{\left( \sum_{j=1}^{T} \beta_1^{T-j} \|g_j^i\|^2_{x'_i} \right)}{(1 - \beta_2) \sum_{j=1}^{T} \beta_2^{T-j} \|g_j^i\|^2_{x'_i}}
\]

(4.30)

\[
\leq \frac{\alpha}{(1 - \beta_1) \sqrt{T}} \sum_{j=1}^{T} \frac{\left( \beta_1^{T-j} \|g_j^i\|^2_{x'_i} \right)}{(1 - \beta_2) \beta_2^{T-j} \|g_j^i\|^2_{x'_i}}
\]

(4.31)

\[
= \frac{\alpha}{(1 - \beta_1) \sqrt{T} (1 - \beta_2)} \sum_{j=1}^{T} \gamma^{T-j} \|g_j^i\|_{x'_i}
\]

(4.32)

With this inequality, we can now bound every term of Eq. (4.22):

\[
\sum_{i=1}^{T} \frac{\alpha}{\sqrt{\hat{v}_T}} \|m_i^T\|^2_{x'_i} \leq \sum_{i=1}^{T} \frac{\alpha}{(1 - \beta_1) \sqrt{i(1 - \beta_2)}} \sum_{j=1}^{T} \gamma^{T-j} \|g_j^i\|_{x'_i}
\]

(4.33)

\[
= \frac{\alpha}{(1 - \beta_1) \sqrt{1 - \beta_2}} \sum_{i=1}^{T} \frac{1}{\sqrt{i}} \sum_{j=1}^{T} \gamma^{T-j} \|g_j^i\|_{x'_i}
\]

(4.34)

\[
= \frac{\alpha}{(1 - \beta_1) \sqrt{1 - \beta_2}} \sum_{i=1}^{T} \|g_i^i\|_{x'_i} \sum_{j=1}^{T} \gamma^{j-t} / \sqrt{j}
\]

(4.35)

\[
\leq \frac{\alpha}{(1 - \beta_1) \sqrt{1 - \beta_2}} \sum_{i=1}^{T} \|g_i^i\|_{x'_i} \sum_{j=1}^{T} \gamma^{j-t} / \sqrt{t}
\]

(4.36)

\[
\leq \frac{\alpha}{(1 - \beta_1) \sqrt{1 - \beta_2}} \sum_{i=1}^{T} \|g_i^i\|_{x'_i} \frac{1}{(1 - \gamma) \sqrt{t}}
\]

(4.37)

\[
\leq \frac{\alpha}{(1 - \beta_1)(1 - \gamma) \sqrt{1 - \beta_2}} \left[ \sum_{i=1}^{T} \|g_i^i\|^2_{x'_i} \right] \left[ \sum_{i=1}^{T} \frac{1}{t} \right]
\]

(4.38)

\[
\leq \frac{\alpha}{(1 - \beta_1)(1 - \gamma) \sqrt{1 - \beta_2}} \left[ \sum_{i=1}^{T} \|g_i^i\|^2_{x'_i} \right] \left[ \sum_{i=1}^{T} \frac{1}{t} \right]
\]

(4.39)
Putting together Eqs. (4.15), (4.16), (4.20) and lemma 4.1 lets us bound the regret:

\[ \sum_{t=1}^{T} f_t(x_t) - f_t(x^*) \leq \sum_{i=1}^{n} \sum_{t=1}^{T} \langle -g_t^i, \log x_t^i(x_t^*) \rangle x_t^i \]  
(4.40)

\[ \leq \frac{\sqrt{T}D^2_\infty}{2\alpha(1-\beta_1)} \sum_{i=1}^{n} \sqrt{\hat{\sigma}_i^T} + \frac{D^2_\infty}{2(1-\beta_1)} \sum_{i=1}^{n} \sum_{t=1}^{T} \beta_{1t} \frac{\sqrt{\hat{\sigma}_i^t}}{\alpha_t} \]  
(4.41)

\[ + \frac{\alpha \sqrt{1 + \log T}}{(1-\beta_1)^2(1-\gamma)} \sum_{i=1}^{n} \zeta(\kappa_i, D_\infty) + 1 \sum_{i=1}^{n} \beta_2 \|g_t^i\|^2_{x_t^i} \]  
(4.42)

where we used the facts that \( d \mapsto \zeta(\kappa, d) \) is an increasing function, and that \( \alpha_t/\sqrt{\hat{\sigma}_i^t} \leq \alpha_{t-1}/\sqrt{\hat{\sigma}_{i-1}^t} \), which enables us to bound both the second and third terms of the right-hand side of Eq. (4.15) using lemma 4.1. 

\[ \square \]

**Remark.** Let us notice that similarly as for AMSGRAD, RAMSGRAD also has a regret bounded by \( O(G_\infty \sqrt{T}) \). This is easy to see from the proof of lemma 4.2. Hence the actual upper-bound on the regret is a minimum between the one in \( O(G_\infty \sqrt{T}) \) and the one of Theorem 4.1.

### 4.3.2 RADAMNC

**Theorem 4.2 (Convergence of RADAMNC).** Let \( (x_t) \) and \( (v_t) \) be the sequences obtained from RADAMNC, \( \alpha_t = \alpha/\sqrt{t} \), \( \beta_1 = \beta_{11} \), \( \beta_{1t} = \beta_1 \lambda^{t-1} \), \( \lambda < 1 \), \( \beta_{2t} = 1 - 1/t \). We then have:

\[
R_T \leq \sum_{i=1}^{n} \left( \frac{D_\infty}{2\alpha(1-\beta_1)} + \frac{\alpha(\zeta(\kappa_i, D_\infty) + 1)}{(1-\beta_1)^3} \right) \sqrt{\sum_{t=1}^{T} \|g_t^i\|^2_{x_t^i}} + \frac{\beta_1 D^2_\infty G_\infty n}{2\alpha(1-\beta_1)(1-\lambda)^2}. \]  
(4.43)
Proof. Similarly as for the proof of Theorem 4.1 (and with same notations), we obtain the inequality:

\[
\langle -g_i, \log_{x_i^t}(x_i^*) \rangle_{x_i^t} \leq \frac{\sqrt{v_i}}{2\alpha_t(1-\beta_{11})} \left( d'(x_i^t, x_i^*)^2 - d'(x_{i+1}^t, x_i^*)^2 \right) + \zeta(\nu_i, d(x_i^t, x_i^*)) \frac{\alpha_t}{2(1-\beta_{11})} \|m_i^t\|_{x_i^t}^2 \\
+ \frac{\beta_{11}}{2(1-\beta_{11})} \frac{\alpha_t}{\sqrt{v_i}} \|m_{i-1}^t\|_{x_{i-1}^t}^2 + \frac{\beta_{11}}{2(1-\beta_{11})} \frac{\sqrt{v_i}}{\alpha_t} \|\log_{x_i^t}(x_i^*)\|_{x_i^t}^2.
\] (4.44)

From the geodesic convexity of \( f_t \) for \( 1 \leq t \leq T \), we have

\[
\sum_{i=1}^{n} \sum_{t=1}^{T} \langle -g_i, \log_{x_i^t}(x_i^*) \rangle_{x_i^t} = \sum_{i=1}^{n} \sum_{t=1}^{T} \langle -g_i^i, \log_{x_i^t}(x_i^*) \rangle_{x_i^t}.
\] (4.45)

With the same techniques as before, we obtain the same bound on the first term:

\[
\sum_{i=1}^{n} \sum_{t=1}^{T} \frac{\sqrt{v_i}}{2\alpha_t(1-\beta_{11})} \left( d'(x_i^t, x_i^*)^2 - d'(x_{i+1}^t, x_i^*)^2 \right) \leq \frac{D_{\infty}^2}{2\alpha_T(1-\beta_1)} \sum_{i=1}^{n} \sqrt{v_i}.
\] (4.46)

However, for the other terms, we now need a new lemma:

**Lemma 4.2.**

\[
\sum_{t=1}^{T} \frac{\alpha_t}{\sqrt{v_i}} \|m_i^t\|_{x_i^t}^2 \leq \frac{2\alpha}{(1-\beta_1)^2} \sqrt{\sum_{t=1}^{T} \|g_i^t\|_{x_i^t}^2}.
\] (4.47)

**Proof.** Let’s start by separating the last term.

\[
\sum_{t=1}^{T} \frac{\alpha_t}{\sqrt{v_i}} \|m_i^t\|_{x_i^t}^2 \leq \sum_{i=1}^{T-1} \frac{\alpha_t}{\sqrt{v_i}} \|m_i^t\|_{x_i^t}^2 + \frac{\alpha_T}{\sqrt{v_T}} \|m_T^t\|_{x_T^t}^2.
\] (4.48)

Similarly as before, we have

\[
\|m_T^t\|_{x_T^t}^2 \leq \frac{1}{1-\beta_1} \left( \sum_{j=1}^{T} \beta_{11}^{T-j} \|g_j^i\|_{x_j^i}^2 \right).
\] (4.49)
Let’s now look at $v_i^T$. Since $\beta_{2t} = 1 - 1/t$, it is simply given by

$$v_i^T = \sum_{t=1}^{T} \|g_i^T\|_{x_i^T}^2 / T.$$  \hfill (4.50)

Combining these yields:

$$\frac{\alpha_T}{\sqrt{v_i^T}} \|m_i^T\|_{x_i^T}^2 \leq \frac{\alpha}{1 - \beta_1} \frac{\sum_{j=1}^{T} \beta_1^{T-j} \|g_j^T\|_{x_j^T}^2}{\sqrt{\sum_{i=1}^{T} \|g_i^T\|_{x_i^T}^2}} \leq \frac{\alpha}{1 - \beta_1} \sum_{j=1}^{T} \frac{\beta_1^{T-j} \|g_j^T\|_{x_j^T}^2}{\sqrt{\sum_{k=1}^{T} \|g_k^T\|_{x_k^T}^2}}.$$  \hfill (4.51)

Using this inequality at all time-steps gives

$$\sum_{t=1}^{T} \frac{\alpha_t}{\sqrt{v_i^T}} \|m_i^T\|_{x_i^T}^2 \leq \frac{\alpha}{1 - \beta_1} \sum_{j=1}^{T} \frac{\sum_{i=0}^{T-j} \beta_1^i \|g_j^T\|_{x_j^T}^2}{\sqrt{\sum_{k=1}^{T} \|g_k^T\|_{x_k^T}^2}} \leq \frac{2\alpha}{(1 - \beta_1)^2} \sqrt{T} \sum_{j=1}^{T} \|g_j^T\|_{x_j^T}^2.$$

where the last inequality comes from lemma 4.5. \hfill \Box

Putting everything together, we finally obtain

$$\sum_{t=1}^{T} f_t(x_i) - f_t(x_*) \leq \sum_{i=1}^{n} \sum_{t=1}^{T} \langle -g_i^T, \log x_i^T (x_*^T) \rangle_{x_i^T}$$

$$\leq \frac{\sqrt{T} D_\infty^2}{2\alpha (1 - \beta_1)} \sum_{i=1}^{n} \sqrt{v_i^T} + \frac{D_\infty^2}{2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{t=1}^{T} \beta_1^t \frac{\sqrt{v_i^T}}{\alpha_t}$$

$$+ \frac{\alpha}{(1 - \beta_1)^3} \sum_{i=1}^{n} (\zeta_\beta (k_i, D_\infty) + 1) \sqrt{T} \sum_{i=1}^{n} \|g_i^T\|_{x_i^T}^2 \leq \alpha_{t-1} \sqrt{v_i^{t-1}} \leq \alpha_{t-1} \sqrt{v_i^{t-1}}.$$  \hfill (4.56)

where we used that for this choice of $\alpha_t$ and $\beta_{2t}$, we have $\alpha_t / \sqrt{v_i^T} \leq \alpha_{t-1} / \sqrt{v_i^{t-1}}$. Finally,
D^2_∞ \sum_{i=1}^n \sum_{t=1}^T \beta_{1t} \frac{\sqrt{v^i_t}}{\alpha_t} \leq \frac{D^2_∞ G_∞ n}{2\alpha(1-\beta_1)} \sum_{t=1}^T \sqrt{\beta_{1t}} \leq \frac{\beta_1 D^2_∞ G_∞ n}{2\alpha(1-\beta_1)(1-\lambda)^2}.

(4.58)

This combined with Eq. (4.50) yields the final result.

Remark. Notice the appearance of a factor $n/\alpha$ in the last term of the last equation. This term is missing in corollaries 1 and 2 of [RKK18], which is a mistake. However, this dependence in $n$ is not too harmful here, since this term does not depend on $T$.

4.3.3 Intuition Behind the Proofs

The Role of Convexity. Note how the notion of convexity in Theorem 4.3 got replaced by the notion of geodesic convexity in Theorem 4.1. Let us compare the two definitions: the differentiable functions $f: \mathbb{R}^n \rightarrow \mathbb{R}$ and $g: \mathcal{M} \rightarrow \mathbb{R}$ are respectively convex and geodesically convex if for all $x, y \in \mathbb{R}^n$, $u, v \in \mathcal{M}$:

$$f(x) - f(y) \leq \langle \nabla f(x), x - y \rangle, \quad g(u) - g(v) \leq \rho_u(\nabla g(u), -\log_u(v)).$$

(4.59)

But how does this come at play in the proofs? Regret bounds for convex objectives are usually obtained by bounding $\sum_{t=1}^T f_t(x_t) - f_t(x_*)$ using Eq. (4.59) for any $x_* \in \mathcal{X}$, which boils down to bounding each $\langle g_t, x_t - x_* \rangle$. In the Riemannian case, this term becomes $\rho_{x_t}(g_t, -\log_{x_t}(x_*))$.

The Role of the Cosine Law. How does one obtain a bound on $\langle g_t, x_t - x_* \rangle$? For simplicity, let us look at the particular case of an SGD update, from Eq. (4.1). Using a cosine law, this yields

$$\langle g_t, x_t - x_* \rangle = \frac{1}{2\alpha} (\|x_t - x_*\|^2 - \|x_{t+1} - x_*\|^2) + \frac{\alpha}{2}\|g_t\|^2.$$

(4.60)

One now has two terms to bound: (i) when summing over $t$, the first one simplifies as a telescopic summation; (ii) the second term $\sum_{t=1}^T \alpha_t \|g_t\|^2$ will require a well chosen decreasing schedule for $\alpha$. In Riemannian manifolds, this step is generalized using the analogue lemma 4.3 introduced by [ZS16], valid in all Alexandrov spaces, which
includes our setting of geodesically convex subsets of Riemannian manifolds with lower bounded sectional curvature. The curvature dependent quantity \( \zeta \) of Eq. (4.10) appears from this lemma, letting us bound \( \rho_{x_i}^i (g_i^i - \log_{x_i}^i (x_i^i)) \).

**THE BENEFIT OF ADAPTIVITY.** Let us also mention that the above bounds significantly improve for sparse (per-manifold) gradients. In practice, this could happen for instance for algorithms embedding each word \( i \) (or node of a graph) in a manifold \( M_i \) and when just a few words are updated at a time.

**ON THE CHOICE OF \( \varphi^i \).** The fact that our convergence theorems (see lemma 4.1) do not require specifying \( \varphi^i \) suggests that the regret bounds could be improved by exploiting momentum/acceleration in the proofs for a particular \( \varphi^i \). Note that this remark also applies to Amsgrad [RKK18].

### 4.4 EMPIRICAL VALIDATION

We empirically assess the quality of the proposed algorithms: Radam, Rmsggrad and Radagrad compared to the non-adaptive RSGD method (Eq. 4.2). For this, we follow [NK17a] and embed the transitive closure of the WordNet noun hierarchy [Mil+90] in the \( n \)-dimensional Poincaré model \( D^n \) of hyperbolic geometry which is well-known to be better suited to embed tree-like graphs than the Euclidean space [De +18b]; [Gro87]. In this case, each word is embedded in the same space of constant curvature \(-1\), thus \( M_i = D^n, \forall i \). Note that it would also be interesting to explore the benefit of our optimization tools for algorithms proposed in [De +18b]; [GBH18b]; [NK18]. The choice of the Poincaré model is justified by the access to closed form expressions for all the quantities used in Alg. 4.1a:

- **Metric tensor:** \( \rho_x = \lambda_x^2 I_n, \forall x \in D^n \), where \( \lambda_x = \frac{2}{1 - \|x\|^2} \) is the conformal factor.

- **Riemannian gradients are rescaled Euclidean gradients:** \( \text{grad} f(x) = (1/\lambda_x^2) \nabla_E f(x) \).

- **Distance function and geodesics,** [GBH18c]; [NK17a]; [Ungo08].
4.4 Empirical Validation

- Exponential and logarithmic maps:
  \[ \exp_x(v) = x \oplus \left( \tanh \left( \frac{\lambda_x \|v\|}{2} \right) \frac{v}{\|v\|} \right), \]
  where \( \oplus \) is the generalized M"obius addition [GBH18c]; [Ung08].

- Parallel transport along the unique geodesic from \( x \) to \( y \):
  \[ P_{x \to y}(v) = \frac{\lambda_x}{\lambda_y} \text{gyr}[y, -x]v. \]
  This formula was derived from [GBH18c]; [Ung08], gyr being given in closed form in [Ung08, Eq. (1.27)].

**Dataset & Model.** The transitive closure of the WordNet taxonomy graph consists of 82,115 nouns and 743,241 hypernymy Is-A relations (directed edges \( E \)). These words are embedded in \( \mathbb{D}^n \) such that the distance between words connected by an edge is minimized, while being maximized otherwise. We minimize the same loss function as [NK17a] which is similar with log-likelihood, but approximating the partition function using sampling of negative word pairs (non-edges), fixed to 10 in our case. Note that this loss does not use the direction of the edges in the graph

\[
\mathcal{L}(\theta) = \sum_{(u,v) \in E} \frac{e^{-d_{\mathbb{D}}(u,v)}}{\sum_{u' \in \mathcal{N}(v)} e^{-d_{\mathbb{D}}(u',v)}}
\]

(4.61)

**Metrics.** We report both the loss value and the mean average precision (MAP) [NK17a]: for each directed edge \( (u, v) \), we rank its distance \( d(u, v) \) among the full set of ground truth negative examples \( \{d(u', v) | (u', v) \notin E\} \). We use the same two settings as [NK17a], namely: **reconstruction** (measuring representation capacity) and **link prediction** (measuring generalization). For link prediction we sample a validation set of 2% edges from the set of transitive closure edges that contain no leaf node or root. We only focused on 5-dimensional hyperbolic spaces.

**Training Details.** For all methods we use the same “burn-in phase” described in [NK17a] for 20 epochs, with a fixed learning rate of 0.03 and using RSGD with retraction as explained in Sec. 4.1.2. Solely during this phase, we sampled negative words based on their graph degree raised at power 0.75. This strategy improves all metrics. After that, when different optimization methods start, we sample negatives uniformly. We use \( n = 5 \), following [NK17a].

---

8In a pair \((u, v)\), \( u \) denotes the parent, i.e. \( u \) entails \( v \) — parameters \( \theta \) are coordinates of all \( u, v \).
Adaptive optimization in non-Euclidean geometry

Optimization methods. Experimentally we obtained slightly better results for RADAM over RAMSGRAD, so we will mostly report the former. Moreover, we unexpectedly observed convergence to lower loss values when replacing the true exponential map with its first order approximation – i.e. the retraction $R_x(v) = x + v$ – in both RSGD and in our adaptive methods from Alg. 4.1a. One possible explanation is that retraction methods need fewer steps and smaller gradients to “escape” points sub-optimally collapsed on the ball border of $\mathbb{D}^n$ compared to fully Riemannian methods. As a consequence, we report “retraction”-based methods in a separate setting as they are not directly comparable to their fully Riemannian analogues.

Results. We show in Figures 4.2 and 4.3 results for “exponential” based and “retraction” based methods. We ran all our methods with different learning rates from the set \{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}. For the RSGD baseline we show in orange the best learning rate setting, but we also show the previous lower (slower convergence, in blue) and the next higher (faster overfitting, in green) learning rates. For RADAM and RAMSGRAD we only show the best settings. We always use $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for these methods as these achieved the lowest training loss. RADAGRAD was consistently worse, so we do not report it. As can be seen, RADAM always achieves the lowest training loss. On the MAP metric for both reconstruction and link prediction settings, the same method also outperforms all the other methods for the full Riemannian setting (i.e. Tab. 4.2). Interestingly, in the “retraction” setting, RADAM reaches the lowest training loss value and is on par with RSGD on the MAP evaluation for both reconstruction and link prediction settings. However, RAMSGRAD is faster to converge in terms of MAP for the link prediction task, suggesting that this method has a better generalization capability.

4.5 Additional related work

After RSGD was introduced by [Bon13], a plethora of other first order Riemannian methods arose, such as Riemannian svrg [ZRS16], Riemannian Stein variational gradient descent [LZ17], Riemannian accelerated gradient descent [Liu+17]; [ZS18b] or averaged RSGD [Tri+18], along with new methods for their convergence analysis in the geodesically convex case [ZS16]. Stochastic gradient Langevin dynamics was gen-
Figure 4.2: Results for methods doing updates with the exponential map. From left to right we report: training loss, MAP on the train set, MAP on the validation set.
Figure 4.3: Results for methods doing updates with the retraction. From left to right we report: training loss, MAP on the train set, MAP on the validation set.
4.6 Conclusive Summary

Driven by recent work in learning non-Euclidean embeddings for symbolic data, we propose to generalize popular adaptive optimization tools (e.g. Adam, Amsgrad, Adagrad) to Cartesian products of Riemannian manifolds in a principled and intrinsic manner. We derive convergence rates that are similar to the Euclidean corresponding models. Experimentally we show that our methods outperform popular non-adaptive methods such as RSGD on the realistic task of hyperbolic word taxonomy embedding.

4.7 Reminders of Knowns Results

**Theorem 4.3 (Convergence of Amsgrad).** Let \((f_t)\) be a family of differentiable, convex functions from \(\mathbb{R}^n\) to \(\mathbb{R}\). Let \((x_t)\) and \((v_t)\) be the sequences obtained from Algorithm 4.1b, \(\alpha_t = \alpha / \sqrt{t}\), \(\beta_1 = \beta_{11}\), \(\beta_{1t} \leq \beta_1\) for all \(t \in [T]\) and \(\gamma = \beta_1 / \sqrt{\beta_2} < 1\). Assume that each \(X_i \subset \mathbb{R}\) has a diameter bounded by \(D_\infty\) and that for all \(1 \leq i \leq n\), \(t \in [T]\) and \(x \in X_i\), \(\|\nabla f_t(x)\|_\infty \leq G_\infty\).
For \((x_t)\) generated using the AMSGRAD (Algorithm 4.1b), we have the following bound on the regret

\[
R_T \leq \frac{\sqrt{T}D_\infty^2}{2\alpha (1 - \beta_1)} \sum_{i=1}^{n} \sqrt{\delta_T} + \frac{D_\infty^2}{2(1 - \beta_1)} \sum_{i=1}^{n} \sum_{t=1}^{T} \sqrt{\delta_T} + \frac{\alpha \sqrt{1 + \log T}}{(1 - \beta_1)^2(1 - \gamma) \sqrt{1 - \beta_2}} \sum_{i=1}^{n} \sqrt{T \sum_{t=1}^{T} (g_t^i)^2} \quad (4.62)
\]

Proof. See Theorem 4 of [RKK18].

Useful lemmas. The following lemma is a user-friendly inequality developed by [ZS16] in order to prove convergence of gradient-based optimization algorithms, for geodesically convex functions, in Alexandrov spaces.

**Lemma 4.3** (Cosine inequality in Alexandrov spaces). If \(a, b, c,\) are the sides (i.e., side lengths) of a geodesic triangle in an Alexandrov space with curvature lower bounded by \(\kappa,\) and \(A\) is the angle between sides \(b\) and \(c,\) then

\[
a^2 \leq \frac{\sqrt{|\kappa|}c}{\tanh(\sqrt{|\kappa|}c)} b^2 + c^2 - 2bc \cos(A). \quad (4.63)
\]

Proof. See section 3.1, lemma 6 of [ZS16].

**Lemma 4.4** (An analogue of Cauchy-Schwarz). For all \(p, k \in \mathbb{N}^+, u_1, ..., u_k \in \mathbb{R}^p, a_1, ..., a_k \in \mathbb{R}_+,\) we have

\[
\| \sum_{i} a_i u_i \|_2^2 \leq \left( \sum_{i} a_i \right) \left( \sum_{i} a_i \| u_i \|_2^2 \right). \quad (4.64)
\]
Proof. The proof consists in applying Cauchy-Schwarz’ inequality two times:

\[
\| \sum_i a_i u_i \|_2^2 = \sum_{i,j} a_i a_j u_i^T u_j \quad (4.65)
\]

\[
= \sum_{i,j} \sqrt{a_i a_j} (\sqrt{a_i} u_i)^T (\sqrt{a_j} u_j) \quad (4.66)
\]

\[
\leq \sum_{i,j} \sqrt{a_i a_j} \| \sqrt{a_i} u_i \|_2 \| \sqrt{a_j} u_j \|_2 \quad (4.67)
\]

\[
= \left( \sum_i \sqrt{a_i} \| \sqrt{a_i} u_i \|_2 \right)^2 \quad (4.68)
\]

\[
\leq \left( \sum_i a_i \right) \left( \sum_i \sqrt{a_i} \| u_i \|_2^2 \right) \quad (4.69)
\]

Finally, this last lemma is used by [RKK18] in their convergence proof for ADAMNC. We need it too, in an analogue lemma.

**Lemma 4.5 ([ACG02]).** *For any non-negative real numbers \(y_1, \ldots, y_t\), the following holds:*

\[
\sum_{i=1}^t \frac{y_i}{\sqrt{\sum_{j=1}^i y_j}} \leq 2 \sqrt{\sum_{i=1}^t y_i}. \quad (4.70)
\]
In this chapter, we present two natural extensions of GCN leveraging non-Euclidean geometries: $\kappa$-GCN and OTGNN. More introductory context is provided in section 1.3.5. We start by introducing background and mathematical preliminaries, and then follow on by motivating and introducing our models.

5.1 BACKGROUND & MATHEMATICAL PRELIMINARIES

5.1.1 The Geometry of Spaces of Constant Curvature

Riemannian geometry. A manifold $M$ of dimension $d$ is a generalization to higher dimensions of the notion of surface, and is a space that locally looks like $\mathbb{R}^d$. At each point $x \in M$, $M$ can be associated a tangent space $T_x M$, which is a vector space of dimension $d$ that can be understood as a first order approximation of $M$ around $x$. A Riemannian metric $g$ is given by an inner-product $g_x(\cdot, \cdot)$ at each tangent space $T_x M$, $g_x$ varying smoothly with $x$. A given $g$ defines the geometry of $M$, because it can be used to define the distance between $x$ and $y$ as the infimum of the lengths of smooth paths $\gamma : [0, 1] \to M$ from $x$ to $y$, where the length is defined as $\ell(\gamma) := \int_0^1 \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))} \, dt$. Under certain assumptions, a given $g$ also defines a curvature at each point.

Figure 5.1: Geodesics in the Poincaré disk (left) and the stereographic projection of the sphere (right).
UNIFYING ALL CURVATURES $\kappa$. There exist several models of respectively constant positive and negative curvatures. For positive curvature, we choose the stereographic projection of the sphere, while for negative curvature we choose the Poincaré model which is the stereographic projection of the Lorentz model. As explained below, this choice allows us to generalize the gyrovector space framework and unify spaces of both positive and negative curvature $\kappa$ into a single model which we call the $\kappa$-stereographic model.

THE $\kappa$-STEREOGRAPHIC MODEL. For a curvature $\kappa \in \mathbb{R}$ and a dimension $d \geq 2$, we study the model $\text{st}_\kappa^d$ defined as $\text{st}_\kappa^d = \{ x \in \mathbb{R}^d \mid -\kappa \| x \|^2 < 1 \}$ equipped with its Riemannian metric $g^\kappa_x = \frac{4}{(1+\kappa \| x \|^2)^2} I = (\lambda_x^\kappa)^2 I$. Note in particular that when $\kappa \geq 0$, $\text{st}_\kappa^d$ is $\mathbb{R}^d$, while when $\kappa < 0$ it is the open ball of radius $1/\sqrt{-\kappa}$.

GYROVECTOR SPACES & RIEMANNIAN GEOMETRY. As discussed earlier, the gyrovector space formalism is used to generalize vector spaces to the Poincaré model of hyperbolic geometry [Ung05]; [Ung08]. In addition, important quantities from Riemannian geometry can be rewritten in terms of the M"obius vector addition and scalar-vector multiplication [GBH18c]. We here extend gyrovector spaces to the $\kappa$-stereographic model, i.e. allowing positive curvature.

For $\kappa > 0$ and any point $x \in \text{st}_\kappa^d$, we will denote by $\tilde{x}$ the unique point of the sphere of radius $\kappa^{-\frac{1}{2}}$ in $\mathbb{R}^{d+1}$ whose stereographic projection is $x$. It is given by:

$$\tilde{x} := (\lambda_x^\kappa x, \kappa^{-\frac{1}{2}} (\lambda_x^\kappa - 1)). \quad (5.1)$$
5.1 Background & mathematical preliminaries

Let us remind why. Fix the south pole $z = (0, -1/\sqrt{\kappa})$ of the sphere of curvature $\kappa > 0$, i.e. of radius $R := \kappa^{-\frac{1}{2}}$. The **stereographic projection** is the map:

$$\Phi : S^n_{R} \rightarrow \mathbb{R}^n, x' \mapsto x = \frac{1}{1 + \sqrt{\kappa}x'_{n+1}}x_{1:n'}$$  \hspace{1cm} (5.2)

with the inverse given by

$$\Phi^{-1} : \mathbb{R}^n \rightarrow S^n_{R}, x \mapsto x' = \left(\frac{\lambda_x^\kappa}{\sqrt{\kappa}}x, \frac{1}{\sqrt{\kappa}}(\lambda_x^\kappa - 1)\right),$$  \hspace{1cm} (5.3)

where we define $\lambda_x^\kappa = \frac{2}{1 + |x|^2}$. Again we take the image of the sphere $S^n_{R}$ under the extended projection $\Phi((0, \ldots, 0, -\frac{1}{\kappa})) = 0$, leading to the stereographic model of the sphere. The metric tensor transforms as:

$$g_{ij}^\kappa = (\lambda_x^\kappa)^2 \delta_{ij}.$$  \hspace{1cm} (5.4)

For $x, y \in \text{st}_\kappa^d$, we now define the $\kappa$-**addition**, in the $\kappa$-stereographic model by:

$$x \oplus_\kappa y = \frac{(1 - 2\kappa x^T y - \kappa||y||^2)x + (1 + \kappa||x||^2)y}{1 - 2\kappa x^T y + \kappa^2||x||^2||y||^2} \in \text{st}_\kappa^d.$$  \hspace{1cm} (5.5)

The $\kappa$-addition is defined in all the cases except for spherical geometry and $x = y/(\kappa||y||^2)$ as stated by the following theorem.

**Theorem 5.1** (Definiteness of $\kappa$-addition). *We have $1 - 2\kappa x^T y + \kappa^2||x||^2||y||^2 = 0$ if and only if $\kappa > 0$ and $x = y/(\kappa||y||^2)$.*

**Proof.** Using Cauchy-Schwarz’s inequality, we have $A := 1 - 2\kappa x^T y + \kappa^2||x||^2||y||^2 \geq 1 - 2|\kappa||x||y|| + \kappa^2||x||^2||y||^2 = (1 - |\kappa||x||y||)^2 \geq 0$. Since equality in the Cauchy-Schwarz inequality is only reached for colinear vectors, we have that $A = 0$ is equivalent to $\kappa > 0$ and $x = y/(\kappa||y||^2)$. \(\square\)

For $s \in \mathbb{R}$ and $x \in \text{st}_\kappa^d$ (and $|s\tan^{-1}_\kappa ||x||| < \kappa^\frac{1}{2}\pi/2$ if $\kappa > 0$), the $\kappa$-**scaling** in the $\kappa$-stereographic model is given by:

$$s \otimes_\kappa x = \tan_\kappa (s \cdot \tan^{-1}_\kappa ||x||) \frac{x}{||x||} \in \text{st}_\kappa^d,$$  \hspace{1cm} (5.6)

where $\tan_\kappa$ equals $\kappa^{-1/2}$ tan if $\kappa > 0$ and $(-\kappa)^{-1/2}$ tanh if $\kappa < 0$. This formalism yields simple closed-forms for various quantities including
the distance function (see fig. 5.2) inherited from the Riemannian manifold \( (\mathfrak{st}_k^d, g^κ) \), the exp and log maps, and geodesics (see fig. 5.1), as shown by the following theorem.

**Theorem 5.2** (Extending gyrovector spaces to positive curvature). For \( x, y \in \mathfrak{st}_k^d, x \neq y, v \neq 0, \) (and \( x \neq -y/(κ\|y\|^2) \) if \( κ > 0 \)), the distance function is given by:

\[
d_κ(x, y) = 2|κ|^{-1/2} \tan_1^{-1} \| -x \oplus_κ y \|, \tag{5.7}
\]

the unit-speed geodesic from \( x \) to \( y \) is unique and given by

\[
γ_{x \rightarrow y}(t) = x \oplus_κ \left( t \otimes_κ (-x \oplus_κ y) \right), \tag{5.8}
\]

and finally the exponential and logarithmic maps are described as:

\[
\exp_κ^x(v) = x \oplus_κ \left( \tan_κ \left( |κ|^{1/2} \frac{λ_κ^x \|v\|}{2} \right) \frac{v}{\|v\|} \right), \tag{5.9}
\]

\[
\log_κ^x(y) = \frac{2|κ|^{-1/2}}{λ_κ^x} \tan_1^{-1} \| -x \oplus_κ y \| \frac{-x \oplus_κ y}{\| -x \oplus_κ y \|}. \tag{5.10}
\]

\[^a\text{We write } -x \oplus y \text{ for } (-x) \oplus y \text{ and not } -(x \oplus y).\]

**Proof.** Let us start by proving that for \( x \in \mathbb{R}^n \) and \( v \in T_x\mathbb{R}^n \) the **exponential map** is given by

\[
\exp_κ^x(v) = \frac{λ_κ^x \left( α - \sqrt{κ} x^T \frac{v}{\|v\|} \right) x + \frac{1}{\sqrt{κ}} β \frac{v}{\|v\|} }{1 + (λ_κ^x - 1)α - \sqrt{κ}λ_κ^x x^T \frac{v}{\|v\|} β}, \tag{5.11}
\]

where \( α = \cos_κ (λ_κ^x \|v\|) \) and \( β = \sin_κ (λ_κ^x \|v\|) \).

Indeed, take a unit speed geodesic \( γ_{x,v}(t) \) starting from \( x \) with direction \( v \). Notice that the unit speed geodesic on the sphere starting from \( x' \in S^{n-1} \) is given by \( Γ_{x',v}(t) = x' \cos_κ(t) + \frac{1}{\sqrt{κ}} \sin_κ(t)v' \). By the Egregium theorem, we know that \( Φ(γ_{x,v}(t)) \) is again a unit speed geodesic in the sphere where \( Φ^{-1}: x \mapsto x' = (λ_κ^x x, \frac{1}{\sqrt{κ}} (λ_κ^x - 1)) \). Hence
\( \Phi(\gamma_{x,v}(t)) \) is of the form of \( \Gamma \) for some \( x' \) and \( v' \). We can determine those by

\[
x' = \Phi^{-1}(\gamma(0)) = \Phi^{-1}(x) = \left( \lambda^\kappa_x x, \frac{1}{\sqrt{\kappa}} (\lambda^\kappa_x - 1) \right)
\]

\[
v' = \Gamma(0) = \frac{\partial \Phi^{-1}(y)}{\partial y} \gamma(0) \dot{\gamma}(0)
\]

Notice that

\[
\nabla_x \lambda^\kappa_x = -\kappa (\lambda^\kappa_x)^2 x
\]

and we thus get

\[
v' = \begin{pmatrix}
-2\kappa (\lambda^\kappa_x)^2 x^T v x + \lambda^\kappa_x v \\
- \sqrt{\kappa} (\lambda^\kappa_x)^2 x^T v
\end{pmatrix}
\]

We can obtain \( \gamma_{x,v} \) again by inverting back by calculating \( \gamma_{x,v}(t) = \Phi(\Gamma_{x',v}(t)), \) resulting in

\[
\gamma_{x,v}(t) = \frac{(\lambda^\kappa_x \cos_\kappa(t) - \sqrt{\kappa} (\lambda^\kappa_x)^2 x^T v \sin_\kappa(t)) x}{1 + (\lambda^\kappa_x - 1) \cos_\kappa(t) - \sqrt{\kappa} (\lambda^\kappa_x)^2 x^T v \sin_\kappa(t)}
+ \frac{1}{\sqrt{\kappa}} \lambda^\kappa_x \sin_\kappa(t) v
\]

Denoting \( g^\kappa_x(v,v) = ||v||^2 \lambda^\kappa_x \) we have that

\[
\exp^\kappa_x(v) = \gamma_x, \frac{1}{\sqrt{g^\kappa_x(v,v)}} v \left( \sqrt{g^\kappa_x(v,v)} \right), \quad (5.12)
\]

which concludes the proof of the above formula of the exponential map.

One then notices that it can be re-written in terms of the \( \kappa \)-addition. The formula for the logarithmic map is easily checked by verifying that it is indeed the inverse of the exponential map. Finally, the distance formula is obtained via the well-known identity \( d_\kappa(x,y) = ||\log^\kappa_x(y)||_x \) where \( ||v||_x = \sqrt{g^\kappa_x(v,v)} \).

Note that as expected, \( \exp^\kappa_x(v) \to_{\kappa \to 0} x + v \), converging to the Euclidean exponential map.

\[ \]  

**Around \( \kappa = 0 \).** One notably observes that choosing \( \kappa = 0 \) yields all corresponding Euclidean quantities, which guarantees a *continuous* interpolation between \( \kappa \)-stereographic models of different curvatures, via Euler’s formula \( \tan(x) = -i \tanh(ix) \) where \( i := \sqrt{-1} \). But is this interpolation *differentiable* with respect to \( \kappa \)? It is, as shown by the following theorem.
Theorem 5.3 (Smoothness of $st^\kappa_\kappa$ w.r.t. $\kappa$ around 0). Let $v \neq 0$ and $x, y \in \mathbb{R}^d$, such that $x \neq y$ and $x \neq -y/(\kappa\|y\|^2)$ if $\kappa > 0$. Quantities in Eqs. (5.7, 5.8, 5.9, 5.10) are well-defined for $|\kappa| < 1/\min(\|x\|^2, \|y\|^2)$, i.e. for $\kappa$ small enough. Their first order derivatives at $0^-$ and $0^+$ exist and are equal. Moreover, for the distance we have up to quadratic terms in $\kappa$:

$$d_\kappa(x, y) \approx 2\|x - y\| - 2\kappa\left(\|x - y\|^3/3 + (x^T y)\|x - y\|^2\right) \quad (5.13)$$

Proof. We first compute a Taylor development of the $\kappa$-addition w.r.t $\kappa$ around zero:

$$x \oplus_\kappa y = \frac{(1 - 2\kappa x^T y - \kappa\|y\|^2)x + (1 + \kappa\|x\|^2)y}{1 - 2\kappa x^T y + \kappa^2\|x\|^2\|y\|^2}$$

$$= [(1 - 2\kappa x^T y - \kappa\|y\|^2)x + (1 + \kappa\|x\|^2)y][1 + 2\kappa x^T y + O(\kappa^2)]$$

$$= (1 - 2\kappa x^T y - \kappa\|y\|^2)x + (1 + \kappa\|x\|^2)y$$

$$+ 2\kappa x^T y[x + y] + O(\kappa^2)$$

$$= (1 - \kappa\|y\|^2)x + (1 + \kappa\|x\|^2)y + 2\kappa(x^T y)y$$

$$+ O(\kappa^2)$$

$$= x + y + \kappa[\|x\|^2 y - \|y\|^2 x + 2(x^T y)y] + O(\kappa^2).$$

We then notice that using the Taylor of $\| \cdot \|_2$, given by $\|x + v\|_2 = \|x\|_2 + \langle x, v \rangle + O(\|v\|_2^2)$ for $v \to 0$, we get

$$\|x \oplus_\kappa y\| = \|x + y\| + \kappa(\|x\|^2 y - \|y\|^2 x$$

$$+ 2(x^T y)y, x + y\rangle + O(\kappa^2)$$

$$= \|x + y\| + \kappa(x^T y)\|x + y\|^2 + O(\kappa^2). \quad (5.15)$$

Finally Taylor developments of $\tan_\kappa(|\kappa|^{1/2} u)$ and $|\kappa|^{-1/2} \tan_\kappa^{-1}(u)$ w.r.t $\kappa$ around 0 for fixed $u$ yield For $\kappa \to 0^+$

$$\tan_\kappa(|\kappa|^{1/2} u) = \kappa^{-1/2} \tan(\kappa^{1/2} u)$$

$$= \kappa^{-1/2}(\kappa^{1/2} u + \kappa^{3/2} u^3/3O(\kappa^5))$$

$$= u + \kappa u^3/3 + O(\kappa^2). \quad (5.16)$$
For $\kappa \to 0^-$,
\[
\tan_\kappa(|\kappa|^{\frac{1}{2}}u) = (-\kappa)^{-\frac{1}{2}} \tanh((\frac{\kappa}{2})^2 u) \\
= (-\kappa)^{-\frac{1}{2}} ((-\kappa)^{\frac{3}{2}} u - (-\kappa)^{\frac{3}{2}} u^3/3 + O(\kappa^5)) \\
= u + \kappa u^3/3 + O(\kappa^2).
\] (5.17)

The left and right derivatives match, hence even though $\kappa \mapsto |\kappa|^{\frac{1}{2}}$ is not differentiable at $\kappa = 0$, the function $\kappa \mapsto \tan_\kappa(|\kappa|^{\frac{1}{2}}u)$ is. A similar analysis yields the same conclusion for $\kappa \mapsto |\kappa|^{-\frac{1}{2}} \tan^{-1}_\kappa(u)$ yielding
\[
\text{For } \kappa \to 0, \quad |\kappa|^{-\frac{1}{2}} \tan^{-1}_\kappa(u) = u - \kappa u^3/3 + O(\kappa^2).
\] (5.18)

Since a composition of differentiable functions is differentiable, we consequently obtain that $\otimes_\kappa, \exp^\kappa, \log^\kappa$ and $d_\kappa$ are differentiable functions of $\kappa$, under the assumptions on $x, y, v$ stated in Theorem 3. Finally, the Taylor development of $d_\kappa$ follows by composition of Taylor developments:
\[
d_\kappa(x, y) = 2\|\kappa\|^{-\frac{1}{2}} \tan^{-1}_\kappa(\|(-x) \oplus_\kappa y\|) \\
= 2(\|x - y\| + \kappa((-x)^T y)\|x - y\|^2)(1 - \\
(\kappa/3)(\|x - y\| + O(\kappa))^2) + O(\kappa^2) \\
= 2(\|x - y\| + \kappa((-x)^T y)\|x - y\|^2)(1 \\
- (\kappa/3)\|x - y\|^2) + O(\kappa^2) \\
= 2\|x - y\| - 2\kappa (x^T y)\|x - y\|^2 + \|x - y\|^3/3) \\
+ O(\kappa^2).
\]

Note that for $x^T y \geq 0$, this tells us that an infinitesimal change of curvature from zero to small negative, i.e. towards $0^-$, while keeping $x, y$ fixed, has the effect of increasing their distance.

As a consequence, we have a unified formalism that interpolates smoothly between all three geometries of constant curvature.

5.1.2 Optimal Transport Geometry

OT is a mathematical framework that defines distances or similarities between objects such as probability distributions, either discrete or
Figure 5.3: We illustrate, for a given 2D point cloud, the optimal transport plan obtained from minimizing the Wasserstein costs; $c(\cdot, \cdot)$ denotes the Euclidean distance (top) or squared difference (bottom). $A, B$ are the Euclidean distance matrices obtained from point clouds $X, Y$. A higher dotted-line thickness illustrates a greater mass transport.

continuous, as the cost of an OT plan from one to the other. In this work we will invoke several variants: the Wasserstein, Gromov-Wasserstein and Fused-Gromov-Wasserstein geometries.

**Wasserstein for point clouds.** Let a point cloud $X = \{x_i\}_{i=1}^n$ of size $n$ be a set of $n$ points $x_i \in \mathbb{R}^d$. Given point clouds $X, Y$ of respective sizes $n, m$, a transport plan (or coupling) is a matrix $T$ of size $n \times m$ with entries in $[0, 1]$, satisfying the two following marginal constraints: $T1_m = \frac{1}{n} 1_n$ and $T^T 1_n = \frac{1}{m} 1_m$. Intuitively, the marginal constraints mean that $T$ preserves the mass from $X$ to $Y$. We denote the set of such couplings as $C_{XY}$.

Given a cost function $c$ on $\mathbb{R}^d$, its associated Wasserstein discrepancy is defined as

$$W(X, Y) = \min_{T \in C_{XY}} \sum_{ij} T_{ij} c(x_i, y_j). \quad (5.19)$$

We further describe the shape of optimal transports for point clouds of same sizes in Section 5.4.5.3.

**Gromov-Wasserstein for adjacency tensors.** We consider an adjacency tensor of dimension $d$ over a set $X = \{x_i\}_{i=1}^n$ to be defined as
a squared matrix \( A \) of size \( n \), where each entry \( A_{ij} \in \mathbb{R}^d \) represents an embedding of the (directed) relation from \( x_i \) to \( x_j \). Given two adjacency tensors \( A \) and \( B \) over respective sets \( X, Y \) of sizes \( n, m \), and a cost function \( c \) on \( \mathbb{R}^d \), we define a Gromov-Wasserstein discrepancy as:

\[
\mathcal{GW}(A, B) = \min_{T \in \mathcal{C}_{XY}} \sum_{ij} \sum_{kl} T_{ij} T_{kl} c(A_{ik}, B_{jl}).
\]

**Intuition.** A first observation is that the Wasserstein geometry relates point clouds living in the same embedding space \( \mathbb{R}^d \), while the Gromov-Wasserstein discrepancy operates on abstract sets which may live in “uncomparable” spaces. The Wasserstein distance represents the cost of moving one point cloud to the other in the same embedding space, while the Gromov-Wasserstein one reflects the cost of aligning relation embeddings within each set. This is illustrated in Figure 5.3. If \( c \) is a similarity function instead, then replacing the above minimum by a maximum yields a (Gromov-)Wasserstein similarity. Peyré & Cuturi [PC+19] and Villani [Vil08] respectively provide excellent resources for the computational and mathematical aspects of OT.

**Fused Gromov-Wasserstein.** Finally, as proposed by Titouan et al. [Tit+19a], it is possible to combine these two geometries with the below definition:

\[
\mathcal{FGW}_\alpha(A, B) = \min_{T \in \mathcal{C}_{XY}} \sum_{ij} \alpha \sum_{kl} T_{ij} T_{kl} c(A_{ik}, B_{jl}) + (1 - \alpha) \sum_{ij} T_{ij} c(x_i, y_j),
\]

where \( \alpha \in [0, 1] \) is a parameters allowing to pounder the trade-off between the two geometries.

### 5.1.3 GCN: Graph Convolutional Networks

The problem of node classification on a graph has long been tackled with explicit regularization using the graph Laplacian [Wes+12]. Namely, for a directed graph with adjacency matrix \( A \), by adding the following term to the loss: \( \sum_{i,j} A_{ij} \| f(x_i) - f(x_j) \|^2 = \| f(X) - L f(X) \|_2 \), where \( L = D - A \) is the (unnormalized) graph Laplacian, \( D_{ii} := \sum_k A_{ik} \) defines the (diagonal) degree matrix, \( f \) contains the trainable parameters of the model and \( X = (x_i^T)_{ij} \) the node features of the model. Such a regularization is expected to improve generalization if connected
nodes in the graph tend to share labels; node $i$ with feature vector $x_i$ is represented as $f(x_i)$ in a Euclidean space.

With the aim to obtain more scalable models, [DBV16b]; [KW17a] propose to make this regularization implicit by incorporating it into what they call GCN, which they motivate as a first order approximation of spectral graph convolutions, yielding the following scalable layer architecture:

$$
H^{(t+1)} = \sigma \left( \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(t)} W^{(t)} \right)
$$

(5.22)

where $\tilde{A} = A + I$ has added self-connections, $\tilde{D}_{ii} = \sum_k \tilde{A}_{ik}$ defines its diagonal degree matrix, $\sigma$ is a non-linearity such as sigmoid, tanh or ReLU = $\max(0, \cdot)$, and $W^{(t)}$ and $H^{(t)}$ are the parameter and activation matrices of layer $t$ respectively, with $H^{(0)} = X$ the input feature matrix.

5.1.4 Directed Message Passing Neural Networks

We briefly remind here of the simplified Directed Message Passing Neural Network (DMPNN) [DDS16] architecture which was successfully used for molecular property prediction by Yang et al. [Yan+19].

This model takes as input a directed graph $G = (V, E)$, with node and edge features denoted by $x_v$ and $e_{vw}$ respectively, for $v, w$ in the vertex set $V$ and when $v \rightarrow w$ is an edge in $E$. The parameters of DMPNN are the below weight matrices $\{W_i, W_m, W_o\}$.

It keeps track of messages $m_{vw}^{t}$ and hidden states $h_{vw}^{t}$, for each step $t$, defined as follows. An initial hidden state is set to $h_{vw}^{0} = \text{ReLU}(W_i \text{cat}(x_v, e_{vw}))$ where “cat” denotes concatenation. Then, the message passing operates as

$$
\begin{align*}
m_{vw}^{t+1} &= \sum_{k \in N(v) \setminus \{w\}} h_{kv}^{t}, \\
h_{vw}^{t+1} &= \text{ReLU}(h_{vw}^{0} + W_m m_{vw}^{t+1}),
\end{align*}
$$

(5.23)

where $N(v) = \{k \in V | (k, v) \in E\}$ denotes $v$’s incoming neighbors. After $T$ steps of message passing, node embeddings are obtained by summing edge embeddings:

$$
\begin{align*}
m_{v} &= \sum_{w \in N(v)} h_{vw}^{T}, \\
h_{v} &= \text{ReLU}(W_o \text{cat}(x_v, m_{v})).
\end{align*}
$$

(5.24)

A final graph embedding is then obtained as $h = \sum_{v \in V} h_{v}$, which is usually fed to a multilayer perceptron (MLP) for classification or regression.
5.2 \( \kappa \)-GCN: CONSTANT CURVATURE GCN

As explained earlier, interest has been rising lately towards methods representing data in non-Euclidean spaces, e.g. hyperbolic or spherical, that provide specific inductive biases useful for certain real-world data properties, e.g. scale-free, hierarchical or cyclical. However, the popular GNN are currently limited in modeling data only via Euclidean Geometry and associated vector space operations. Here, we bridge this gap by proposing mathematically grounded generalizations of GCN to (products of) constant curvature spaces. We do this by (i) introducing a unified formalism that can interpolate smoothly between all geometries of constant curvature, (ii) leveraging gyro-barycentric coordinates that generalize the classic Euclidean concept of the center of mass. Our class of models smoothly recover their Euclidean counterparts when the curvature goes to zero from either side. Empirically, we outperform Euclidean GCNs in the tasks of node classification and distortion minimization for symbolic data exhibiting non-Euclidean behavior, according to their discrete curvature. The following content has in part been published at ICML 2020 [BBG20].

We start by introducing the methods upon which we build. We present our models for spaces of constant sectional curvature, in the \( \kappa \)-stereographic model. However, the generalization to cartesian products of such spaces [Gu+19b] follows naturally from these tools.

5.2.1 Tools for a \( \kappa \)-GCN

Learning a parametrized function \( f_\theta \) that respects hyperbolic geometry has been studied in [GBH18c]: neural layers and hyperbolic softmax. We generalize their definitions into the \( \kappa \)-stereographic model, unifying operations in positive and negative curvature. We explain how curvature introduces a fundamental difference between left and right matrix multiplications, depicting the M"obius matrix multiplication of [GBH18c] as a right multiplication, independent for each embedding. We then introduce a left multiplication by extension of gyromidpoints which ties the embeddings, which is essential for GNN.
NON-EUCLIDEAN GEOMETRY FOR GRAPH-LIKE DATA

5.2.2 $\kappa$-Right-Matrix-Multiplication

Let $X \in \mathbb{R}^{n \times d}$ denote a matrix whose $n$ rows are $d$-dimensional embeddings in $st_{\kappa}^{d}$, and let $W \in \mathbb{R}^{d \times e}$ denote a weight matrix. Let us first understand what a right matrix multiplication is in Euclidean space: the Euclidean right multiplication can be written row-wise as $(XW)_{i} = X_{i}W$. Hence each $d$-dimensional Euclidean embedding is modified independently by a right matrix multiplication. A natural adaptation of this operation to the $\kappa$-stereographic model yields the following definition.

**Definition 5.4.** Given a matrix $X \in \mathbb{R}^{n \times d}$ holding $\kappa$-stereographic embeddings in its rows and weights $W \in \mathbb{R}^{d \times e}$, the $\kappa$-right-matrix-multiplication is defined row-wise as

$$(X \otimes_{\kappa} W)_{i} = \exp_{0}^{\kappa} \left( (\log_{0}^{\kappa}(X)W)_{i} \right)$$

$$= \tan_{\kappa} \left( \alpha_{i} \tan_{\kappa}^{-1} \left( \frac{||X_{i}||}{||W_{i}||} \right) \right) \frac{(XW)_{i}}{||(XW)_{i}||} \quad (5.25)$$

where $\alpha_{i} = \frac{||(XW)_{i}||}{||X_{i}||}$ and $\exp_{0}^{\kappa}$ and $\log_{0}^{\kappa}$ denote the exponential and logarithmic map in the $\kappa$-stereographic model.

This definition is in perfect agreement with the hyperbolic scalar multiplication for $\kappa < 0$, which can also be written as $r \otimes_{\kappa} x =$
exp_0^\kappa(r \log_0^\kappa(x)). This operation is known to have desirable properties such as associativity [GBH18c].

5.2 κ-GCN: Constant Curvature GCN

5.2.3 κ-Left-Matrix-Multiplication: Midpoint Extension

For GNN we also need the notion of message passing among neighboring nodes, i.e. an operation that combines/aggregates the respective embeddings together. In Euclidean space such an operation is given by the left multiplication of the embeddings matrix with the (preprocessed) adjacency \( \hat{A} \): \( H^{(l+1)} = \sigma(\hat{A}Z^{(l)}) \) where \( Z^{(l)} = H^{(l)}W^{(l)} \). Let us consider this left multiplication. For \( A \in \mathbb{R}^{n \times n} \), the matrix product is given row-wise by:

\[
(AX)_{i \bullet} = A_{i1}X_{1 \bullet} + \cdots + A_{in}X_{n \bullet}
\]

This means that the new representation of node \( i \) is obtained by calculating the linear combination of all the other node embeddings, weighted by the \( i \)-th row of \( A \). An adaptation to the \( \kappa \)-stereographic model hence requires a notion of weighted linear combination.

We propose such an operation in \( \text{st}_\kappa^d \) by performing a \( \kappa \)-scaling of a gyromidpoint — whose definition is reminded below. Indeed, in Euclidean space, the weighted linear combination \( \alpha x + \beta y \) can be re-written as \( (\alpha + \beta)m_E(x, y; \alpha, \beta) := \frac{\alpha}{\alpha + \beta}x + \frac{\beta}{\alpha + \beta}y \). See fig. 5.5 for a geometric illustration. This motivates generalizing the above operation to \( \text{st}_\kappa^d \) as follows.

**Definition 5.5.** Given a matrix \( X \in \mathbb{R}^{n \times d} \) holding \( \kappa \)-stereographic embeddings in its rows and weights \( A \in \mathbb{R}^{n \times n} \), the **κ-left-matrix-multiplication** is defined row-wise as

\[
(A \boxtimes_\kappa X)_{i \bullet} := \left( \sum_j A_{ij} \right) \otimes_\kappa m_\kappa(X_{1 \bullet}, \cdots, X_{n \bullet}; A_{i \bullet}).
\]  

The \( \kappa \)-scaling is motivated by the fact that \( d_\kappa(0, r \otimes_\kappa x) = |r|d_\kappa(0, x) \) for all \( r \in \mathbb{R} \), \( x \in \text{st}_\kappa^d \). We remind that the gyromidpoint is defined when \( \kappa \leq 0 \) in the \( \kappa \)-stereographic model as [Ung10]:

\[
m_\kappa(x_1, \cdots, x_n; \alpha) = \frac{1}{2} \otimes_\kappa \left( \sum_{i=1}^n \frac{\alpha_i \lambda_{x_i}^\kappa}{\sum_{j=1}^n \alpha_j (\lambda_{x_j}^\kappa - 1)} x_i \right),
\]  

127
**Figure 5.5:** Weighted Euclidean midpoint $ax + by$

**Figure 5.6:** Heatmap of the distance from a $st_k^2$-hyperplane to $x \in st_k^2$ for $\kappa = -0.254$ (left) and $\kappa = 0.248$ (right)

with $\lambda_x^\kappa = 2/(1 + \kappa \|x\|^2)$. Whenever $\kappa > 0$, we have to further require the following condition:

$$\sum_j \alpha_j(\lambda_x^\kappa - 1) \neq 0. \quad (5.28)$$

For two points, one can calculate that $(\lambda_x^\kappa - 1) + (\lambda_y^\kappa - 1) = 0$ is equivalent to $\kappa \|x\| \|y\| = 1$, which holds in particular whenever $x = -y/(\kappa \|y\|^2)$. See fig. 5.4 for illustrations of gyromidpoints.

Our operation $\boxtimes_\kappa$ satisfies interesting properties:

**Theorem 5.6** (Neuter element & $\kappa$-scalar-associativity). We have $I_n \boxtimes_\kappa X = X$, and for $r \in \mathbb{R}$,

$$r \otimes_\kappa (A \boxtimes_\kappa X) = (rA) \boxtimes_\kappa X.$$
\textbf{Proof.} If } A = I_n \text{ then for all } i \text{ we have } \sum_j A_{ij} = 1, \text{ hence }

\[
(I_n \boxtimes \mathbf{X})_{i*} = \frac{1}{2} \otimes_\kappa \left( \sum_j \delta_{ij} \lambda_{x_i}^\kappa \right. 
\]

\[
= \frac{1}{2} \otimes_\kappa \left( \frac{\lambda_{x_i}^\kappa}{(\lambda_{x_i}^\kappa - 1)} x_i \right)
\]

\[
= \frac{1}{2} \otimes_\kappa (2 \otimes_\kappa x_i)
\]

\[
= x_i
\]

\[
= (\mathbf{X})_{i*}. \tag{5.33}
\]

For associativity, we first note that the gyromidpoint is unchanged by a scalar rescaling of } A. \text{ The property then follows by scalar associativity of the } \kappa\text{-scaling.} \quad \square

\textbf{THE MATRIX } a. \text{ In most GNN, the matrix } A \text{ is intended to be a pre-}

\text{processed adjacency matrix, } i.e. \text{ renormalized by the diagonal degree}

\text{matrix } D_{ii} = \sum_k A_{ik}. \text{ This normalization is often taken either } (i) \text{ to}

\text{the left: } D^{-1}A, (ii) \text{ symmetric: } D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \text{ or } (iii) \text{ to the right: } AD^{-1}. \text{ Note that the latter case makes the matrix right-stochastic\footnote{M is right-stochastic if for all } i, \sum_j M_{ij} = 1, \text{ which is a property that is preserved by matrix product and exponentiation. For this case, we provide the following result:}

\textbf{Theorem 5.7 (}\kappa\text{-left-multiplication by right-stochastic matrices is intrinsic).} \text{ If } A, B \text{ are right-stochastic, } \phi \text{ is a isometry of } \text{st}^d_\kappa \text{ and } X, Y \text{ are two}

\text{matrices holding } \kappa\text{-stereographic embeddings:}

\[
\forall i, \quad d_\phi = d_\kappa ((A \boxtimes_\kappa \phi(X))_{i*}, (B \boxtimes_\kappa \phi(Y))_{i*})
\]

\[
= d_\kappa ((A \boxtimes_\kappa X)_{i*}, (B \boxtimes_\kappa Y)_{i*}). \tag{5.34}
\]

\textbf{Proof.} \text{It is proved in } [\text{Ung05}] \text{ that the gyromidpoint commutes with}

\text{isometries. The exact same proof holds for positive curvature, with}

\text{the same algebraic manipulations. Moreover, when the matrix } A \text{ is}

\text{right-stochastic, for each row, the sum over columns gives } 1, \text{ hence}

\text{our operation } \boxtimes_\kappa \text{ reduces to a gyromidpoint. As a consequence, our}

\text{ } \boxtimes_\kappa \text{ commutes with isometries in this case. Since isometries preserve}

\text{distance, we have proved the theorem.} \quad \square
The above result means that $A$ can easily be preprocessed as to make its $\kappa$-left-multiplication intrinsic to the metric space $(\text{st}_d^d, d_\kappa)$. At this point, one could wonder: does there exist other ways to take weighted centroids on a Riemannian manifold? We comment on two plausible alternatives.

**Fréchet/Karcher means.** They are obtained as:

$$\arg \min_x \sum_i \alpha_i d_\kappa(x, x_i)^2. \tag{5.35}$$

Note that although they are also intrinsic, they usually require solving an optimization problem which can be prohibitively expensive, especially when one requires gradients to flow through the solution — moreover, for the space $\text{st}_d^d$, it is known that the minimizer is unique if and only if $\kappa \geq 0$.

**Tangential aggregations.** The linear combination is here lifted to the tangent space by means of the exponential and logarithmic map and were in particular used in the recent works of [Cha+19] and [LNK19].

**Definition 5.8.** The tangential aggregation of $x_1, \ldots, x_n \in \text{st}_d^d$ w.r.t. weights $\{\alpha_i\}_{1 \leq i \leq n}$, at point $x \in \text{st}_d^d$ (for $x_i \neq -x/(\kappa \|x\|^2)$ if $\kappa > 0$) is defined by:

$$\text{tg}_x^\kappa(x_1, \ldots, x_n; \alpha_1, \ldots, \alpha_n) := \exp_x^\kappa \left( \sum_{i=1}^n \alpha_i \log_x^\kappa(x_i) \right). \tag{5.36}$$

The below theorem describes that for the $\kappa$-stereographic model, this operation is also intrinsic.

**Theorem 5.9 (Tangential aggregation is intrinsic).** For any isometry $\phi$ of $\text{st}_d^d$, we have

$$\text{tg}_{\phi(x)}(\{\phi(x_i)\}; \{\alpha_i\}) = \phi(\text{tg}_x(\{x_i\}; \{\alpha_i\})). \tag{5.37}$$

**Proof.** We begin our proof by stating the left-cancellation law:

$$x \oplus_\kappa (-x \oplus_\kappa y) = y \tag{5.38}$$
and the following simple identity stating that orthogonal maps commute with $\kappa$-addition

$$Rx \oplus_k Ry = R(x \oplus_k y), \quad \forall R \in O(d) \quad (5.39)$$

Next, we generalize the gyro operator from Möbius gyrovector spaces as defined in [Ungo8]:

$$\text{gyr}[u, v]w := -(u \oplus_k v) \oplus_k (u \oplus_k (v \oplus_k w)) \quad (5.40)$$

Note that this definition applies only for $u, v, w \in \text{st}_k$ for which the $\kappa$-addition is defined (see theorem 1). Following [Ungo8], we have an alternative formulation (verifiable via computer algebra):

$$\text{gyr}[u, v]w = w + 2\frac{Au + Bv}{D}. \quad (5.41)$$

where the quantities $A, B, D$ have the following closed-form expressions:

$$A = -\kappa^2 \langle u, w \rangle \|v\|^2 - \kappa \langle v, w \rangle + 2\kappa^2 \langle u, v \rangle \cdot \langle v, w \rangle, \quad (5.42)$$

$$B = -\kappa^2 \langle v, w \rangle \|u\|^2 + \kappa \langle u, w \rangle, \quad (5.43)$$

$$D = 1 - 2\kappa \langle u, v \rangle + \kappa^2 \|u\|^2 \|v\|^2. \quad (5.44)$$

We then have the following relations:

**Lemma 5.1.** For all $u, v, w \in \text{st}_k$ for which the $\kappa$-addition is defined we have the following relations: i) gyration is a linear map, ii) $u \oplus_k v = \text{gyr}[u, v](v \oplus_k u)$, iii) $-(z \oplus_k u) \oplus_k (z \oplus_k v) = \text{gyr}[z, u](-u \oplus_k v)$, iv) $\|\text{gyr}[u, v]w\| = \|w\|$. 

**Proof.** The proof is similar with the one for negative curvature given in [Ungo8]. The fact that gyration is a linear map can be easily verified from its definition. For the second part, we have

$$-\text{gyr}[u, v](v \oplus_k u) = \text{gyr}[u, v](-v \oplus_k u))$$

$$= -(u \oplus_k v) \oplus_k (u \oplus_k (v \oplus_k (-v \oplus_k u)))) \quad (5.45)$$

\[131]
where the first equality is a trivial consequence of the fact that gyration is a linear map, while the last equality is the consequence of left-cancellation law.

The third part follows easily from the definition of the gyration and the left-cancellation law. The fourth part can be checked using the alternate form in equation (5.41).

We now follow [Ung14] and describe all isometries of $st^d_\kappa$ spaces:

**Theorem 5.10.** Any isometry $\phi$ of $st^d_\kappa$ can be uniquely written as:

$$
\phi(x) = z \oplus_\kappa Rx, \quad \text{where } z \in st^d_\kappa, R \in O(d)
$$

The proof is exactly the same as in theorems 3.19 and 3.20 of [Ung14], so we will skip it.

We can now prove the main theorem. Let $\phi(x) = z \oplus_\kappa Rx$ be any isometry of $st^d_\kappa$, where $R \in O(d)$ is an orthogonal matrix. Let us denote by $v := \sum_{i=1}^n \alpha_i \log^\kappa(x_i)$. Then, using lemma 5.1 and the formula of the log map from theorem 2, one obtains the following identity:

$$
\sum_{i=1}^n \alpha_i \log^\kappa(\phi(x_i)) = \frac{\lambda_x^\kappa}{\lambda_{\phi(x)}^\kappa} \text{gyr}[z, Rx]Rv
$$

and, thus, using the formula of the exp map from theorem 2 we obtain:

$$
\text{tg}_{\phi(x)}(\{\phi(x_i)\}; \{\alpha_i\}) = \phi(x) \oplus_\kappa \text{gyr}[z, Rx]R(-x \oplus_\kappa \exp^\kappa_x(v))
$$

Using eq. (5.40), we get that $\forall w \in st^d_\kappa$:

$$
\text{gyr}[z, Rx]Rw = -\phi(x) \oplus_\kappa (z \oplus_\kappa (Rx \oplus_\kappa Rw))
$$

giving the desired

$$
\text{tg}_{\phi(x)}(\{\phi(x_i)\}; \{\alpha_i\}) = z \oplus_\kappa R \exp^\kappa_x(v)
= \phi(\text{tg}_x(\{x_i\}; \{\alpha_i\})).
$$

\Box

132
5.2.4 Logits

Finally, we need the logit and softmax layer, a necessity for any classification task. We here use the model of [GBH18c], which was obtained in a principled manner for the case of negative curvature. Their derivation rests upon the closed-form formula for distance to a hyperbolic hyperplane. We naturally extend this formula to $\text{st}_\kappa^d$, hence also allowing for $\kappa > 0$ but leave for future work the adaptation of their theoretical analysis.

$$p(y = k|x) = S \left( \frac{||a_k||_{p_k} \sin^{-1} \left( \frac{2\sqrt{||z_k||^2||a_k||}}{(1 + \kappa||z_k||^2)||a_k||} \right)}{\sqrt{\kappa}} \right), \quad (5.51)$$
where $a_k \in T_{0}\text{st}_\kappa^d \cong \mathbb{R}^d$, $x \in \text{st}_\kappa^d$, $p_k \in \text{st}_\kappa^d$ and $S(\cdot)$ is the softmax function.

We reference the reader to fig. 5.6 for an illustration of eq. 5.51.

5.2.5 Final Architecture of the $\kappa$-GCN

We are now ready to introduce our $\kappa$-stereographic GCN [KW17a], denoted by $\kappa$-GCN$^2$. Assume we are given a graph with node level features $G = (V, A, X)$ where $X \in \mathbb{R}^{n \times d}$ with each row $X_i \in \text{st}_\kappa^d$ and adjacency $A \in \mathbb{R}^{n \times n}$. We first perform a preprocessing step by mapping the Euclidean features to $\text{st}_\kappa^d$ via the projection $X \mapsto X / (2\sqrt{||X||_\text{max}})$, where $||X||_\text{max}$ denotes the maximal Euclidean norm among all stereographic embeddings in $X$. For $l \in \{0, \ldots, L - 2\}$, the $(l + 1)$-th layer of $\kappa$-GCN is given by:

$$H^{(l+1)} = \sigma^{\otimes x} \left( \hat{\mathbf{A}} \otimes_\kappa \left( H^{(l)} \otimes_\kappa W^{(l)} \right) \right), \quad (5.52)$$
where $H^{(0)} = X$, $\sigma^{\otimes x}(x) := \exp^x_0(\sigma(\log^x_0(x)))$ is the Möbius version [GBH18c] of a pointwise non-linearity $\sigma$ and $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{A} \tilde{\mathbf{D}}^{-\frac{1}{2}}$. The final layer is a $\kappa$-logit layer.

$$H^{(L)} = \text{softmax} \left( \hat{\mathbf{A}} \logit_\kappa \left( H^{(L-1)} , W^{(L-1)} \right) \right), \quad (5.53)$$
where $W^{(L-1)}$ contains the parameters $a_k$ and $p_k$ of the $\kappa$-logits layer. A very important property of $\kappa$-GCN is that its architecture recovers the Euclidean GCN when we let curvature go to zero:

---

$^2$To be pronounced “kappa” GCN; the greek letter $\kappa$ being commonly used to denote sectional curvature.
non-Euclidean geometry for graph-like data

\[ \kappa\text{-GCN} \xrightarrow{\kappa \to 0} \text{GCN.} \]

5.3 EXPERIMENTS: GRAPH DISTORSION & NODE CLASSIFICATION

5.3.1 Distorsion of Graph Embeddings

We evaluate the architectures introduced in the previous sections on the tasks of node classification and minimizing embedding distortion for several synthetic as well as real datasets.

Experimental details. We split the data into training, early stopping, validation and test set. Namely we first split the dataset into a known subset of size \( n_{\text{known}} \) and an unknown subset consisting of the rest of the nodes. For all the graphs we use \( n_{\text{known}} = 1500 \) except for the airport dataset, where we follow the setup of [Cha+19] and use \( n_{\text{known}} = 2700 \). For the citation graphs, the known subset is further split into a training set consisting of 20 data points per label, an early stopping set of size 500 and a validation set of the remaining nodes. For airport, we separate the known subset into 2100 training nodes, 300 validation nodes and 300 early stopping nodes. Notice that the whole structure of the graph and all the node features are used in an unsupervised fashion since the embedding of a training node might for instance depend on the embedding of a node from the validation set. But when calculating the loss, we only provide supervision with the training data. The unknown subset serves as the test data and is only used for the final evaluation of the model. Hyperparameter-tuning is performed on the validation set. We further use early stopping in all the experiments. We stop training as soon as the early stopping cross entropy loss has not decreased in the last \( n_{\text{patience}} = 200 \) epochs or as soon as we have reached \( n_{\text{max}} = 2000 \) epochs. The model chosen is the one with the highest accuracy score on the early stopping set. For the final evaluation we test the model on five different data splits and two runs each and report mean accuracy and bootstrapped confidence intervals. We use the described setup for both the Euclidean and non-Euclidean models to ensure a fair comparison.
5.3 Experiments: Graph Distorsion & Node Classification

Table 5.1: Minimum achieved average distortion of the different models. H and S denote hyperbolic and spherical models respectively.

<table>
<thead>
<tr>
<th>Model</th>
<th>Tree</th>
<th>Toroidal</th>
<th>Spherical</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E^{10}$ (Linear)</td>
<td>0.045</td>
<td>0.0607</td>
<td>0.0415</td>
</tr>
<tr>
<td>$E^{10}$ (ReLU)</td>
<td>0.0502</td>
<td>0.0603</td>
<td>0.0409</td>
</tr>
<tr>
<td>$H^{10}$ (κ-GCN)</td>
<td>0.0029</td>
<td>0.272</td>
<td>0.267</td>
</tr>
<tr>
<td>$S^{10}$ (κ-GCN)</td>
<td>0.473</td>
<td>0.0485</td>
<td>0.0337</td>
</tr>
<tr>
<td>$H^5 \times H^5$ (κ-GCN)</td>
<td>0.0048</td>
<td>0.112</td>
<td>0.152</td>
</tr>
<tr>
<td>$S^5 \times S^5$ (κ-GCN)</td>
<td>0.51</td>
<td><strong>0.0464</strong></td>
<td>0.0359</td>
</tr>
<tr>
<td>$(H^2)^4$ (κ-GCN)</td>
<td>0.025</td>
<td>0.084</td>
<td>0.062</td>
</tr>
<tr>
<td>$(S^2)^4$ (κ-GCN)</td>
<td>0.312</td>
<td>0.0481</td>
<td>0.0378</td>
</tr>
</tbody>
</table>

Minimizing Distortion  Our first goal is to evaluate the graph embeddings learned by our GCN models on the representation task of fitting the graph metric in the embedding space. We desire to minimize the average distortion, i.e. defined similarly as in [Gu+19b]:

\[
\frac{1}{n^2} \sum_{i,j} \left( \left( \frac{d(x_i, x_j)}{d_G(i, j)} \right)^2 - 1 \right)^2 ,
\]

where $d(x_i, x_j)$ is the distance between the embeddings of nodes $i$ and $j$, while $d_G(i, j)$ is their graph distance (shortest path length).

We create three synthetic datasets that best reflect the different geometries of interest: i) “Tree”: a balanced tree of depth 5 and branching factor 4 consisting of 1365 nodes and 1364 edges. ii) “Torus”: We sample points (nodes) from the (planar) torus, i.e. from the unit connected square; two nodes are connected by an edge if their toroidal distance (the warped distance) is smaller than a fixed $R = 0.01$; this gives 1000 nodes and 30626 edges. iii) “Spherical Graph”: we sample points (nodes) from $S^2$, connecting nodes if their distance is smaller than 0.2, leading to 1000 nodes and 17640 edges.

For the GCN models, we use 1-hot initial node features. We use two GCN layers with dimensions 16 and 10. The non-Euclidean models
Table 5.2: Node classification: Average accuracy across 5 splits with estimated uncertainties at 95 percent confidence level via bootstrapping on our dataspits. $\mathbb{H}$ and $\mathbb{S}$ denote hyperbolic and spherical models respectively. Implementation of these experiments was done in collaboration with co-authors in [BBG20].

<table>
<thead>
<tr>
<th>Model</th>
<th>Citeseer</th>
<th>Cora</th>
<th>Pubmed</th>
<th>Airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{E}^{16}$ [KW17a]</td>
<td>0.729 ± 0.0054</td>
<td>0.814 ± 0.004</td>
<td>0.792 ± 0.0039</td>
<td>0.814 ± 0.0029</td>
</tr>
<tr>
<td>$\mathbb{H}^{16}$ [Cha+19]</td>
<td>0.71 ± 0.0049</td>
<td>0.803 ± 0.0046</td>
<td>0.798 ± 0.0043</td>
<td>0.844 ± 0.0041</td>
</tr>
<tr>
<td>$\mathbb{H}^{16}$ (κ-GCN)</td>
<td><strong>0.732 ± 0.0051</strong></td>
<td>0.812 ± 0.005</td>
<td>0.785 ± 0.0036</td>
<td>0.819 ± 0.0033</td>
</tr>
<tr>
<td>$\mathbb{S}^{16}$ (κ-GCN)</td>
<td>0.721 ± 0.0045</td>
<td><strong>0.819 ± 0.0045</strong></td>
<td>0.788 ± 0.0049</td>
<td>0.809 ± 0.0058</td>
</tr>
<tr>
<td>Prod-GCN (κ-GCN)</td>
<td>0.711 ± 0.0059</td>
<td>0.808 ± 0.0041</td>
<td>0.781 ± 0.006</td>
<td>0.817 ± 0.0044</td>
</tr>
</tbody>
</table>

do not use additional non-linearities between layers. All Euclidean parameters are updated using the ADAM optimizer with learning rate 0.01. Curvatures are learned using (stochastic) gradient descent and learning rate of 0.0001. All models are trained for 10000 epochs and we report the minimal achieved distortion.

distortion results. The obtained distortion scores shown in table 5.1 reveal the benefit of our models. The best performing architecture is the one that matches the underlying geometry of the graph.

5.3.2 Node Classification

We consider the popular node classification datasets Citeseer [Sen+08], Cora-ML [McC+00] and Pubmed [Nam+12]. Node labels correspond to the particular subfield the published document is associated with. We compare against the Euclidean model [KW17a] and the recently proposed hyperbolic variant [Cha+19].

curvature estimations of datasets To understand how far are the real graphs of the above datasets from the Euclidean geometry, we first estimate the graph curvature of the four studied datasets
5.3 Experiments: Graph Distorsion & Node Classification

using the deviation from the Parallelogram Law [Gu+19b]. Curvature histograms are shown in fig. 5.7: It can be noticed that the datasets are mostly non-Euclidean, thus offering a good motivation to apply our constant-curvature GCN architectures.

Training Details We trained the baseline models in the same setting as done in [Cha+19]. Namely, for GCN we use one hidden layer of size 16, dropout on the embeddings and the adjacency of rate 0.5 as well as $L^2$-regularization for the weights of the first layer. We used ReLU as the non-linear activation function.
Table 5.3: Average curvature obtained for node classification. \( \mathbb{H} \) and \( \mathbb{S} \) denote hyperbolic and spherical models respectively. Curvature for Pubmed was fixed for the product model. Implementation of these experiments was done in collaboration with co-authors in [BBG20].

<table>
<thead>
<tr>
<th>Model</th>
<th>Citeseer</th>
<th>Cora</th>
<th>Pubmed</th>
<th>Airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{H}^{16} ) (( \kappa )-GCN)</td>
<td>(-1.306 \pm 0.08)</td>
<td>(-1.51 \pm 0.11)</td>
<td>(-1.42 \pm 0.12)</td>
<td>(-0.93 \pm 0.08)</td>
</tr>
<tr>
<td>( \mathbb{S}^{16} ) (( \kappa )-GCN)</td>
<td>(0.81 \pm 0.05)</td>
<td>(1.24 \pm 0.06)</td>
<td>(0.71 \pm 0.15)</td>
<td>(1.49 \pm 0.08)</td>
</tr>
<tr>
<td>Prod ( \kappa )-GCN</td>
<td>([1.21, -0.64] \pm [0.09, 0.07])</td>
<td>([-0.83, -1.61] \pm [0.04, 0.06])</td>
<td>([-1, -1])</td>
<td>([1.23, -0.89] \pm [0.07, 0.11])</td>
</tr>
</tbody>
</table>

For the non-Euclidean architectures, we used a combination of dropout and dropconnect for the non-Euclidean models as reported in [Cha+19], as well as \( L^2 \)-regularization for the first layer. All models have the same number of parameters and for fairness are compared in the same setting, without attention. We use one hidden layer of dimension 16. For the product models we consider two-component spaces (e.g. \( \mathbb{H}^8 \times \mathbb{S}^8 \)) and we split the embedding space into equal dimensions of size 8. We also distribute the input features equally among the components. Non-Euclidean models use the Möbius version of ReLU as activation function. Euclidean parameters use a learning rate of 0.01 for all models using ADAM. The curvatures are learned using gradient descent with a learning rate of 0.01. We show the values of the learned curvatures in Table 5.3.

We use early stopping: we first train for a maximum of 2000 epochs, then we check every 200 epochs for improvement in the validation cross entropy loss; if that is not observed, we stop.

Node classification results. These are shown in table 5.2. It can be seen that our models are competitive with the Euclidean \( \text{GCN} \) considered and outperforms [Cha+19] on Citeseer and Cora, showcasing the benefit of our proposed architecture.
5.4 OTGNN: OPTIMAL TRANSPORT GRAPH NEURAL NETWORKS

We now present our second GNN model. Recall that the aim of the previously introduced model ($\kappa$-GCN) was to leverage geometries of constant sectional curvature for node and graph embedding, replacing Euclidean aggregation by its corresponding analogue respectively in hyperbolic or spherical space.

In the following work, we take a different direction, focusing on smaller input graphs (about less than 50 nodes per graph, for instance small molecules, e.g. antibiotics), and another weakness of current architectures. Indeed, current GNN architectures naively average or sum node embeddings into an aggregated graph representation — potentially losing structural or semantic information. We here introduce OTGNN that computes graph embeddings from OT distances between the set of GNN node embeddings and “prototype” point clouds as free parameters. This allows different prototypes to highlight key facets of different graph subparts — as illustrated in Figure 5.8. We show that our function class on point clouds satisfies a universal approximation theorem, a fundamental property which was lost by sum aggregation. Nevertheless, empirically the model has a natural tendency to collapse back to the standard aggregation during training. We address this optimization issue by proposing an efficient noise contrastive regularizer, steering the model towards truly exploiting the optimal transport geometry. Our model consistently exhibits better generalization performance on several molecular property prediction tasks, yielding also smoother representations.

This work has in part been the subject of a recent submission, currently under reviewed and referred to as an ArXiv preprint [Béc+20]. It was done while the author was at and in collaboration with the Computer Science and Artificial Intelligence Lab (CSAIL) at the Massachusetts Institute of Technology (MIT), funded via the Machine Learning for Pharmaceutical Discovery and Synthesis (MLPDS3) and DARPA AMD programs.

3https://mlpds.mit.edu
Figure 5.8: Intuition for our Wasserstein prototype model. We assume that a few prototypes, e.g. some functional groups, highlight key facets or structural features of graphs in a particular graph classification/regression task at hand. We then express graphs by relating them to these abstract prototypes represented as free point cloud parameters. Note that we do not learn the graph structure of the prototypes.

5.4.1 Model & Architecture

REFORMULATING STANDARD ARCHITECTURES. As mentioned at the end of Section 5.1.4, the final graph embedding \( \mathbf{h} \) obtained by aggregating node embeddings is usually fed to a MLP performing a matrix-multiplication \( \langle \mathbf{R} \mathbf{h} \rangle_i = \langle \mathbf{r}_i, \mathbf{h} \rangle \). Replacing \( \langle \cdot, \cdot \rangle \) by a distance/kernel \( k(\cdot, \cdot) \) allows the processing of more general graph representations than just vectors in \( \mathbb{R}^d \), such as point clouds or adjacency tensors.

FROM A SINGLE POINT TO A POINT CLOUD. We propose to replace the aggregated graph embedding \( \mathbf{h} = \sum_{v \in V} \mathbf{h}_v \) by the point cloud (of unaggregated node embeddings) \( \mathbf{H} = \{ \mathbf{h}_v \}_{v \in V} \), and the inner-products \( \langle \mathbf{h}, \mathbf{r}_i \rangle \) by the below written Wasserstein discrepancy:

\[
\mathcal{W}(\mathbf{H}, \mathbf{Q}_i) := \min_{T \in C_{\mathbf{h}0}} \sum_{v,j} T_{v,j} \mathcal{C}(\mathbf{h}_v, \mathbf{q}_j),
\]  

(5.54)
where the $Q_i = \{q^j_i\}_j$ are point clouds and free parameters, and the cost is chosen as $c = \| \cdot - \cdot \|_2^2$ or $c = -\langle \cdot, \cdot \rangle$. Note that both options yield identical OT plans.

**GREATER REPRESENTATIONAL POWER.** We formulate mathematically in Section 5.4.5 to what extent this kernel has a strictly greater representational power than the kernel corresponding to standard inner-product on top of a sum aggregation, to distinguish between different point clouds. In practice, we would also like our model to exploit its additional representational power. This practical concern is discussed in the next subsection.

**ADDITIONAL REMARKS.** Although we also experimented in practice with the Gromov-Wasserstein and Fused-Gromov-Wasserstein geometries, our best performing models ended up being the simplest Wasserstein models. However, it is worth mentioning that further empirical investigations would be needed before making this empirical finding and definitive conclusion.

**FROM A POINT CLOUD TO AN ADJACENCY TENSOR.** For GNN architectures performing an aggregation edges $\rightarrow$ node such as the DMPNN presented in Section 5.1.4, one can also further remove this aggregation by directly considering the adjacency tensor of edge embeddings as the graph representation $X$, computing a **Gromov-Wasserstein discrepancy** as:

$$GW(X, P_i) := \min_{T \in \mathcal{C}_{X,P}} \sum_{u} \sum_{v} T_{uj} T_{vk} c(X_{uv}, p_j^k),$$

where the $P_i = \{p_{ik}^j\}_{jk}$ are adjacency tensors and free parameters.

### 5.4.2 Contrastive Regularization

What would happen to $\mathcal{W}(H, Q_i)$ if all points $q^j_i$ belonging to point cloud $Q_i$ would collapse to the same point $q_i$? All transport plans would yield the same cost, giving for $c = -\langle \cdot, \cdot \rangle$:

$$\mathcal{W}(H, Q_i) = -\sum_{uj} T_{uj} \langle h_u, q^j_i \rangle = -\langle h, q_i/|V| \rangle.$$  

(5.56)
In this scenario, our proposition would simply over-parametrize the standard Euclidean model.

A first obstacle. Our first empirical trials with OT-enhanced GNNs showed that a model trained with only the Wasserstein component would sometimes perform similarly to the Euclidean baseline, in spite of its greater representational power. Since these two models achieved both similar test and train performance, the absence of improvement in generalization was most likely not due to overfitting.

The cause. Further investigation revealed that the Wasserstein model would naturally displace the points in each of its free point clouds in such a way that the OT plan $T$ obtained by maximizing $\sum_{\alpha \beta} T_{\alpha \beta} \langle h_\alpha, q_\beta \rangle$ was not discriminative, i.e. many other transports would yield a similar Wasserstein cost. Indeed, as shown in Eq. (5.56), if each point cloud collapses to its mean, then the Wasserstein geometry collapses to Euclidean geometry. In this scenario, any transport plan yields the same Wasserstein cost. One may speculate that it was locally easier for the model to extract valuable information if it would behave like the Euclidean component, preventing it from exploring other roads of the
optimization landscape. To better understand this situation, consider the scenario in which a subset of points in a free point cloud “collapses”, i.e. become close to each other (see Figure 5.9), thus sharing similar distances to all the node embeddings of real input graphs. The submatrix of the OT matrix corresponding to these collapsed points can be equally replaced by any other submatrix with the same marginals (i.e. same two vectors obtained by summing rows or columns), meaning that the OT matrix is not discriminative. In general, we want to avoid any two rows or columns in the Wasserstein cost matrix being proportional. An additional problem of point collapsing is that it is a non-escaping situation when using gradient-based learning methods. The reason is that gradients of these collapsed points would become and remain identical, thus nothing will encourage them to “separate” in the future.

**Contrastive regularization.** This observation has lead us to consider the use of a regularizer which would encourage the model to displace its free point clouds such that the OT plans it computes would be discriminative against chosen contrastive transport plans. Namely, consider a point cloud $Y$ of node embeddings and let $T^i$ be an OT plan obtained in the computation of $W(Y, Q_i)$; for each $T^i$, we then build a set $N(T^i) \subset C_{YQ_i}$ of noisy/contrastive transports. If we denote by $W_T(X,Y) := \sum_{kl} T_{kl} c(x_k, y_l)$ the Wasserstein cost obtained for the particular transport $T$, then our contrastive regularization consists in maximizing the term:

$$\sum_i \log \left( \frac{e^{-W_{T^i}(Y,Q_i)}}{e^{-W_{T^i}(Y,Q_i)} + \sum_{T \in N(T^i)} e^{-W_T(Y,Q_i)}} \right), \quad (5.57)$$

which can be interpreted as the log-likelihood that the correct transport $T_i$ be (as it should) a better minimizer of $W_T(Y,Q_i)$ than its negative samples. This can be considered as an approximation of $\log(Pr(T_i \mid Y, Q_i))$, where the partition function is approximated by our selection of negative examples, as done e.g. by Nickel & Kiela [NK17b]. Its effect of is shown in Figure 5.9.

**On the choice of contrastive samples.** The selection of negative examples must reflect the following trade-off: (i) to not be too large, for computational efficiency while (ii) containing sufficiently meaningful and challenging contrastive samples. Our experiments
were conducted with ten negative samples for each correct transport plan. Five of them were obtained by initializing a matrix with uniform
\(i.i.d\) entries from \([0, 10)\) and performing around five Sinkhorn iterations
[Cut13] in order to make the matrix satisfy the marginal constraints. The other five were obtained by randomly permuting the columns of the correct transport plan. The latter choice has the desirable effect of penalizing the points of a free point cloud \(Q_i\) to collapse onto the same point. Indeed, the rows of \(T_i \in C_{HQ_i}\) index points in \(H\), while its columns index points in \(Q_i\). Note that replacing the set \(N(T_i)\) with a singleton \(\{T\}\) for a contrastive random variable \(T\) would let us rewrite Eq. (5.57) as\(^4\) \(\sum_i \log \sigma(W_T - W_{T_i})\), reminiscent of noise contrastive estimation
[GH10].

### 5.4.3 Optimization & Differentiation

Backpropagating gradients through \(\text{OT}\) has been the subject of recent research investigations: Genevay et al. [GPC17] explain how to unroll and differentiate through the Sinkhorn procedure solving \(\text{OT}\), which was extended by Schmitz et al. [Sch+18] to Wasserstein barycenters. However, more recently, Xu [Xu19] proposed to simply invoke the envelop theorem [Afr71] to support the idea of keeping the \(\text{OT}\) plan fixed during the back-propagation of gradients through Wasserstein distances. For the sake of simplicity and training stability, we resort to the latter procedure: keeping \(T\) fixed during back-propagation.

### 5.4.4 Computational Complexity

\textbf{Wasserstein.} Computing the Wasserstein \(\text{OT}\) plan between two point clouds consists in the minimization of a linear function under linear constraints. It can either be performed exactly by using network simplex methods or interior point methods as done by [PW09] in time \(\tilde{O}(n^3)\), or approximately up to \(\epsilon\) via the Sinkhorn algorithm [Cut13] in time \(\tilde{O}(n^2/\epsilon^3)\). More recently, [DGK18] proposed an algorithm solving \(\text{OT}\) up to \(\epsilon\) with time complexity \(\tilde{O}(\min\{n^{9/4}/\epsilon, n^2/\epsilon^2\})\) via a primal-dual method inspired from accelerated gradient descent. In our experiments, we used the Python Optimal Transport (POT) library [FC17]. We noticed empirically that the EMD solver yielded faster and

---

\(^4\)where \(\sigma(\cdot)\) is the sigmoid function.
more accurate solutions than Sinkhorn for our datasets, because the graphs and point clouds were small enough (< 30 elements). However, Sinkhorn may take the lead for larger graphs. An important note is that in order to accurately solve OT via Sinkhorn, one has to choose a low entropy regularization $\epsilon$. However the lower $\epsilon$, the more unstable Sinkhorn becomes. To counter this effect, a stabilized version of Sinkhorn [PC+19] allows for lower choices of $\epsilon$ while remaining stable, at the cost of a slight additional memory requirement.

**GROMOV-WASSERSTEIN.** Computing the Gromov-Wasserstein OT plan between two adjacency tensors consists in the minimization of a non-convex, quadratic function under linear constraints. Note that if $n$ is the number of points, evaluating the Gromov-Wasserstein discrepancy naively for a given transport plan costs $\Omega(n^4)$ operations. However, a smart refactoring [PCS16, proposition 1] allows to reduce it to $O(n^3)$ whenever the ground metric has a particular structure, e.g. $\| \cdot - \cdot \|^2$ or $-\langle \cdot, \cdot \rangle$; note that since we use order-3 adjacency tensors instead of (order-2) similarity matrices, we had to generalize this trick to the multidimensional case (indeed, we compare edge embeddings instead of scalar similarities). Regarding the choice of optimization algorithm, we adapted the exact solver of Flamary et al. [FC17] to multidimensional adjacency tensors, which internally iteratively calls an EMD solver on the successive linearizations described in [PC+19, Eq.(10.28)].

**GENERAL REMARKS.** Significant speed up could potentially be obtained by rewriting the POT library for it to solve OT in batches over GPUs. In our experiments, we ran all jobs on CPUs. A slow-down in speed by a factor 4 was observed from a purely Euclidean to purely Wasserstein models and a factor 10 from purely Euclidean to purely Gromov-Wasserstein.

5.4.5 *Theoretical Analysis: Universality & Definiteness*

In this section we show that the standard architecture lacks a fundamental property of universal approximation of functions defined on point clouds, and that our proposed architecture recovers this property. We will denote by $\mathcal{X}_d^n$ the set of point clouds $X = \{x_i\}_{i=1}^n$ of size $n$ in $\mathbb{R}^d$. 
5.4.5.1 Universality

As seen in Section 5.4.1, we have replaced the sum aggregation — followed by the Euclidean inner-product — by Wasserstein discrepancies. How does this affect the function class and representations?

A common framework used to analyze the geometry inherited from similarities and discrepancies is that of kernel theory. A kernel $k$ on a set $\mathcal{X}$ is a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which can either measure similarities or discrepancies. An important property of a given kernel on a space $\mathcal{X}$ is whether simple functions defined on top of this kernel can approximate any continuous function on the same space. This is called universality: a crucial property to regress unknown target functions.

**Universal kernels.** A kernel $k$ defined on $\mathcal{X}^n$ is said to be universal if the following holds: for any compact subset $\mathcal{X} \subset \mathcal{X}^n$, the set of functions in the form\(^5\) $\sum_{j=1}^{m} \alpha_j \sigma(k(\cdot, \theta_j) + \beta_j)$ is dense in the set $C(\mathcal{X})$ of continuous functions from $\mathcal{X}$ to $\mathbb{R}$, w.r.t the sup norm $\| \cdot \|_{\infty, \mathcal{X}}$, $\sigma$ denoting the sigmoid. Although the notion of universality does not indicate how easy it is in practice to learn the correct function, it at least guarantees the absence of a fundamental bottleneck of the model using this kernel.

**Theorem 5.11.** We have that:

1. The aggregation kernel $\text{agg}$ is not universal.
2. The Wasserstein kernel $\mathcal{W}_{L^2}$ defined in Theorem 5.13 is universal.

**Proof:**

1. Let us first justify why $\text{agg}$ is not universal. Consider a function $f \in C(\mathcal{X})$ such that there exists $X, Y \in \mathcal{X}$ satisfying both $f(X) \neq f(Y)$ and $\sum_k x_k = \sum_i y_i$. Clearly, any function of the form $\sum_i \alpha_i \sigma(\text{agg}(W_i, \cdot) + \theta_i)$ would take equal values on $X$ and $Y$ and hence would not approximate $f$ arbitrarily well.

2. To justify that $\mathcal{W}$ is universal, we take inspiration from the proof of universality of neural networks [Cyb89].

\(^5\)For $m \in \mathbb{N}^*$, $\alpha_j \beta_j \in \mathbb{R}$ and $\theta_j \in \mathcal{X}^n$. 

146
**Notation.** Denote by $M(\mathcal{X})$ the space of finite, signed regular Borel measures on $\mathcal{X}$.

**Definition.** We say that $\sigma$ is discriminatory w.r.t a kernel $k$ if for a measure $\mu \in M(\mathcal{X})$,

$$\int_{\mathcal{X}} \sigma(k(Y, X) + \theta) d\mu(X) = 0$$

for all $Y \in \mathcal{X}_d^n$ and $\theta \in \mathbb{R}$ implies that $\mu \equiv 0$.

We start by reminding a lemma coming from the original paper on the universality of neural networks by Cybenko [Cyb89].

**Lemma.** If $\sigma$ is discriminatory w.r.t. $k$ then $k$ is universal.

**Proof:** Let $S$ be the subset of functions of the form $\sum_{i=1}^{m} \alpha_i \sigma(k(\cdot, Q_i) + \theta_i)$ for any $\theta_i \in \mathbb{R}$, $Q_i \in \mathcal{X}_d^n$ and $m \in \mathbb{N}^*$ and denote by $\bar{S}$ the closure of $S$ in $C(\mathcal{X})$. Assume by contradiction that $\bar{S} \neq C(\mathcal{X})$. By the Hahn-Banach theorem, there exists a bounded linear functional $L$ on $C(\mathcal{X})$ such that for all $h \in \bar{S}$, $L(h) = 0$ and such that there exists $h' \in C(\mathcal{X})$ s.t. $L(h') \neq 0$. By the Riesz representation theorem, this bounded linear functional is of the form:

$$L(h) = \int_{X \in \mathcal{X}} h(X) d\mu(X),$$

for all $h \in C(\mathcal{X})$, for some $\mu \in M(\mathcal{X})$. Since $\sigma(k(Q, \cdot) + \theta)$ is in $\bar{S}$, we have

$$\int_{\mathcal{X}} \sigma(k(Q, X) + \theta) d\mu(X) = 0$$

for all $Q \in \mathcal{X}_d^n$ and $\theta \in \mathbb{R}$. Since $\sigma$ is discriminatory w.r.t. $k$, this implies that $\mu = 0$ and hence $L \equiv 0$, which is a contradiction with $L(h') \neq 0$. Hence $\bar{S} = C(\mathcal{X})$, i.e. $S$ is dense in $C(\mathcal{X})$ and $k$ is universal.

□

Now let us look at the part of the proof that is new.

---

$^6$W.r.t the topology defined by the sup norm $\|f\|_{\infty, \mathcal{X}} := \sup_{X \in \mathcal{X}} |f(X)|$. 

---

147
Lemma. \( \sigma \) is discriminatory w.r.t. \( \mathcal{W}_{L2} \).

Proof: Note that for any \( X, Y, \theta, \varphi \), when \( \lambda \to +\infty \) we have that \( \sigma(\lambda(\mathcal{W}_{L2}(X, Y) + \theta) + \varphi) \) goes to 1 if \( \mathcal{W}_{L2}(X, Y) + \theta > 0 \), to 0 if \( \mathcal{W}_{L2}(X, Y) + \theta < 0 \) and to \( \sigma(\varphi) \) if \( \mathcal{W}_{L2}(X, Y) + \theta = 0 \).

Denote by \( \Pi_{Y, \theta} := \{ X \in \mathcal{X} \mid \mathcal{W}_{L2}(X, Y) - \theta = 0 \} \) and \( B_{Y, \theta} := \{ X \in \mathcal{X} \mid \sqrt{\mathcal{W}_{L2}(X, Y)} < \theta \} \) for \( \theta \geq 0 \) and \( \emptyset \) for \( \theta < 0 \). By the Lebesgue Bounded Convergence Theorem we have:

\[
0 = \int_{X \in \mathcal{X}} \lim_{\lambda \to +\infty} \sigma(\lambda(\mathcal{W}_{L2}(X, Y) - \theta) + \varphi) d\mu(X) = \sigma(\varphi) \mu(\Pi_{Y, \theta}) + \mu(\mathcal{X} \setminus B_{Y, \sqrt{\theta}}).
\]

Since this is true for any \( \varphi \), it implies that \( \mu(\Pi_{Y, \theta}) = \mu(\mathcal{X} \setminus B_{Y, \sqrt{\theta}}) = 0 \). From \( \mu(\mathcal{X}) = 0 \) (because \( B_{Y, \sqrt{\theta}} = \emptyset \) for \( \theta < 0 \)), we also have \( \mu(B_{Y, \sqrt{\theta}}) = 0 \). Hence \( \mu \) is zero on all balls defined by the metric \( \sqrt{\mathcal{W}_{L2}} \).

From the Hahn decomposition theorem, there exist disjoint Borel sets \( P, N \) such that \( \mathcal{X} = P \cup N \) and \( \mu = \mu^+ - \mu^- \) where \( \mu^+(A) := \mu(A \cap P) \), \( \mu^-(A) := \mu(A \cap N) \) for any Borel set \( A \) with \( \mu^+, \mu^- \) being positive measures. Since \( \mu^+ \) and \( \mu^- \) coincide on all balls on a finite dimensional metric space, they coincide everywhere \([Hof76]\) and hence \( \mu \equiv 0 \).

\[\square\]

Universality of the Wasserstein kernel \( \mathcal{W}_{L2} \) essentially comes from the fact that its square-root defines a metric, and in particular from the axiom of separation of distances: if \( d(x, y) = 0 \) then \( x = y \).

5.4.5.2 Definiteness

For the sake of simplified mathematical analysis, similarity kernels are often required to be positive definite (p.d.), which corresponds to discrepancy kernels being conditionally negative definite (c.n.d.). Although such a property has the benefit of yielding the mathematical framework of Reproducing Kernel Hilbert Spaces, it essentially implies linearity, i.e. the possibility to embed the geometry defined by that kernel in a linear vector space.

We now show that, interestingly, the Wasserstein kernel we used does not satisfy this property, and hence constitutes an interesting instance of a universal, non p.d. kernel. Let us remind these notions.
Kernel definiteness. A kernel $k$ is positive definite (p.d.) on $\mathcal{X}$ if for $n \in \mathbb{N}^*$, $x_1, ..., x_n \in \mathcal{X}$ and $c_1, ..., c_n \in \mathbb{R}$, we have $\sum_{ij} c_i c_j k(x_i, x_j) \geq 0$. It is conditionally negative definite (c.n.d.) on $\mathcal{X}$ if for $n \in \mathbb{N}^*$, $x_1, ..., x_n \in \mathcal{X}$ and $c_1, ..., c_n \in \mathbb{R}$ such that $\sum_i c_i = 0$, we have $\sum_{ij} c_i c_j k(x_i, x_j) \leq 0$.

These two notions relate to each other via the below result [BTB05]:

**Proposition 5.12.** Let $k$ be a symmetric kernel on $\mathcal{X}$, let $x_0 \in \mathcal{X}$ and define the kernel:

$$\tilde{k}(x, y) := -\frac{1}{2} [k(x, y) - k(x, x_0) - k(y, x_0) + k(x_0, x_0)].$$  \hspace{1cm} (5.58)

Then $\tilde{k}$ is p.d. if and only if $k$ is c.n.d. Example: $k = \| \cdot - \cdot \|_2^2$ and $x_0 = 0$ yield $\tilde{k} = \langle \cdot, \cdot \rangle$.

The aggregating kernel against which we wish to compare the Wasserstein kernel is the inner-product over a summation of the points in the point clouds: $\operatorname{agg}(X, Y) := \langle \sum_i x_i, \sum_j y_j \rangle$.

One can easily show that this also defines a p.d. kernel, and that $\operatorname{agg}(\cdot, \cdot) \leq n^2 \mathcal{W}(\cdot, \cdot)$. However, the Wasserstein kernel is not p.d., as shown by the below theorem.

**Theorem 5.13.** We have that:

1. The (similarity) Wasserstein kernel $\mathcal{W}_{\text{dot}}$ is not positive definite;
2. The (discrepancy) Wasserstein kernel $\mathcal{W}_{\text{L2}}$ is not conditionally negative definite, where:

$$\mathcal{W}_{\text{L2}}(X, Y) := \min_{T \in XY} \sum_{ij} T_{ij} \| x_i - y_j \|_2^2, \quad \mathcal{W}_{\text{dot}}(X, Y) := \max_{T \in XY} \sum_{ij} T_{ij} \langle x_i, y_j \rangle.$$  \hspace{1cm} (5.59)

**Proof:**

1. We build a counter example. We consider 4 point clouds of size $n = 2$ and dimension $d = 2$. First, define $u_i = ([i/2], i \% 2)$ for $i \in \{0, ..., 3\}$. Then take $X_1 = \{u_0, u_1\}$, $X_2 = \{u_0, u_2\}$, $X_3 = \{u_0, u_3\}$ and $X_4 = \{u_1, u_2\}$. On the one hand, if $\mathcal{W}(X_i, X_j) = 0$, then all vectors in the two point clouds are orthogonal, which can only happen for $\{i, j\} =$
{1, 2}. On the other hand, if $\mathcal{W}(X_i, X_j) = 1$, then either $i = j = 3$ or $i = j = 4$. This yields the following Gram matrix

\[
(W(X_i, X_j))_{0 \leq i, j \leq 3} = \begin{pmatrix}
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 2 & 1 \\
1 & 1 & 1 & 2 \\
\end{pmatrix}
\] (5.60)

whose determinant is $-1/16$, which implies that this matrix has a negative eigenvalue.

2. This comes from proposition 5.12. Choosing $k = \mathcal{W}_{L2}$ and $x_0 = 0$ to be the trivial point cloud made of $n$ times the zero vector yields $\tilde{k} = \mathcal{W}_{\text{dot}}$. Since $\tilde{k}$ is not positive definite from the previous point of the theorem, $k$ is not conditionally negative definite from proposition 5.12.

\[\square\] 5.4.5.3 \textit{Shape of the Optimal Transport Plan for Point Clouds of Same Size}

The below result describes the shape of OT plans for point clouds of same size. For the sake of curiosity, we also illustrate in Figure 5.3 the OT for point clouds of different sizes. We note that non-square transports seem to remain relatively sparse as well. This is in line with our empirical observations.

\textbf{Proposition 5.14.} For $X, Y \in \mathcal{X}_{n,d}$ there exists a rescaled permutation matrix $\frac{1}{n} (\delta_{\sigma(j)})_{1 \leq i, j \leq n}$ which is an optimal transport plan, i.e.

\[
\mathcal{W}_{L2}(X, Y) = \frac{1}{n} \sum_{j=1}^{n} \|x_{\sigma(j)} - y_j\|_2^2, \quad \mathcal{W}_{\text{dot}}(X, Y) = \frac{1}{n} \sum_{j=1}^{n} \langle x_{\sigma(j)}, y_j \rangle. \quad (5.61)
\]

\textbf{Proof.} It is well known from Birkhoff’s theorem that every squared doubly-stochastic matrix is a convex combination of permutation matrices. Since the Wasserstein cost for a given transport $T$ is a linear function, it is also a convex/concave function, and hence it is maximized/minimized over the convex compact set of couplings at one of its extremal points, namely one of the rescaled permutations, yielding the desired result. \[\square\]
5.4.6 Additional Related Work

GNN were introduced by Gori et al. [GMS05] and Scarselli et al. [Sca+08] as a form of recurrent neural network. GCN made their first appearance later on in various forms. Duvenaud et al. [Duv+15] and Atwood et al. [AT16] proposed a propagation rule inspired from convolution and diffusion, although these methods do not scale to graphs with either large degree distribution or node cardinality, respectively. Niepert et al. [NAK16] defined a GCN as a 1D convolution on a chosen node ordering. Kearnes et al. [Kea+16] also used graph convolutions with great success to generate high quality molecular fingerprints. Efficient spectral methods were also proposed [Bru+13]; [DBV16a]. Kipf & Welling [KW17b] simplified their propagation rule, motivated as well from spectral graph theory [HVG11], achieving impressive empirical results. Most of these different architectures were later unified into a message passing neural networks framework by Gilmer et al. [Gil+17], which applies them to molecular property prediction. A directed variant of message passing was motivated by Dai et al. [DDS16], which was later used to improve state-of-the-art in molecular property prediction on a wide variety of datasets by ChemProp [Yan+19]. Another notable application includes recommender systems [Yin+18a]. Ying et al. [Yin+18b] proposed DiffPool, which performs a pooling operation for GNN in a hierarchical fashion. Inspired by DeepSets [Zah+17], Xu et al. [Xu+19b] suggest both a simplification and generalization of certain GNN architectures, which should theoretically be powerful enough to discriminate between any different local neighborhoods, provided that hidden dimensions grow as much as the input size. Other recent approaches suggest to modify the sum-aggregation of node embeddings in the GCN architecture with the aim to preserve more information [Kon+18]; [Pei+20]. On the other hand, Hongbin et al. [Pei+20] propose to preserve more semantic information by performing a bi-level aggregation which depends on the local geometry of the neighborhood of the given node in the graph. Other recent geometry-inspired GNN include adaptations to embeddings lying in hyperbolic spaces [Cha+19]; [LNK19] or spaces of constant sectional curvature [BBG20].

[Ye+20] also suggests an architecture enhancement incorporating information relating to the discrete Ricci-Ollivier curvature of the graph.
We refer the reader to [Wu+19] for a more extensive review of the GNN literature.

**Optimal Transport.** Comprehensive and extensive introductions to the mathematical and computational aspects of OT are respectively provided by Villani [Vil08] and Peyr’è & Cuturi [PC+19].

Gromov-Wasserstein OT was recently introduced in the mathematical community by Mémoli [Mém11], in order to compare metric measure spaces. Peyré et al. [PCS16] later contributed to its computational aspect by suggesting to solve via iteratively optimizing an intermediate Wasserstein cost. Efficient approximations of both Wasserstein and Gromov-Wasserstein distance were proposed by [Bon+15] and [Tit+19b] respectively, who propose “slice” the distributions by taking random 1D projections, and then solve the problem in closed form in 1D. Other approaches learning embeddings involving Wasserstein distances include Wasserstein dictionary learning [Sch+18] and the GOT framework for graph comparison [Mar+19], while Togninalli et al. [Tog+19] propose a non-parametric graph kernel involving Wasserstein distances. On the other hand, Gromov-Wasserstein OT was applied to compare word embeddings in different languages [AJ18]. The very related problem of graph matching [ZS18a] was also tackled via the use of Gromov-Wasserstein geometry [Xu+19a]. Titouan et al. [Tit+19a] proposes a “fused” version of Wasserstein and Gromov-Wasserstein OT, aimed at capturing interactions between node feature distributions and structural information in graphs, while Xu [Xu19] suggests to decompose graphs into “factors” in the Gromov-Wasserstein geometry via the use of Fréchet barycenters. Finally, note that although our work was empirically validated for relatively small graphs, one could explore potential extensions to large graphs via hierarchical [SS13], structured [AJJ18] or large scale stochastic [Gen+16] optimal transport.

### 5.5 Empirical Validation on Small Molecular Graphs

#### 5.5.1 Experimental Setup

We test our model on 4 benchmark molecular property prediction datasets [Yan+19] including both regression (ESOL, Lipophilicity) and classification (BACE, BBBP) tasks. These datasets cover a variety of different complex chemical properties (e.g. ESOL - water solubility, LIPO - octanol/water distribution coefficient, BACE - inhibition of
human β-secretase, BBBP - blood-brain barrier penetration). We show that our models improves over state-of-the-art baselines. **GNN** is the state-of-the-art GNN that we use as our primary baseline, as well as the underlying graph model for our prototype models. Its architecture is described in section 5.1.4. **ProtoW-L2/Dot** is the model that treats point clouds as point sets, and computes the Wasserstein distances to each point cloud (using either L2 distance or (minus) dot product cost functions) as the molecular embedding. **ProtoS-L2** is a special case of **ProtoW-L2**, in which the point clouds have a **single** point. Instead of using Wasserstein distances, we instead just compute simple Euclidean distances between the aggregated graph embedding and point clouds. Here, we omit using dot product distances, as that model is mathematically equivalent to the **GNN** model.

We use the the POT library [FC17] to compute Wasserstein distances using the network simplex algorithm (ot.emd), which we find empirically to be faster than using the Sinkhorn algorithm, due to the small size of the graphs present in our datasets. We define the cost matrix by taking the pairwise L2 or negative dot product distances. As mentioned in Section 5.4.3, we fix the transport plan, and only backprop through the cost matrix for computational efficiency. Additionally, since the sum aggregation operator easily accounts for the sizes of input graphs, we multiply the OT distance between two point clouds by their respective sizes. To avoid the problem of point clouds collapsing, we employ the contrastive regularizer defined in Section 5.4.2. More details about experimental setup in Section 5.5.3. We also tried extensions to our prototype models using Gromov-Wasserstein geometry. However, we found that these models proved much more difficult to optimize in practice.

### 5.5.2 Experimental Results

**Regression and Classification.** The results on the property prediction datasets are shown in Table 5.4. We find that the prototype models outperform the **GNN** on all 4 property prediction tasks, showing that this model paradigm can be more powerful than conventional **GNN** models. Moreover, the prototype models using Wasserstein distance (**ProtoW-L2/Dot**) achieves better performance on 3 out of 4 of the datasets compared to the prototype model using only Euclidean dis-
Table 5.4: Results of different models on the property prediction datasets. **Best** in bold, second best underlined. Proto methods are ours. Lower RMSE is better, while higher AUC is better. The prototype-based models generally outperform the GNN, and the Wasserstein models perform better than the model using only simple Euclidean distances, suggesting that the Wasserstein distance provides more powerful representations. Wasserstein models trained with contrastive regularization as described in section 5.4.2 outperform those without. Implementation of these experiments was done in collaboration with co-authors in [Béc+20].

<table>
<thead>
<tr>
<th># graphs</th>
<th>ESOL (RMSE)</th>
<th>Lipo (RMSE)</th>
<th>BACE (AUC)</th>
<th>BBBP (AUC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 1128</td>
<td>.635 ± .027</td>
<td>.646 ± .041</td>
<td>.865 ± .013</td>
<td>.915 ± .010</td>
</tr>
<tr>
<td>n = 4199</td>
<td>.611 ± .034</td>
<td>.580 ± .016</td>
<td>.865 ± .010</td>
<td>.918 ± .009</td>
</tr>
<tr>
<td>n = 1512</td>
<td>.608 ± .029</td>
<td>.637 ± .018</td>
<td>.867 ± .014</td>
<td>.919 ± .009</td>
</tr>
<tr>
<td>n = 2039</td>
<td>.594 ± .031</td>
<td>.629 ± .015</td>
<td>.871 ± .014</td>
<td>.919 ± .009</td>
</tr>
<tr>
<td>GNN/Chemprop</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProtoW-Dot</td>
<td>.616 ± .028</td>
<td>.615 ± .025</td>
<td>.870 ± .012</td>
<td>.920 ± .010</td>
</tr>
<tr>
<td>ProtoW-L2</td>
<td>.605 ± .029</td>
<td>.604 ± .014</td>
<td>.873 ± .015</td>
<td>.920 ± .010</td>
</tr>
<tr>
<td>ProtoW-L2 (no reg.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProtoW-Dot (no reg.)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Distances (ProtoS-L2). This confirms our hypothesis that Wasserstein distance confers greater discriminative power compared to traditional aggregation methods (summation).

Noise Contrastive Regularizer. Without any constraints, the Wasserstein prototype model will often collapse the set of points in a point cloud into a single point. As mentioned in Section 5.4.2, we use a contrastive regularizer to force the model to meaningfully distribute point clouds in the embedding space. We show 2D embeddings in Fig. 5.9, illustrating that without contrastive regularization, prototype point clouds are often displaced close to their mean, while regularization forces them to nicely scatter.
5.5 Empirical Validation on Small Molecular Graphs

Learned Embedding Space: Qualitative and Quantitative Results. To further support our claim that Wasserstein distance provides more powerful representations, we also examine the embedding space of the GNN baseline and our Wasserstein model. Using the best performing models, we compute the pairwise difference in embedding vectors and the labels for each test data point on the ESOL dataset. Then, we compute two measures of rank correlation, Spearman correlation coefficient ($\rho$) and Pearson correlation coefficient ($r$). This procedure is reminiscent of evaluation tasks for word embeddings w.r.t how semantic similarity in embedding space correlates with human labels [LSM13].

Our ProtoW-L2 achieves better $\rho$ and $r$ scores compared to the GNN model (Table 5.5), that indicating our Wasserstein model constructs more meaningful embeddings with respect to the label distribution. Indeed, Figure 5.10 plots the pairwise scores for the GNN model (left) and the ProtoW-L2 model (right). Our ProtoW-L2 model, trained to optimize distances in the embedding space, produces more meaningful representations with respect to the label of interest.

Table 5.5: The Spearman and Pearson correlation coefficients on the ESOL dataset for the GNN and ProtoW-L2 model w.r.t. the pairwise difference in embedding vectors and labels. Implementation of these experiments was done in collaboration with co-authors in [Béc+20].

<table>
<thead>
<tr>
<th></th>
<th>Spearman $\rho$</th>
<th>Pearson $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNN</td>
<td>.424 ± .029</td>
<td>.393 ± .049</td>
</tr>
<tr>
<td>ProtoS-L2</td>
<td>.561 ± .087</td>
<td>.414 ± .141</td>
</tr>
<tr>
<td>ProtoW-Dot</td>
<td>.592 ± .150</td>
<td>.559 ± .216</td>
</tr>
<tr>
<td>ProtoW-L2</td>
<td><strong>.815 ± .026</strong></td>
<td><strong>.828 ± .020</strong></td>
</tr>
</tbody>
</table>

Moreover, as can be seen in Figure 5.11, our model also provides more robust molecular embeddings compared to the baseline, in the following sense: we observe that a small perturbation of a molecular embedding corresponds to a small change in predicted property value – a desirable phenomenon that holds rarely for the baseline GNN model. Qualitatively, this is shown in Figure 5.11. Our Wasserstein prototype
models yields smoother heatmaps, which is desirable for molecular optimization in the latent space via gradient methods.

5.5.3 Further Experimental Details

Each dataset is split randomly 5 times into 80%:10%:10% train, validation and test sets. For each of the 5 splits, we run each model 5 times to reduce the variance in particular data splits (resulting in each model being run 25 times). We search hyperparameters for each split of the data, and then take the average performance over all the splits. The hyperparameters are separately searched for each data split, so that the model performance is based on a completely unseen test set, and that there is no data leakage across data splits. The models are trained for 150 epochs with early stopping if the validation error has not improved in 50 epochs and a batch size of 16. We train the models using the Adam optimizer with a learning rate of 5e-4. For the prototype models, we use different learning rates for the GNN and the point clouds (5e-4 and 5e-3 respectively), because empirically we find that the gradients are much smaller for the point clouds. The molecular datasets used for experiments here are small in size (varying from 1-4k data points), so this is a fair method of comparison, and is indeed what is done in other works on molecular property prediction [Yan+19]. More details in Table 5.6.

Figure 5.10: Left: GNN model; Right: ProtoW-L2 model; Comparison of the correlation between graph embedding distances (X axis) and label distances (Y axis) on the ESOL dataset.
5.6 CONCLUSIVE SUMMARY

In this chapter, we introduced two natural extensions of GCNs: one to the stereographic models of both positive and negative curvatures in a unified manner, another dispensing with the final step of node aggregation by leveraging OT. We show how κ-GCN permits to smoothly interpolate between positive and negative curvature, allowing the curvature of the model to be trained independent of an initial sign choice, and we introduce an efficient regularizer which prevents our OTGNN from collapsing back to standard aggregation.

Figure 5.11: 2D heatmaps of T-SNE [MH08] projections of molecular embeddings (before the last linear layer) w.r.t. their associated predicted labels. Heat colors are interpolations based only on the test molecules from each dataset. Comparing (a) vs (b) and (c) vs (d), we can observe a smoother space of our model compared to the GNN baseline as explained in the main text.
Table 5.6: The parameters for our models (the prototype models all use the same GNN base model), and the values that we used for hyperparameter search. When there is only a single value in the search list, it means we did not search over this value, and used the specified value for all models.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Search Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n_epochs</td>
<td>{150}</td>
<td>Number of epochs trained</td>
</tr>
<tr>
<td>batch_size</td>
<td>{16}</td>
<td>Size of each batch</td>
</tr>
<tr>
<td>lr</td>
<td>{5e-4}</td>
<td>Overall learning rate for model</td>
</tr>
<tr>
<td>lr_pc</td>
<td>{5e-3}</td>
<td>Learning rate for the free parameter point clouds</td>
</tr>
<tr>
<td>n_layers</td>
<td>{5}</td>
<td>Number of layers in the GNN</td>
</tr>
<tr>
<td>n_hidden</td>
<td>{50, 200}</td>
<td>Size of hidden dimension in GNN</td>
</tr>
<tr>
<td>n_ffn_hidden</td>
<td>{1e2, 1e3, 1e4}</td>
<td>Size of the output feed forward layer</td>
</tr>
<tr>
<td>dropout_gnn</td>
<td>{0.}</td>
<td>Dropout probability for GNN</td>
</tr>
<tr>
<td>dropout_fnn</td>
<td>{0., 0.1, 0.2}</td>
<td>Dropout probability for feed forward layer</td>
</tr>
<tr>
<td>n_pc</td>
<td>{10, 20}</td>
<td>Number of free parameter point clouds in prototype models</td>
</tr>
<tr>
<td>pc_size</td>
<td>{10}</td>
<td>Number of points in free parameter point clouds</td>
</tr>
<tr>
<td>pc_hidden</td>
<td>{5, 10}</td>
<td>Size of hidden dimension in point clouds</td>
</tr>
<tr>
<td>nc_coef</td>
<td>{0., 0.01, 0.1, 1}</td>
<td>Coefficient for noise contrastive regularization</td>
</tr>
</tbody>
</table>
In this dissertation, we started by analyzing the intrinsic geometry of data manifolds embedded in Euclidean spaces, seeking to unveil how CNN representations get disentangled during training. We then took interest into explicitly embedding data into non-Euclidean domains, with a particular emphasis on Riemannian manifolds of constant sectional curvature: hyperbolic and spherical spaces. We developed a word embedding method in the Poincaré disk, as well as a family of elementary functions playing the role of neural networks. Then, motivated by the necessity of powerful Riemannian adaptive optimization algorithms to let our hyperbolic methods compete with their Euclidean counterparts, we developed Riemannian analogues to ADAM and ADAGRAD. Finally, we extended current state-of-the-art GCN and DMPNN architectures to spaces of constant curvature, and dispensed with their — potentially destructive — final step node aggregation by leveraging tools from OT geometry.

A natural and interesting extension to our work would be to combine our proposed $\kappa$-GCN and OTGNN architectures into a $\kappa$-OTGNN.


BIBLIOGRAPHY


