Hyperbolic Graph Convolutional Neural Networks

Ines Chami*‡ Rex Ying* † Christopher Ré† Jure Leskovec†

†Department of Computer Science, Stanford University
‡Institute for Computational and Mathematical Engineering, Stanford University
{chami, rexying, chrismre, jure}@cs.stanford.edu

October 11, 2019

Abstract

Graph convolutional neural networks (GCNs) map nodes in a graph to Euclidean embeddings, which have been shown to incur a large distortion when embedding real-world graphs with scale-free or hierarchical structure. Hyperbolic geometry offers an exciting alternative, as it enables embeddings with much smaller distortion. However, extending GCNs to hyperbolic geometry presents several unique challenges. It is not clear how to define neural network operations, such as feature transformation and aggregation, in hyperbolic space. Furthermore, since input features are often Euclidean, it is unclear how to transform the features into hyperbolic embeddings with the right amount of curvature. Here we propose Hyperbolic Graph Convolutional Neural Network (HYPERGCN), the first inductive hyperbolic GCN that leverages both the expressiveness of GCNs and hyperbolic geometry to learn inductive node representations for hierarchical and scale-free graphs. We derive GCNs operations in the hyperboloid model of hyperbolic space and map Euclidean input features to embeddings in hyperbolic spaces with different trainable curvatures at each layer. Experiments demonstrate that HYPERGCN learns embeddings that preserve hierarchical structure, and leads to improved performance when compared to Euclidean analogs, even with very low dimensional embeddings: compared to state-of-the-art GCNs, HYPERGCN achieves an error reduction of up to 63.1% in ROC AUC for link prediction (LP) and of up to 47.5% in F1 score for node classification (NC), also improving state-of-the art on the Pubmed dataset.

1 Introduction

Graph convolutional neural networks (GCNs) are state-of-the-art models for representation learning in graphs, where nodes of the graph are mapped to points in Euclidean space [14, 20, 39, 43]. However, many real-world graphs, such as protein interaction networks and social networks, often exhibit scale-free or hierarchical structure [7, 48] and Euclidean embeddings, used in existing GCNs, have a high distortion when embedding these graph structures [6, 30]. In particular, if volume in graphs is defined as the number of nodes within some distance of a center node, it grows exponentially with respect to that distance for regular trees. However, the volume of balls in Euclidean space only grows polynomially with respect to the radius, leading to high distortion tree embeddings [32, 33], while in hyperbolic space, this volume grows exponentially.

Hyperbolic geometry offers an exciting alternative as it enables embeddings with much smaller distortion. However, current hyperbolic embedding techniques only account for the graph structure and do not leverage rich node features [27]. For instance, Poincaré embeddings [28] capture the hyperbolic properties of real graphs by learning shallow embeddings with hyperbolic distance metric and Riemannian optimization. Compared to deep alternatives such as GCNs, shallow embeddings do not take into account features of nodes, lack scalability, and inductive capability. Furthermore, in practice, optimization in hyperbolic space is challenging.

While extending GCNs to hyperbolic geometry has the potential to lead to more faithful embeddings and accurate models, it also poses challenges: (1) input node features are usually Euclidean and it is not clear how to optimally use

*Equal contribution

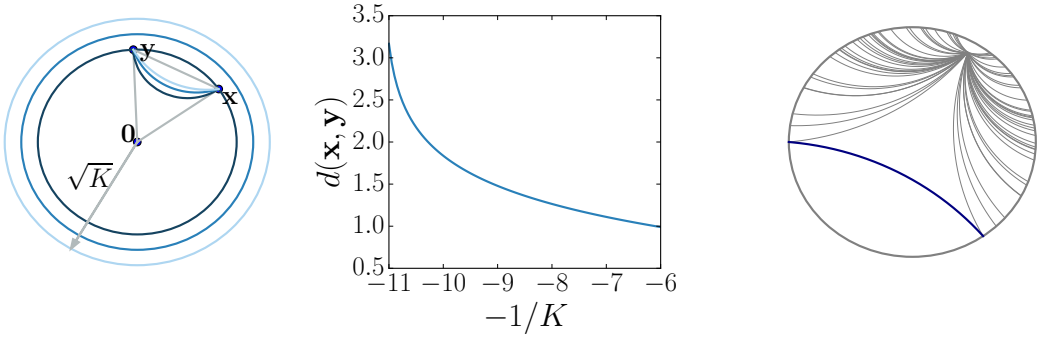


Figure 1: Left: Poincaré disk geodesics (shortest path) connecting $\mathbf{x}$ and $\mathbf{y}$ for different curvatures. As curvature $(-1/K)$ decreases, the distance between $\mathbf{x}$ and $\mathbf{y}$ increases and the geodesics lines get closer to the origin. Center: Hyperbolic distance vs curvature. Right: Poincaré geodesic lines.

them as input to hyperbolic neural networks, (2) it is not clear how to perform set aggregation, a key step in message passing, in hyperbolic space, and (3) one needs to choose hyperbolic spaces with the right curvature at every layer of GCN.

Here we propose Hyperbolic Graph Convolutional Networks (HYPERGCN), a class of graph representation learning models that combine the expressiveness of GCNs and hyperbolic geometry to learn improved representations for real-world hierarchical and scale-free graphs in inductive settings. In HYPERGCN, we solve the above challenges: (1) We derive the core transformation of GCNs in the hyperboloid model of hyperbolic space to transform the input features which lie in Euclidean space into hyperbolic embeddings. (2) We introduce a hyperbolic attention-based aggregation scheme that captures node hierarchies. (3) We apply feature transformations in hyperbolic spaces of different trainable curvatures at different layers, to learn hyperbolic embeddings that preserve the graph structure and a notion of hierarchy for nodes in the graph.

The transformation between different hyperbolic spaces at different layers allows HYPERGCN to find the best geometry of hidden layers to achieve low distortion and high separation of class labels. Our approach jointly trains the weights for hyperbolic graph convolution operators, layer-wise curvatures and hyperbolic attention weights to learn inductive embeddings that reflect hierarchies in graphs.

Compared to Euclidean GCNs, HYPERGCN offers improved expressiveness for scale-free or hierarchical graph data. We demonstrate the efficacy of HYPERGCN on LP and NC tasks on a wide range of open datasets of graphs which exhibit different extent of scale-free/hierarchical structure. HYPERGCN achieves new state-of-the-art results on the standard PUBMED benchmark. Experiments show that HYPERGCN significantly outperforms all Euclidean-based state-of-the-art graph neural networks on scale-free/hierarchical graphs and reduces error from 11.5% up to 47.5% on NC tasks, and from 28.2% up to 63.1% on LP tasks. Finally, we also analyze the notion of hierarchy learned by HYPERGCN and show how the geometry of embeddings transform from entirely Euclidean features to purely hyperbolic embeddings.

2 Related Work

The problem of graph representation learning belongs to the field of geometric deep learning. There exist two major types of approaches: transductive shallow embeddings and inductive GCNs.

Transductive, shallow embeddings. The first type of approach attempts to optimize node embeddings as parameters by minimizing a reconstruction error. In other words, the mapping from nodes in graph to embeddings is an embedding look-up. Examples include matrix factorization [23, 3] and random walk methods [29, 12]. Shallow embedding methods have also been developed in hyperbolic geometry [28, 27] for reconstructing trees [33] and graphs [21, 5], or embedding text [37]. However, shallow (Euclidean and hyperbolic) embedding methods have three major downsides: (1) They fail to leverage rich node feature information, which can be crucial in tasks such as node classification. (2) These methods

are transductive, and therefore cannot be used for inference on unseen graphs. And, (3) they scale poorly, as the number of model parameters grows linearly with the number of nodes.

(Euclidean) Graph Neural Networks. Instead of learning shallow embeddings, an alternative approach is to learn a mapping from input graph structure as well as node features to embeddings, parameterized by neural networks [20, 24, 14, 39, 45, 43]. While various Graph Neural Network architectures resolve the disadvantages of shallow embeddings, they generally embed nodes into a Euclidean space, which leads to a large distortion when embedding real-world graphs with scale-free or hierarchical structure. Our work builds on GNNs and extends them to hyperbolic geometry.

Hyperbolic Neural Networks. Hyperbolic geometry has been applied to neural networks, to problems of computer vision or natural language processing [17, 13, 36, 8]. More recently, hyperbolic neural networks [10] were proposed, where core neural network operations are in hyperbolic space. In contrast to previous work, we derive core neural network operations in a more stable model of hyperbolic space, and propose new operations for set aggregation, which enable HYPERGCN to perform deep graph convolutions in hyperbolic space with trainable curvature.

3 Background

Problem setting. Without loss of generality we describe graph representation learning on a single graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$, and let $(\mathbf{x}_i^{0,E})_{i\in\mathcal{V}}$ be d-dimensional input node features. We use the superscript ^{E} to indicate that node features lie in a Euclidean space and use X^H to denote hyperbolic features. The goal in graph representation learning is to learn a mapping f which maps nodes to embedding vectors

$$f : (\mathcal{V}, \mathcal{E}, (\mathbf{x}_i^{0,E})_{i\in\mathcal{V}}) \to Z \in \mathbb{R}^{|\mathcal{V}|\times d'},$$

where $d' \ll |\mathcal{V}|$. These embeddings should capture both structural and semantic information and can then be used as input for downstream tasks such as NC or LP.

Review of Graph Convolution Networks (GCNs). Let $\mathcal{N}(i) = \{j : (i,j) \in \mathcal{E}\}$ denote the set of neighbors of $i \in \mathcal{V}$, $(W^\ell, \mathbf{b}^\ell)$ be weights and bias parameters for layer ℓ, and $\sigma(\cdot)$ be a non-linear activation function. The general GCN message passing rule at layer ℓ for node i consists of

$$\mathbf{h}_i^{\ell,E} = W^\ell \mathbf{x}_i^{\ell-1,E} + \mathbf{b}^\ell \qquad \text{(feature transform)} \tag{1}$$

$$\mathbf{x}_i^{\ell,E} = \sigma(\mathbf{h}_i^{\ell,E} + \sum_{j\in\mathcal{N}(i)} w_{ij}\mathbf{h}_j^{\ell,E}) \qquad \text{(neighborhood aggregation)} \tag{2}$$

where aggregation weights w_{ij} can be computed with different mechanisms [20, 14, 39]. The message passing is performed for multiple layers to propagate messages over neighborhoods. Unlike shallow methods, GCNs leverage node features and can be applied to unseen graphs in inductive settings.

The hyperboloid model of hyperbolic space. We review basic concepts of hyperbolic geometry that serve as building blocks for HYPERGCN. Hyperbolic geometry is a non-Euclidean geometry with a constant negative curvature, where curvature measures how a geometric object deviates from a flat plane (*cf.* [31] for an introduction to differential geometry). Here, we work with the hyperboloid model for its simplicity and its numerical stability [27]. We generalize results for curvature -1 to any constant negative curvature, as this allows us to learn curvature as a model parameter, leading to better optimization (*cf.* Section 4.5 for more details).

Hyperboloid manifold. We first introduce our notation for the hyperboloid model of hyperbolic space. Let $\langle .,. \rangle_{\mathcal{L}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$ denote the Minkowski inner product, $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \coloneqq -x_0 y_0 + x_1 y_1 + \ldots + x_d y_d$. We denote $\mathbb{H}^{d,K}$ as the hyperboloid manifold in d dimensions with constant negative **curvature** $-1/K$ $(K > 0)$, and $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ the (Euclidean) **tangent space** centered at point $\mathbf{x}$

$$\mathbb{H}^{d,K} \coloneqq \{\mathbf{x} \in \mathbb{R}^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -K, x_0 > 0\} \qquad \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K} \coloneqq \{\mathbf{v} \in \mathbb{R}^{d+1} : \langle \mathbf{v}, \mathbf{x} \rangle_{\mathcal{L}} = 0\}. \tag{3}$$

Now for $\mathbf{v}$ and $\mathbf{w}$ in $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$, $g_{\mathbf{x}}^K(\mathbf{v}, \mathbf{w}) \coloneqq \langle \mathbf{v}, \mathbf{w} \rangle_{\mathcal{L}}$ is a Riemannian metric tensor [31] and $(\mathbb{H}^{d,K}, g_{\mathbf{x}}^K)$ is a Riemannian manifold with negative curvature $-1/K$. $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ is a local, first-order approximation of the hyperbolic manifold at $\mathbf{x}$ and the restriction of the Minkowski inner product to $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ is positive definite. $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ is useful to perform Euclidean operations undefined in hyperbolic space and we denote $||\mathbf{v}||_{\mathcal{L}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathcal{L}}}$ the norm of $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$.

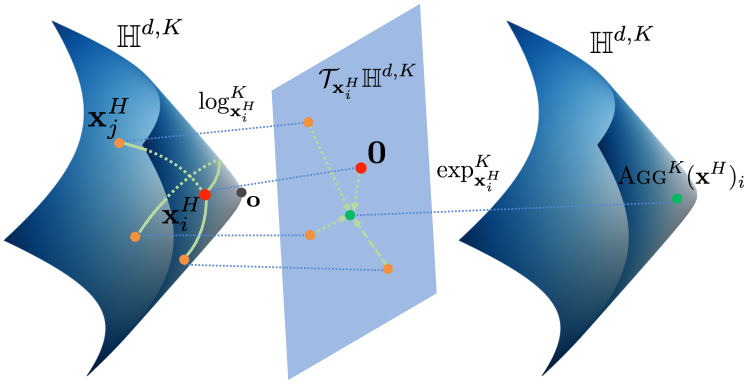


Figure 2: HYPERGCN aggregation (Equation 9)

Geodesics and induced distances. Next, we introduce the notion of geodesics and distances in manifolds, which are generalizations of shortest paths in graphs or straight lines in Euclidean geometry (Figure 1). Geodesics and distance functions are particularly important in graph embedding algorithms, as a common optimization objective is minimizing geodesic distances between connected nodes. Let $\mathbf{x} \in \mathbb{H}^{d,K}$ and $\mathbf{u} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$. Assume $\mathbf{u}$ is unit-speed, *i.e.* $\langle \mathbf{u}, \mathbf{u} \rangle_{\mathcal{L}} = 1$. We have the following result.

Proposition 3.1. *Let $\mathbf{x} \in \mathbb{H}^{d,K}$, $\mathbf{u} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ be unit-speed. The unique unit-speed geodesic $\gamma_{\mathbf{x}\to\mathbf{u}}(\cdot)$ such that $\gamma_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{x}$, $\dot{\gamma}_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{u}$ is $\gamma^K_{\mathbf{x}\to\mathbf{u}}(t) = \cosh\left(\frac{t}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{t}{\sqrt{K}}\right)\mathbf{u}$, and the intrinsic distance function between two points $\mathbf{x}, \mathbf{y}$ in $\mathbb{H}^{d,K}$ is then*

$$d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \sqrt{K}\text{arcosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}/K). \tag{4}$$

Exponential and logarithmic maps. Mapping between tangent space and hyperbolic space is done by the exponential and logarithmic maps. Given $\mathbf{x} \in \mathbb{H}^{d,K}$ and a tangent vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$, the exponential map $\exp^K_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K} \to \mathbb{H}^{d,K}$ is the map that assigns to $\mathbf{v}$ the point $\exp^K_{\mathbf{x}}(\mathbf{v}) := \gamma(1)$, where γ is the unique geodesic satisfying $\gamma(0) = \mathbf{x}$ and $\dot{\gamma}(0) = \mathbf{v}$. The logarithmic map is the reverse map that maps back to the tangent space at $\mathbf{x}$ such that $\log^K_{\mathbf{x}}(\exp^K_{\mathbf{x}}(\mathbf{v})) = \mathbf{v}$. In general Riemannian manifolds, these operations are only defined locally but in the hyperbolic space, they form a bijection between the hyperbolic space and the tangent space at a point. We have the following direct expressions of the exponential and the logarithmic maps which allow us to perform operations on points on the hyperboloid manifold by mapping them to tangent spaces and and vice-versa.

Proposition 3.2. *For $\mathbf{x} \in \mathbb{H}^{d,K}$, $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ and $\mathbf{y} \in \mathbb{H}^{d,K}$ such that $\mathbf{v} \neq \mathbf{0}$ and $\mathbf{y} \neq \mathbf{x}$, the exponential and logarithmic maps of the hyperboloid model are given by*

$$\exp^K_{\mathbf{x}}(\mathbf{v}) = \cosh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}, \quad \log^K_{\mathbf{x}}(\mathbf{y}) = d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y})\frac{\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}\mathbf{x}}{||\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}}}.$$

4 Hyperbolic Graph Convolutional Networks

We introduce HYPERGCN, a generalization of inductive GCNs in hyperbolic geometry that benefits from the expressiveness of both graph neural networks and hyperbolic embeddings. We introduce the essential components of HYPERGCN. First, since input features are often Euclidean, we derive a mapping from Euclidean features to hyperbolic space. Next, we derive the two components of graph convolution: The analogs of the Euclidean feature transformation and aggregation (Equations 1, 2) in the hyperboloid model. Finally, we introduce the HYPERGCN algorithm with trainable curvature.

4.1 Mapping from Euclidean to hyperbolic spaces

HYPERGCN first maps input features to the hyperboloid manifold via the $\exp$ map. Let $\mathbf{x}^{0,E} \in \mathbb{R}^d$ denote input Euclidean features. For instance, these features could be produced by pre-trained Euclidean neural networks. Let $\mathbf{o} := \{\sqrt{K}, 0, \ldots, 0\} \in \mathbb{H}^{d,K}$ denote the north pole (origin) in $\mathbb{H}^{d,K}$, which we use as a reference point to perform tangent space operations. We have $\langle (0, \mathbf{x}^{0,E}), \mathbf{o} \rangle = 0$. Therefore, we interpret $(0, \mathbf{x}^{0,E})$ as a point in $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K}$ and use Proposition 3.2 to map it to $\mathbb{H}^{d,K}$ with

$$\mathbf{x}^{0,H} = \exp_{\mathbf{o}}^K((0, \mathbf{x}^{0,E})) = \left(\sqrt{K}\cosh\left(\frac{||\mathbf{x}^{0,E}||_2}{\sqrt{K}} \right), \sqrt{K}\sinh\left(\frac{||\mathbf{x}^{0,E}||_2}{\sqrt{K}} \right) \frac{\mathbf{x}^{0,E}}{||\mathbf{x}^{0,E}||_2} \right). \tag{5}$$

4.2 Feature transform in hyperbolic space

The feature transform in Equation 1 is used in GCN to map the embedding space of one layer to the next layer embedding space and capture large neighborhood structures. We now want to learn transformations of points on the hyperboloid manifold. However, there is no notion of vector space structure in hyperbolic space. We build upon Hyperbolic Neural Network (HNN) [10] and derive transformations in the hyperboloid model. The main idea is to leverage the $\exp$ and $\log$ maps in Corollary 3.2 so that we can use the tangent space $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K}$ to perform Euclidean transformations.

Hyperboloid linear transform. Linear transformation requires a multiplication of the embedding vector by a weight matrix, followed by bias translation. To compute matrix vector multiplication, we first use the $\log$ map to project hyperbolic points $\mathbf{x}^H$ to $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K}$. Thus the matrix representing the transform is defined on the tangent space, which is Euclidean and isomorphic to $\mathbb{R}^d$. We then project the vector back to the manifold using the exponential map. Let W be a $d' \times d$ weight matrix, we define the hyperboloid matrix mutiplication as

$$W \otimes^K \mathbf{x}^H \coloneqq \exp_{\mathbf{o}}^K(W \log_{\mathbf{o}}^K(\mathbf{x}^H)), \tag{6}$$

where $\log_{\mathbf{o}}^K(\cdot)$ is on $\mathbb{H}^{d,K}$ and $\exp_{\mathbf{o}}^K(\cdot)$ maps to $\mathbb{H}^{d',K}$. In order to perform bias addition, we use a result from the HNN model and define $\mathbf{b}$ as an Euclidean vector located at $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K}$. We then parallel transport $\mathbf{b}$ to the tangent space of the hyperbolic point of interest and map it to the manifold. The hyperboloid bias addition is then defined as

$$\mathbf{x}^H \oplus^K \mathbf{b} \coloneqq \exp_{\mathbf{x^H}}^K(P_{\mathbf{o} \to \mathbf{x}^H}^K(\mathbf{b})), \tag{7}$$

where $P_{\mathbf{o} \to \mathbf{x}^H}^K(\cdot)$ is the parallel transport from $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d',K}$ to $\mathcal{T}_{\mathbf{x}^H}\mathbb{H}^{d',K}$ (*c.f.* Appendix A for details).

4.3 Neighborhood aggregation on the hyperboloid manifold

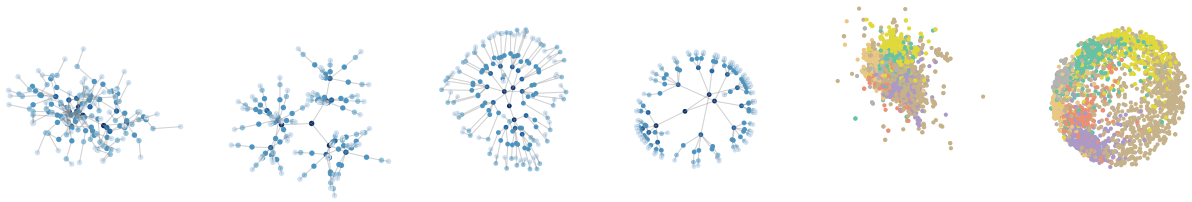
Aggregation (Equation 2) is a crucial step in GCNs as it captures neighborhood structures and features with message passing. Suppose that $\mathbf{x}_i$ aggregates information from its neighbors $(\mathbf{x}_j)_{j \in \mathcal{N}(i)}$ with weights $(w_j)_{j \in \mathcal{N}(i)}$. Mean aggregation in Euclidean GCN computes the weighted average $\sum_{j \in \mathcal{N}(i)} w_j \mathbf{x}_j$. An analog of mean aggregation in hyperbolic space is the Frechet mean [9], which has no closed form solution. Instead, we propose to perform aggregation in tangent spaces using hyperbolic attention.

Attention based aggregation. Attention in GCNs learns a notion of neighbors' importance, and aggregates neighbors' messages according to their importance to the center node. However, attention on Euclidean embeddings does not take into account the nodes' hierarchies. Thus, we further propose hyperbolic attention-based aggregation. Given hyperbolic embeddings $(\mathbf{x}_i^H, \mathbf{x}_j^H)$, we first map $\mathbf{x}_i^H$ and $\mathbf{x}_j^H$ to the tangent space of the origin to compute attention weights w_{ij} with concatenation and Euclidean Multi-layer Percerptron (MLP). We then propose a hyperbolic aggregation to average nodes' representations

$$w_{ij} = \text{SOFTMAX}_{j \in \mathcal{N}(i)}(\text{MLP}(\log_{\mathbf{o}}^K(\mathbf{x}_i^H) || \log_{\mathbf{o}}^K(\mathbf{x}_j^H))) \tag{8}$$

$$\text{AGG}^K(\mathbf{x}^H)_i = \exp_{\mathbf{x}_i^H}^K\left(\sum_{j \in \mathcal{N}(i)} w_{ij} \log_{\mathbf{x}_i^H}^K(\mathbf{x}_j^H) \right). \tag{9}$$

Note that our proposed aggregation is directly performed in the tangent space of each center point $\mathbf{x}_i^H$, as this is where the Euclidean approximation is best (*cf.* Figure 2). We show in our ablation experiments (*cf.* Table 2) that this local



(a) GCN layers. (b) HYPERGCN layers. (c) GCN (left), HYPERGCN (right).

Figure 3: Visualization of embeddings for LP on DISEASE and NC on CORA (visualization on the Poincaré disk for HYPERGCN). (a) GCN embeddings in first and last layers for DISEASE LP hardly capture hierarchy (depth indicated by color). (b) In contrast, HYPERGCN preserves node hierarchies. (c) On CORA NC, HYPERGCN leads to better class separation (indicated by different colors).

aggregation outperforms aggregation in tangent space at the origin ($\text{AGG}_{\mathbf{o}}$), due to the fact that relative distances have lower distortion with our approach.

Non-linear activation with different curvatures. Analogous to Euclidean aggregation (Equation 2), HYPERGCN uses a non-linear activation function, $\sigma(\cdot)$ such that $\sigma(0) = 0$, to learn non-linear transformations. Given hyperbolic curvatures $-1/K_{\ell-1}, -1/K_\ell$ at layer $\ell-1$ and ℓ respectively, we introduce a hyperbolic non-linear activation $\sigma^{\otimes^{K_{\ell-1},K_\ell}}$ with different curvatures. This step is crucial as it allows us to smoothly vary curvature at each layer. More concretely, HYPERGCN applies the Euclidean non-linear activation in $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{d,K_{\ell-1}}$ and then maps back to $\mathbb{H}^{d,K_\ell}$

$$\sigma^{\otimes^{K_{\ell-1},K_\ell}}(\mathbf{x}^H) = \exp_{\mathbf{o}}^{K_\ell}(\sigma(\log_{\mathbf{o}}^{K_{\ell-1}}(\mathbf{x}^H))). \tag{10}$$

Note that in order to apply the exponential map at $\mathbf{o}$, we need to make sure that points are located in the corresponding tangent space. Fortunately, tangent spaces of the north pole are shared across hyperboloid manifolds of the same dimension that have different curvatures, making Equation 10 mathematically correct.

4.4 HYPERGCN architecture

Having introduced all the building blocks of HYPERGCN, we now summarize the model architecture. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and input Euclidean features $(\mathbf{x}^{0,E})_{i\in\mathcal{V}}$, the first layer of HYPERGCN is a mapping from Euclidean to hyperbolic space detailed in Section 4.1. HYPERGCN stacks multiple hyperbolic graph convolution layers. At each layer HYPERGCN transforms and aggregates neighbour's embeddings in the tangent space of the center node and projects the result to a hyperbolic space with different curvature. Hence the message passing in a HYPERGCN layer is

$$\mathbf{h}_i^{\ell,H} = (W^\ell \otimes^{K_{\ell-1}} \mathbf{x}_i^{\ell-1,H}) \oplus^{K_{\ell-1}} \mathbf{b}^\ell \qquad \text{(hyperbolic feature transform)} \tag{11}$$

$$\mathbf{y}_i^{\ell,H} = \text{AGG}^{K_{\ell-1}}(\mathbf{h}^{\ell,H})_i \qquad \text{(attention-based neighborhood aggregation)} \tag{12}$$

$$\mathbf{x}_i^{\ell,H} = \sigma^{\otimes^{K_{\ell-1},K_\ell}}(\mathbf{y}_i^{\ell,H}) \qquad \text{(non-linear activation with different curvatures)} \tag{13}$$

where, $-1/K_{\ell-1}$ and $-1/K_\ell$ are the hyperbolic curvatures at layer $\ell-1$ and ℓ respectively. Hyperbolic embeddings $(\mathbf{x}^{L,H})_{i\in\mathcal{V}}$ at the last layer with curvature $-1/K_L$ can then be used to predict node attributes or links. For LP, we use the Fermi-Dirac decoder [22, 28], a generalization of sigmoid, to compute probability scores for edges

$$p((i,j) \in \mathcal{E} | \mathbf{x}_i^{L,H}, \mathbf{x}_j^{L,H}) = \left[e^{(d_{\mathcal{L}}^{K_L}(\mathbf{x}_i^{L,H}, \mathbf{x}_j^{L,H})^2 - r)/t} + 1\right]^{-1}, \tag{14}$$

where $d_{\mathcal{L}}^{K_L}(\cdot,\cdot)$ is the hyperbolic distance and r and t are hyper-parameters. We then train HYPERGCN by minimizing the cross-entropy loss using negative sampling.

For NC, we map the output of the last HYPERGCN layer to the tangent space of the origin with the logarithmic map $\log_{\mathbf{o}}^{K_L}(\cdot)$ and then perform Euclidean multinomial logistic regression. Note that another possibility is to directly

	Dataset	DISEASE		DISEASE-M		HUMAN PPI		AIRPORT		PUBMED		CORA	
	Hyperbolicity δ	$\delta = 0$		$\delta = 0$		$\delta = 1$		$\delta = 1$		$\delta = 3.5$		$\delta = 11$	
	Method	LP	NC	LP	NC	LP	NC	LP	NC	LP	NC	LP	NC
Shallow	EUC	59.8 ± 2.0	32.5 ± 1.1	-	-	-	-	92.0 ± 0.0	60.9 ± 3.4	83.3 ± 0.1	48.2 ± 0.7	82.5 ± 0.3	23.8 ± 0.7
Shallow	HYP [28]	63.5 ± 0.6	45.5 ± 3.3	-	-	-	-	94.5 ± 0.0	70.2 ± 0.1	87.5 ± 0.1	68.5 ± 0.3	87.6 ± 0.2	22.0 ± 1.5
Shallow	EUC-MIXED	49.6 ± 1.1	35.2 ± 3.4	-	-	-	-	91.5 ± 0.1	68.3 ± 2.3	86.0 ± 1.3	63.0 ± 0.3	84.4 ± 0.2	46.1 ± 0.4
Shallow	HYP-MIXED	55.1 ± 1.3	56.9 ± 1.5	-	-	-	-	93.3 ± 0.0	69.6 ± 0.1	83.8 ± 0.3	73.9 ± 0.2	85.6 ± 0.5	45.9 ± 0.3
NN	MLP	72.6 ± 0.6	28.8 ± 2.5	55.3 ± 0.5	55.9 ± 0.3	67.8 ± 0.2	55.3 ± 0.4	89.8 ± 0.5	68.6 ± 0.6	84.1 ± 0.9	72.4 ± 0.2	83.1 ± 0.5	51.5 ± 1.0
NN	HNN[10]	75.1 ± 0.3	41.0 ± 1.8	60.9 ± 0.4	56.2 ± 0.3	72.9 ± 0.3	59.3 ± 0.4	90.8 ± 0.2	80.5 ± 0.5	94.9 ± 0.1	69.8 ± 0.4	89.0 ± 0.1	54.6 ± 0.4
GNN	GCN[20]	64.7 ± 0.5	69.7 ± 0.4	66.0 ± 0.8	59.4 ± 3.4	77.0 ± 0.5	69.7 ± 0.3	89.3 ± 0.4	81.4 ± 0.6	91.1 ± 0.5	78.1 ± 0.2	90.4 ± 0.2	81.3 ± 0.3
GNN	GAT [39]	69.8 ± 0.3	70.4 ± 0.4	69.5 ± 0.4	62.5 ± 0.7	76.8 ± 0.4	70.5 ± 0.4	90.5 ± 0.3	81.5 ± 0.3	91.2 ± 0.1	79.0 ± 0.3	**93.7** ± 0.1	**83.0** ± 0.7
GNN	SAGE [14]	65.9 ± 0.3	69.1 ± 0.6	67.4 ± 0.5	61.3 ± 0.4	78.1 ± 0.6	69.1 ± 0.3	90.4 ± 0.5	82.1 ± 0.5	86.2 ± 1.0	77.4 ± 2.2	85.5 ± 0.6	77.9 ± 2.4
GNN	SGC [42]	65.1 ± 0.2	69.5 ± 0.2	66.2 ± 0.2	60.5 ± 0.3	76.1 ± 0.2	71.3 ± 0.1	89.8 ± 0.3	80.6 ± 0.1	94.1 ± 0.0	78.9 ± 0.0	91.5 ± 0.1	81.0 ± 0.1
Ours	HYPERGCN	**90.8** ± 0.3	**74.5** ± 0.9	**78.1** ± 0.4	**72.2** ± 0.5	**84.5** ± 0.4	**74.6** ± 0.3	**96.4** ± 0.1	**90.6** ± 0.2	**96.3** ± 0.0	**80.3** ± 0.3	92.9 ± 0.1	79.9 ± 0.2
Ours	(%) ERR RED	-63.1%	-13.8%	-28.2%	-25.9%	-29.2%	-11.5%	-60.9%	-47.5%	-27.5%	-6.2%	+12.7%	+18.2%

Table 1: ROC AUC for Link Prediction (LP) and F1 score for Node Classification (NC) tasks. For inductive datasets, we only evaluate inductive methods since shallow methods cannot generalize to unseen nodes/graphs. We report graph hyperbolicity values δ (lower is more hyperbolic).

classify points on the hyperboloid manifold using the hyperbolic multinomial logistic loss [10]. This method performs similarly to Euclidean classification (*cf.* [10] for an empirical comparison). Finally, we also add a LP regularization objective in NC tasks, to encourage embeddings at the last layer to preserve the graph structure.

4.5 Trainable curvature

We further analyze the effect of trainable curvatures in HYPERGCN. Theorem 4.1 (proof in Appendix B) shows that assuming infinite precision, for the LP task, we can achieve the same performance for varying curvatures with an affine invariant decoder, by scaling embeddings.

Theorem 4.1. *For any hyperbolic curvatures $-1/K, -1/K' < 0$, for any node embeddings $H = \{\mathbf{h}_i\} \subset \mathbb{H}^{d,K}$ of a graph G, we can find $H' \subset \mathbb{H}^{d,K'}$, $H' = \{\mathbf{h}'_i | \mathbf{h}'_i = \sqrt{\frac{K'}{K}}\mathbf{h}_i\}$, such that the reconstructed graph from H' via the Fermi-Dirac decoder is the same as the reconstructed graph from H, with different decoder parameters (r, t) and (r', t').*

However, despite the same expressive power, adjusting curvature at every layer is important for good performance in practice due to factors in limited machine precision and normalization. First, with very low or very high curvatures, the scaling factor $\frac{K'}{K}$ in Theorem 4.1 becomes close to 0 or very large, and limited machine precision results in large error due to rounding. This is supported by Figure 4 and Table 2 where adjusting and training curvature lead to significant performance gain. Second, the norm of hidden layers that achieve the same local minimum in training also vary by a factor of $\sqrt{K}$. In practice, however, optimization is much more stable when the values are normalized [15]. In the context of HYPERGCN, trainable curvature provides us a natural way to learn embeddings of the right scale at each layer, improving optimization. See Figure 4 for the effect of decreasing curvature ($K = +\infty$ is the Euclidean case) on LP performance.

5 Experiments

We comprehensively evaluate our experiments on a variety of networks, on both node classification and link prediction tasks, in transductive and inductive settings. We compare the performance of HYPERGCN against a variety of shallow and GNN-based baselines. We further use visualizations to investigate the expressiveness of HYPERGCN in link prediction tasks, and also demonstrate its ability to learn GCN functions that leverage the notion of hierarchy of nodes in graphs.

5.1 Experimental setup

Datasets. We use a variety of open transductive and inductive datasets that we detail below (more details in the Appendix). We compute Gromov's $\delta-$hyperbolicity [1, 26, 16], a notion from group theory that measures how tree-like a graph is. The lower δ, the more hyperbolic is the graph dataset $\delta = 0$ for trees. We conjecture that HYPERGCN works better on graphs with small δ-hyperbolicity.

1. **Citation networks**. CORA [34] and PUBMED [25] are standard benchmarks describing citation networks where nodes represent scientific papers, edges are citations between them, and node labels are academic (sub)areas. CORA contains 2,708 machine learning papers divided into 7 classes while PUBMED has 19,717 publications in the area of medicine grouped in 3 classes.
2. **Disease propagation tree**. We simulate the SIR disease spreading model [2], where the label of a node is whether the node was infected or not. Based on the model, we build tree networks, where the node features indicate the susceptibility to the disease. We build transductive and inductive variants of this dataset, namely DISEASE and DISEASE-M, with 1,044 and 43193 nodes respectively.
3. **Protein-protein interactions (PPI) networks**. We compose a dataset of PPI networks containing the human proteins in all tissues [35], where edges represent interactions between proteins. The node classification task is to predict the stem cell growth rate after 19 days [38]. The 16-dimensional feature for each node represents the RNA expression levels of the corresponding proteins, and we perform log transform on features.
4. **Flight networks**. AIRPORT is a transductive dataset where nodes represent airports and edges represent the airline routes as from OpenFlights.org. Compared to previous compilations [47], this dataset has larger size (2236 nodes). We also augment the graph with geographic information (longitude, latitude and altitude), and GDP of the country where the airport belongs to. We use the population of the country where the airport belongs to as the label for node classification.

Baselines. For shallow methods, we consider Euclidean embeddings (EUC) and Poincaré embeddings (HYP) [28]. We conjecture that Poincaré embeddings will outperform Euclidean embeddings on hierarchical graphs. For fair comparison with HYPERGCN which leverages node features, we also consider EUC-MIXED and HYP-MIXED baselines, where we concatenate the corresponding shallow embeddings with node features, followed by a MLP to predict node labels or links. For state-of-the-art Euclidean GNN models, we consider GCN [20], GraphSAGE (SAGE) [14], Graph Attention Networks (GAT) [39] and Simplified Graph Convolution (SGC) [42][1]. We also consider feature-based approaches: MLP and its hyperbolic variant (HNN) [10], which does not utilize the graph structure.

Training. For all methods, we perform a hyper-parameter search on validation set over initial learning rate, patience for learning rate decay, weight decay, dropout[2], number of layers, and activation functions. We measure performance on the final test set over 10 random seeds. For fairness, we also control the number of dimensions to be the same (16) for all methods. We optimize all models with Adam [18], except Poincaré embeddings which are optimized with RiemannianSGD [4, 46]. More details are included in the Appendix. We open source our implementation[3] of HYPERGCN and baselines.

Evaluation metric. In transductive LP tasks, we randomly split edges into $85/5/10\%$ for training, validation and test sets. For transductive NC, we use $30/10/60\%$ split for nodes in all datasets, except CORA and PUBMED where we use standard splits [44, 20] with 20 examples per class for training. One of the main advantages of HYPERGCN over related hyperbolic graph embedding is its inductive capability. For inductive tasks, the split is performed across graphs. All nodes/edges in training graphs are considered the training set, and the model is asked to predict node class or unseen links for test graphs. Following previous works, we evaluate link prediction by measuring area under the ROC curve on the test set and evaluate node classification performance by measuring F1 score, except for CORA and PUBMED where we report accuracy to compare our results to the literature.

[1] The equivalent of GCN in link prediction is GAE [19]. We did not compare link prediction GNNs based on shallow embeddings such as [47] since they are not inductive.

[2] HYPERGCN uses DropConnect [40], as described in Appendix C.

[3] Code available at https://github.com/ines-chami/hypergcn. We provide HYPERGCN implementations for the hyperboloid and the Poincaré model. In our experiments, we find that both models give similar performance but the hyperboloid model offers more stable optimization. This statement is also supported by the fact that the Poincaré distance function is numerically unstable due to the denominator term [27].

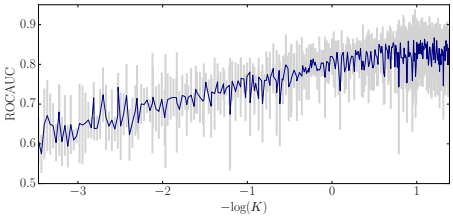


Figure 4: Decreasing curvature ($-1/K$) improves link prediction performance on DISEASE.

Method	DISEASE	AIRPORT
HYPERGCN	78.4 ± 0.3	91.8 ± 0.3
HYPERGCN-ATT$_{\mathbf{o}}$	80.9 ± 0.4	92.3 ± 0.3
HYPERGCN-ATT	82.0 ± 0.2	92.5 ± 0.2
HYPERGCN-C	89.1 ± 0.2	94.9 ± 0.3
HYPERGCN-ATT-C	$\mathbf{90.8} \pm 0.3$	$\mathbf{96.4} \pm 0.1$

Table 2: ROC AUC for link prediction on AIRPORT and DISEASE datasets.

5.2 Results

Table 1 reports the performance of HYPERGCN in comparison to baseline methods. HYPERGCN works best in inductive scenarios where both node features and network topology play an important role. The performance gain of HYPERGCN with respect to Euclidean GNN models is correlated with graph hyperbolicity. HYPERGCN achieves an average of 45.4% (LP) and 12.3% (NC) error reduction compared with the best deep baselines for graphs with high hyperbolicity (low δ), suggesting that GNNs can significantly benefit from hyperbolic geometry, especially in link prediction tasks. Furthermore, the performance gap between HYPERGCN and HNN suggests that neighborhood aggregation has been effective in learning node representations in graphs. For example, in disease spread datasets, both Euclidean attention and hyperbolic geometry lead to significant improvement of HYPERGCN over other baselines. This can be explained by the fact that in disease spread trees, parent nodes contaminate their children. HYPERGCN can successfully model these asymmetric and hierarchical relationships with hyperbolic attention and improves performance over all baselines.

On the CORA dataset with low hyperbolicity, HYPERGCN does not outperform Euclidean GNNs, suggesting that Euclidean geometry is better for its underlying graph structure. However, for small dimensions, HYPERGCN is still significantly more effective than GCN even with CORA. Figure 3c shows 2-dimensional HYPERGCN and GCN embeddings trained with LP objective, where colors denote the label class. HYPERGCN achieves much better label class separation.

5.3 Analysis

Ablations. We further analyze the effect of proposed components in HYPERGCN, namely hyperbolic attention (ATT) and trainable curvature (C) on AIRPORT and DISEASE datasets in Table 2. We observe that both attention and trainable curvature lead to performance gains over HYPERGCN with fixed curvature and no attention. Furthermore, our attention model ATT outperforms ATT$_{\mathbf{o}}$ (aggregation in tangent space at $\mathbf{o}$) and we conjecture that this is because the local Euclidean average is a better approximation near the center point rather than near $\mathbf{o}$. Finally, the addition of both ATT and C improves performance even further, suggesting that both components are important in the HYPERGCN model.
Visualizations. We first visualize the GCN and HYPERGCN embeddings at the first and last layers in Figure 3. We train HYPERGCN with 3-dimensional hyperbolic embeddings and map them to the Poincaré disk which is better for visualization. In contrast to GCN, the tree structure is preserved in HYPERGCN, where nodes close to the center are of higher hierarchy in the tree. HYPERGCN smoothly transforms Euclidean features to Hyperbolic embeddings that preserve node hierarchies.

Figure 5 shows the attention weights in the 2-hop neighborhood of a center node (red) for the DISEASE dataset. The red node is the node where we compute attention. The darkness of the color for other nodes denotes their hierarchy. The attention weights for nodes in the neighborhood are visualized by the intensity of edges. We observe that in HYPERGCN the center node pays more attention to its (grand)parent. In contrast to Euclidean GAT, our aggregation with attention in hyperbolic space allows us to pay more attention to nodes with high hierarchy. Such attention is crucial to good performance in DISEASE, because only sick parents will propagate the disease to their children.

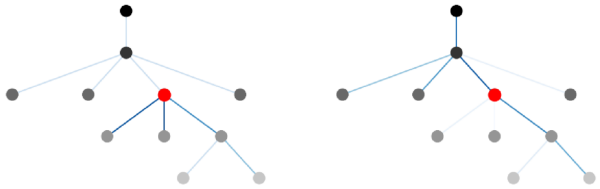


Figure 5: Attention: Euclidean GAT (left), HYPERGCN (right). Each graph represents a 2-hop neighborhood of the DISEASE-M dataset.

6 Conclusion

We introduced HYPERGCN, a novel architecture that learns hyperbolic embeddings using graph convolutional networks. In HYPERGCN, the Euclidean input features are successively mapped to embeddings in hyperbolic spaces with trainable curvatures at every layer. HYPERGCN achieves new state-of-the-art in learning embeddings for real-world hierarchical and scale-free graphs.

Acknowledgments

Jure Leskovec is a Chan Zuckerberg Biohub investigator. This research has been supported in part by NSF OAC-1835598, DARPA MCS, DARPA ASED, ARO MURI, Boeing, Docomo, Hitachi, Huawei, JD, Siemens and Stanford Data Science Initiative. We gratefully acknowledge the support of DARPA under Nos. FA87501720095 (D3M), FA86501827865 (SDH), and FA86501827882 (ASED); NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); ONR under No. N000141712266 (Unifying Weak Supervision); the Moore Foundation, NXP, Xilinx, LETI-CEA, Intel, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, the Okawa Foundation, American Family Insurance, Google Cloud, Swiss Re, and members of the Stanford DAWN project: Teradata, Facebook, Google, Ant Financial, NEC, VMWare, and Infosys. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of DARPA, NIH, ONR, or the U.S. Government.

References

[1] Aaron B Adcock, Blair D Sullivan, and Michael W Mahoney. Tree-like structure in large social and information networks. In *2013 IEEE 13th International Conference on Data Mining*, pages 1–10. IEEE, 2013.

[2] Roy M Anderson and Robert M May. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[4] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 2013.

[5] Benjamin Paul Chamberlain, James Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*, 2017.

[6] Wei Chen, Wenjie Fang, Guangda Hu, and Michael W Mahoney. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics*, 9(4):434–491, 2013.

[7] Aaron Clauset, Cristopher Moore, and Mark EJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98, 2008.

[8] Bhuwan Dhingra, Christopher J Shallue, Mohammad Norouzi, Andrew M Dai, and George E Dahl. Embedding text in hyperbolic spaces. *NAACL HLT*, 2018.

[9] Maurice Fréchet. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, 1948.

[10] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *NeurIPS*, 2018.

[11] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning ICML*, 2018.

[12] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[13] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. In *ICLR*, 2019.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[16] Edmond Jonckheere, Poonsuk Lohsoonthorn, and Francis Bonahon. Scaled gromov hyperbolic graphs. *Journal of Graph Theory*, 2008.

[17] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. *arXiv preprint arXiv:1904.02239*, 2019.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[21] Robert Kleinberg. Geographic routing using hyperbolic space. In *IEEE International Conference on Computer Communications*, 2007.

[22] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 2010.

[23] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 1964.

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

[25] Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. 2012.

[26] Onuttom Narayan and Iraj Saniee. Large-scale curvature of networks. *Physical Review E*, 84(6):066108, 2011.

[27] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. *ICML*, 2018.

[28] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NIPS*, 2017.

[29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[30] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical review E*, 2003.

[31] Joel W Robbin and Dietmar A Salamon. Introduction to differential geometry.

[32] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Re. Representation tradeoffs for hyperbolic embeddings. In *ICML*, 2018.

[33] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, 2011.

[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.

[35] Damian Szklarczyk, John H Morris, Helen Cook, Michael Kuhn, Stefan Wyder, Milan Simonovic, Alberto Santos, Nadezhda T Doncheva, Alexander Roth, Peer Bork, et al. The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible. *Nucleic acids research*, 2016.

[36] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Hyperbolic representation learning for fast and efficient neural question answering. In *WSDM*, 2018.

[37] Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. In *ICLR*, 2019.

[38] Joyce van de Leemput, Nathan C Boles, Thomas R Kiehl, Barbara Corneo, Patty Lederman, Vilas Menon, Changkyu Lee, Refugio A Martinez, Boaz P Levi, Carol L Thompson, et al. Cortecon: a temporal transcriptome analysis of in vitro human cerebral cortex development from human embryonic stem cells. *Neuron*, 2014.

[39] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

[40] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[41] Richard C Wilson, Edwin R Hancock, Elżbieta Pekalska, and Robert PW Duin. Spherical and hyperbolic embeddings of data. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2255–2269, 2014.

[42] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

[43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ICLR*, 2019.

[44] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ICML*, 2016.

[45] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018.

[46] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. In *NIPS*, 2016.

[47] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *NeurIPS*, 2018.

[48] Marinka Zitnik, Rok Sosic, Marcus W. Feldman, and Jure Leskovec. Evolution of resilience in protein interactomes across the tree of life. *Proceedings of the National Academy of Sciences*.

A Review of Differential Geometry

We first recall some definitions of differential and hyperbolic geometry.

A.1 Differential geometry

Manifold. An $d-$dimensional *manifold* $\mathcal{M}$ is a topological space that locally resembles the topological space $\mathbb{R}^d$ near each point. More concretely, for each point $\mathbf{x}$ on $\mathcal{M}$, we can find a *homeomorphism* (continuous bijection with continuous inverse) between a neighbourhood of $\mathbf{x}$ and $\mathbb{R}^d$. The notion of manifold is a generalization of surfaces in high dimensions.
Tangent space. Intuitively, if we think of $\mathcal{M}$ as a $d-$dimensional manifold embedded in $\mathbb{R}^{d+1}$, the *tangent space* $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ at point $\mathbf{x}$ on $\mathcal{M}$ is a $d-$dimensional hyperplane in $\mathbb{R}^{d+1}$ that best approximates $\mathcal{M}$ around $\mathbf{x}$. Another possible interpretation for $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ is that it contains all the possible directions of curves on $\mathcal{M}$ passing through $\mathbf{x}$. The elements of $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ are called *tangent vectors* and the union of all tangent spaces is called the *tangent bundle* $\mathcal{T}\mathcal{M} = \cup_{\mathbf{x}\in\mathcal{M}}\mathcal{T}_{\mathbf{x}}\mathcal{M}$.
Riemannian manifold. A *Riemannian manifold* is a pair $(\mathcal{M}, \mathbf{g})$, where $\mathcal{M}$ is a smooth manifold and $\mathbf{g} = (g_{\mathbf{x}})_{\mathbf{x}\in\mathcal{M}}$ is a *Riemannian metric*, that is a family of smoothly varying inner products on tangent spaces, $g_{\mathbf{x}} : \mathcal{T}_{\mathbf{x}}\mathcal{M} \times \mathcal{T}_{\mathbf{x}}\mathcal{M} \to \mathbb{R}$. Riemannian metrics can be used to measure distances on manifolds.
Distances and geodesics. Let $(\mathcal{M}, \mathbf{g})$ be a Riemannian manifold. For $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$, define the norm of $\mathbf{v}$ by $||\mathbf{v}||_{\mathbf{g}} \coloneqq \sqrt{g_{\mathbf{x}}(\mathbf{v}, \mathbf{v})}$. Suppose $\gamma : [a, b] \to \mathcal{M}$ is a smooth curve on $\mathcal{M}$. Define the length of γ by

$$L(\gamma) \coloneqq \int_a^b ||\gamma'(t)||_{\mathbf{g}} dt.$$

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

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

Geodesic distances are a generalization of straight lines (or shortest paths) to non-Euclidean geometry. A curve $\gamma : [a, b] \to \mathcal{M}$ is *geodesic* if $d(\gamma(t), \gamma(s)) = L(\gamma|_{[t,s]})\forall(t, s) \in [a, b](t < s)$.
Parallel transport. *Parallel transport* is a generalization of translation to non-Euclidean geometry. Given a smooth manifold $\mathcal{M}$, parallel transport $P_{\mathbf{x}\to\mathbf{y}}(\cdot)$ maps a vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathcal{M}$ to $P_{\mathbf{x}\to\mathbf{y}}(\mathbf{v}) \in \mathcal{T}_{\mathbf{y}}\mathcal{M}$. In Riemannian geometry, parallel transport preserves the Riemannian metric tensor (norm, inner products...).
Curvature. At a high level, curvature measures how much a geometric object such as surfaces deviate from a flat plane. For instance, the Euclidean space has zero curvature while spheres have positive curvature. We illustrate the concept of curvature in Figure 6.

A.2 Hyperbolic geometry

Hyperbolic space. The hyperbolic space in d dimensions is the unique complete, simply connected $d-$dimensional Riemannian manifold with constant negative sectional curvature. There exist several models of hyperbolic space such as the Poincaré model or the hyperboloid model (also known as the Minkowski model or the Lorentz model). In what follows, we review the Poincaré and the hyperboloid models of hyperbolic space as well as connections between these two models.

A.2.1 Poincaré ball model

Let $||.||_2$ be the Euclidean norm. The Poincaré ball model with unit radius and constant negative curvature -1 in d dimensions is the Riemannian manifold $(\mathbb{D}^{d,1}, (g_{\mathbf{x}})_{\mathbf{x}})$ where

$$\mathbb{D}^{d,1} \coloneqq \{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x}||^2 < 1\},$$

and

$$g_{\mathbf{x}} = \lambda_{\mathbf{x}}^2 I_d,$$

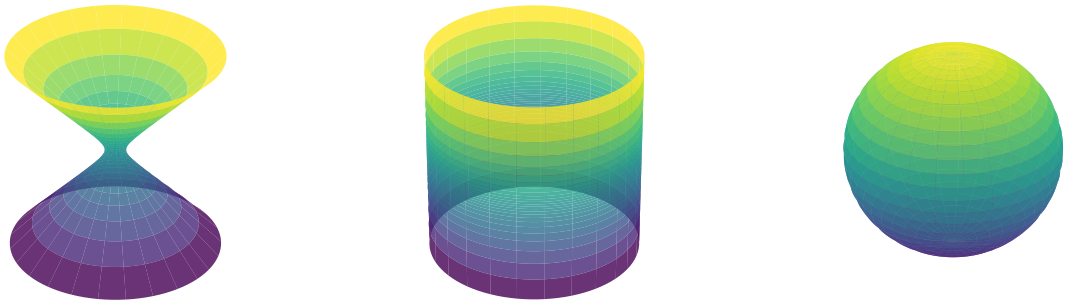


Figure 6: From left to right: a surface of negative curvature, a surface of zero curvature, and a surface of positive curvature.

where $\lambda_{\mathbf{x}} \coloneqq \frac{2}{1-||\mathbf{x}||_2^2}$ and I_d is the identity matrix. The induced distance between two points $(\mathbf{x}, \mathbf{y})$ in $\mathbb{D}^{d,1}$ can be computed as

$$d_{\mathbb{D}}^1(\mathbf{x}, \mathbf{y}) = \operatorname{arcosh}\left(1 + 2\frac{||\mathbf{x}-\mathbf{y}||_2^2}{(1-||\mathbf{x}||_2^2)(1-||\mathbf{y}||_2^2)}\right).$$

A.2.2 Hyperboloid model

Hyperboloid model. Let $\langle ., . \rangle_{\mathcal{L}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$ denote the Minkowski inner product,

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \coloneqq -x_0 y_0 + x_1 y_1 + \ldots + x_d y_d.$$

The hyperboloid model with unit imaginary radius and constant negative curvature -1 in d dimensions is defined as the Riemannian manifold $(\mathbb{H}^{d,1}, (g_{\mathbf{x}})_{\mathbf{x}})$ where

$$\mathbb{H}^{d,1} \coloneqq \{\mathbf{x} \in \mathbb{R}^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, x_0 > 0\},$$

and

$$g_{\mathbf{x}} \coloneqq \begin{bmatrix} -1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}.$$

The induced distance between two points $(\mathbf{x}, \mathbf{y})$ in $\mathbb{H}^{d,1}$ can be computed as

$$d_{\mathcal{L}}^1(\mathbf{x}, \mathbf{y}) = \operatorname{arcosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}).$$

Geodesics. We recall a result that gives the unit speed geodesics in the hyperboloid model with curvature -1 [31]. This result allow us to derive Corollary 3.1 and 3.2 for the hyperboloid manifold with negative curvature $-1/K$, and then learn K as a model parameter in HYPERGCN.

Theorem A.1. *Let $\mathbf{x} \in \mathbb{H}^{d,1}$ and $\mathbf{u} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,1}$ unit-speed (i.e. $\langle \mathbf{u}, \mathbf{u} \rangle_{\mathcal{L}} = 1$). The unique unit-speed geodesic $\gamma_{\mathbf{x} \to \mathbf{u}} : [0, 1] \to \mathbb{H}^{d,1}$ such that $\gamma_{\mathbf{x} \to \mathbf{u}}(0) = \mathbf{x}$ and $\dot{\gamma}_{\mathbf{x} \to \mathbf{u}}(0) = \mathbf{u}$ is given by*

$$\gamma_{\mathbf{x} \to \mathbf{u}}(t) = \cosh(t)\mathbf{x} + \sinh(t)\mathbf{u}.$$

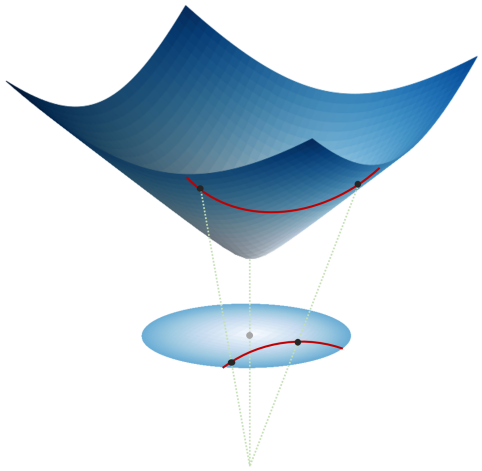


Figure 7: Illustration of the hyperboloid model (top) in 3 dimensions and its connection to the Poincaré disk (bottom).

Parallel Transport. If two points $\mathbf{x}$ and $\mathbf{y}$ on the hyperboloid $\mathbb{H}^{d,1}$ are connected by a geodesic, then the parallel transport of a tangent vector $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,1}$ to the tangent space $\mathcal{T}_{\mathbf{y}}\mathbb{H}^{d,1}$ is

$$P_{\mathbf{x}\to\mathbf{y}}(\mathbf{v}) = \mathbf{v} - \frac{\langle \log_{\mathbf{x}}(\mathbf{y}), \mathbf{v}\rangle_{\mathcal{L}}}{d^1_{\mathcal{L}}(\mathbf{x},\mathbf{y})^2}(\log_{\mathbf{x}}(\mathbf{y}) + \log_{\mathbf{y}}(\mathbf{x})). \tag{15}$$

Projections. Finally, we recall projections to the hyperboloid manifold and its corresponding tangent spaces. A point $\mathbf{x} = (x_0, \mathbf{x}_{1:d}) \in \mathbb{R}^{d+1}$ can be projected on the hyperboloid manifold $\mathbb{H}^{d,1}$ with

$$\Pi_{\mathbb{R}^{d+1}\to\mathbb{H}^{d,1}}(\mathbf{x}) \coloneqq (\sqrt{1 + ||\mathbf{x}_{1:d}||_2^2}, \mathbf{x}_{1:d}). \tag{16}$$

Similarly, a point $\mathbf{v} \in \mathbb{R}^{d+1}$ can be projected on $\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,1}$ with

$$\Pi_{\mathbb{R}^{d+1}\to\mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,1}}(\mathbf{v}) \coloneqq \mathbf{v} + \langle \mathbf{x}, \mathbf{v}\rangle_{\mathcal{L}}\mathbf{x}. \tag{17}$$

In practice, these projections are very useful for optimization purposes as they constrain embeddings and tangent vectors to remain on the manifold and tangent spaces.

A.2.3 Connection between the Poincaré ball model and the hyperboloid model

While the hyperboloid model tends to be more stable for optimization than the Poincaré model [27], the Poincaré model is very interpretable and embeddings can be directly visualized on the Poincaré disk. Fortunately, these two models are isomorphic (*cf.* Figure 7) and there exist a diffeomorphism $\Pi_{\mathbb{H}^{d,1}\to\mathbb{D}^{d,1}}(\cdot)$ mapping one space onto the other

$$\Pi_{\mathbb{H}^{d,1}\to\mathbb{D}^{d,1}}(x_0, \dots, x_d) = \frac{(x_1, \dots, x_d)}{x_0 + 1} \tag{18}$$

$$\text{and} \quad \Pi_{\mathbb{D}^{d,1}\to\mathbb{H}^{d,1}}(x_1, \dots, x_d) = \frac{(1 + ||\mathbf{x}||_2^2, 2x_1, \dots, 2x_d)}{1 - ||\mathbf{x}||_2^2}. \tag{19}$$

B Proofs of Results

B.1 Hyperboloid model of hyperbolic space

For completeness, we re-derive results of hyperbolic geometry for any arbitrary curvature. Similar derivations can be found in the literature [41].

Proposition 3.1. *Let $\mathbf{x} \in \mathbb{H}^{d,K}$, $\mathbf{u} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ be unit-speed. The unique unit-speed geodesic $\gamma_{\mathbf{x}\to\mathbf{u}}(\cdot)$ such that $\gamma_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{x}$, $\dot{\gamma}_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{u}$ is $\gamma^K_{\mathbf{x}\to\mathbf{u}}(t) = \cosh\left(\frac{t}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{t}{\sqrt{K}}\right)\mathbf{u}$, and the intrinsic distance function between two points $\mathbf{x}, \mathbf{y}$ in $\mathbb{H}^{d,K}$ is then*

$$d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \sqrt{K}\text{arcosh}(-\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}/K). \tag{4}$$

Proof. We know that the unique unit-speed geodesic $\gamma^K_{\mathbf{x}\to\mathbf{u}}(.)$ in $\mathbb{H}^{d,K}$ must satisfy

$$\gamma^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{x} \text{ and } \dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{u} \text{ and } \frac{d}{dt}\langle \dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(t), \dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(t)\rangle_{\mathcal{L}} = 0 \; \forall t. \tag{20}$$

Let $\gamma^K_{\mathbf{x}\to\mathbf{u}}(t) = \cosh(\frac{t}{\sqrt{K}})\mathbf{x} + \sqrt{K}\sinh(\frac{t}{\sqrt{K}})\mathbf{u}$. We have $\gamma^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{x}$ and $\dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{u}$. Furthermore, since $\mathbf{u} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$, we have $\langle \mathbf{u}, \mathbf{x}\rangle_{\mathcal{L}} = 0$ and for all t:

$$\begin{aligned}
\langle \gamma^K_{\mathbf{x}\to\mathbf{u}}(t), \gamma^K_{\mathbf{x}\to\mathbf{u}}(t)\rangle_{\mathcal{L}} &= \cosh^2(\frac{t}{\sqrt{K}})\langle \mathbf{x}, \mathbf{x}\rangle_{\mathcal{L}} + K\sinh^2(\frac{t}{\sqrt{K}})\langle \mathbf{u}, \mathbf{u}\rangle_{\mathcal{L}} \\
&= -K\cosh^2(\frac{t}{\sqrt{K}}) + K\sinh^2(\frac{t}{\sqrt{K}}) \\
&= -K.
\end{aligned}$$

Therefore, $\gamma^K_{\mathbf{x}\to\mathbf{u}}(\cdot)$ is a curve on $\mathbb{H}^{d,K}$. Furthermore, we have $\dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(t) = \frac{1}{\sqrt{K}}\sinh(\frac{t}{\sqrt{K}})\mathbf{x} + \cosh(\frac{t}{\sqrt{K}})\mathbf{u}$ and therefore

$$\begin{aligned}
\frac{d}{dt}\langle \dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(t), \dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(t)\rangle_{\mathcal{L}} &= \frac{d}{dt}(\frac{1}{K}\sinh^2(\frac{t}{\sqrt{K}})\langle \mathbf{x}, \mathbf{x}\rangle_{\mathcal{L}} + \cosh^2(\frac{t}{\sqrt{K}})\langle \mathbf{u}, \mathbf{u}\rangle_{\mathcal{L}}) \\
&= \frac{d}{dt}(-\sinh^2(\frac{t}{\sqrt{K}}) + \cosh^2(\frac{t}{\sqrt{K}})) \\
&= 0.
\end{aligned}$$

Finally, $\gamma^K_{\mathbf{x}\to\mathbf{u}}(.)$ verifies all the conditions in Equation 20 and is therefore the unique unit-speed geodesic on $\mathbb{H}^{d,K}$ such that $\gamma^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{x}$ and $\dot{\gamma}^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{u}$.

□

Proposition 3.2. *For $\mathbf{x} \in \mathbb{H}^{d,K}$, $\mathbf{v} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$ and $\mathbf{y} \in \mathbb{H}^{d,K}$ such that $\mathbf{v} \neq \mathbf{0}$ and $\mathbf{y} \neq \mathbf{x}$, the exponential and logarithmic maps of the hyperboloid model are given by*

$$\exp^K_{\mathbf{x}}(\mathbf{v}) = \cosh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}, \;\; \log^K_{\mathbf{x}}(\mathbf{y}) = d^K_{\mathcal{L}}(\mathbf{x}, \mathbf{y})\frac{\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}}{||\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}}}.$$

Proof. We use a similar reasoning to that in Corollary 1.1 in [11]. Let $\gamma^K_{\mathbf{x}\to\mathbf{v}}(.)$ be the unique geodesic such that $\gamma^K_{\mathbf{x}\to\mathbf{v}}(0) = \mathbf{x}$ and $\dot{\gamma}^K_{\mathbf{x}\to\mathbf{v}}(0) = \mathbf{v}$. Let us define $\mathbf{u} := \frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}$ where $||\mathbf{v}||_{\mathcal{L}} = \sqrt{\langle \mathbf{v}, \mathbf{v}\rangle_{\mathcal{L}}}$ is the Minkowski norm of $\mathbf{v}$ and

$$\phi^K_{\mathbf{x}\to\mathbf{u}}(t) := \gamma^K_{\mathbf{x}\to\mathbf{v}}\left(\frac{t}{||\mathbf{v}||_{\mathcal{L}}}\right).$$

$\phi_{\mathbf{x}\to\mathbf{u}}(t)$ satisfies,

$$\phi^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{x} \text{ and } \dot{\phi}^K_{\mathbf{x}\to\mathbf{u}}(0) = \mathbf{u} \text{ and } \frac{d}{dt}\langle \dot{\phi}^K_{\mathbf{x}\to\mathbf{u}}(t), \dot{\phi}^K_{\mathbf{x}\to\mathbf{u}}(t)\rangle_{\mathcal{L}} = 0 \; \forall t.$$

Therefore $\phi^K_{\mathbf{x}\to\mathbf{u}}(.)$ is a unit-speed geodesic in $\mathbb{H}^{d,K}$ and we get

$$\phi^K_{\mathbf{x}\to\mathbf{u}}(t) = \cosh(\frac{t}{\sqrt{K}})\mathbf{x} + \sqrt{K}\sinh(\frac{t}{\sqrt{K}})\mathbf{u}.$$

By identification, this leads to

$$\gamma^K_{\mathbf{x}\to\mathbf{v}}(t) = \cosh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}t\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}t\right)\frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}.$$

We can use this result to derive exponential and logarthimic maps on the hyperboloid model. We know that $\exp^K_{\mathbf{x}}(\mathbf{v}) = \gamma^K_{\mathbf{x}\to\mathbf{v}}(1)$. Therefore we get,

$$\exp^K_{\mathbf{x}}(\mathbf{v}) = \cosh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}.$$

Now let $\mathbf{y} = \exp^K_{\mathbf{x}}(\mathbf{v})$. We have $\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}} = -K\cosh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)$ as $\langle\mathbf{x},\mathbf{x}\rangle_{\mathcal{L}} = -K$ and $\langle\mathbf{x},\mathbf{v}\rangle_{\mathcal{L}} = 0$. Therefore $\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x} = \sqrt{K}\sinh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}$ and we get

$$\mathbf{v} = \sqrt{K}\operatorname{arsinh}\left(\frac{||\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}}{||\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}}},$$

where $||\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}||_{\mathcal{L}}$ is well defined since $\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x} \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^{d,K}$. Note that,

$$\begin{aligned}
||\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}} &= \sqrt{\langle\mathbf{y},\mathbf{y}\rangle_{\mathcal{L}} + \frac{2}{K}\langle\mathbf{x},\mathbf{y}\rangle^2_{\mathcal{L}} + \frac{1}{K^2}\langle\mathbf{x},\mathbf{y}\rangle^2_{\mathcal{L}}\langle\mathbf{x},\mathbf{x}\rangle_{\mathcal{L}}} \\
&= \sqrt{-K + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle^2_{\mathcal{L}}} \\
&= \sqrt{K}\sqrt{\langle\frac{\mathbf{x}}{\sqrt{K}},\frac{\mathbf{y}}{\sqrt{K}}\rangle^2_{\mathcal{L}} - 1} \\
&= \sqrt{K}\sinh\operatorname{arcosh}\Big(-\langle\frac{\mathbf{x}}{\sqrt{K}},\frac{\mathbf{y}}{\sqrt{K}}\rangle_{\mathcal{L}}\Big)
\end{aligned}$$

as $\langle\frac{\mathbf{x}}{\sqrt{K}},\frac{\mathbf{y}}{\sqrt{K}}\rangle_{\mathcal{L}} \leq -1$. Therefore, we finally have

$$\log^K_{\mathbf{x}}(\mathbf{y}) = \sqrt{K}\operatorname{arcosh}\Big(-\langle\frac{\mathbf{x}}{\sqrt{K}},\frac{\mathbf{y}}{\sqrt{K}}\rangle_{\mathcal{L}}\Big)\frac{\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}}{||\mathbf{y} + \frac{1}{K}\langle\mathbf{x},\mathbf{y}\rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}}}.$$

□

B.2 Curvature

Lemma 1. *For any hyperbolic spaces with constant curvatures $-1/K, -1/K' > 0$, and any pair of hyperbolic points $(\mathbf{u},\mathbf{v})$ embedded in $\mathbb{H}^{d,K}$, there exists a mapping $\phi : \mathbb{H}^{d,K} \to \mathbb{H}^{d,K'}$ to another pair of corresponding hyperbolic points in $\mathbb{H}^{d,K'}$, $(\phi(\mathbf{u}),\phi(\mathbf{v}))$ such that the Minkowski inner product is scaled by a constant factor.*

Proof. For any hyperbolic embedding $\mathbf{x} = (x_0, x_1, \ldots, x_d) \in \mathbb{H}^{d,K}$ we have the identity: $\langle\mathbf{x},\mathbf{x}\rangle_{\mathcal{L}} = -x_0^2 + \sum_{i=1}^d x_i^2 = -K$. For any hyperbolic curvature $-1/K < 0$, consider the mapping $\phi(\mathbf{x}) = \sqrt{\frac{K'}{K}}\mathbf{x}$. Then we have the identity $\langle\phi(\mathbf{x}),\phi(\mathbf{x})\rangle_{\mathcal{L}} = -K'$ and therefore $\phi(\mathbf{x}) \in \mathbb{H}^{d,K'}$. For any pair $(\mathbf{u},\mathbf{v})$, $\langle\phi(\mathbf{u}),\phi(\mathbf{v})\rangle_{\mathcal{L}} = \frac{K'}{K}\left(-\mathbf{u}_0\mathbf{v}_0 + \sum_{i=1}^d \mathbf{u}_i\mathbf{v}_i\right) = \frac{K'}{K}\langle\mathbf{u},\mathbf{v}\rangle_{\mathcal{L}}$. The factor $\frac{K'}{K}$ only depends on curvature, but not the specific embeddings. □

Lemma 1 implies that given a set of embeddings learned in hyperbolic space $\mathbb{H}^{d,K}$, we can find embeddings in another hyperbolic space with different curvature, $\mathbb{H}^{d,K'}$, such that the Minkowski inner products for all pairs of embeddings are scaled by the same factor $\frac{K'}{K}$.

For link prediction tasks, Theorem 4.1 shows that with infinite precision, the expressive power of hyperbolic spaces with varying curvatures is the same.

Name	Nodes	Edges	Classes	Node features
CORA	2708	5429	7	1433
PUBMED	19717	88651	3	500
HUMAN PPI	17598	5429	4	17
AIRPORT	3188	18631	4	4
DISEASE	1044	1043	2	1000
DISEASE-M	43193	43102	2	1000

Table 3: Benchmarks' statistics

Theorem 4.1. *For any hyperbolic curvatures $-1/K, -1/K' < 0$, for any node embeddings $H = \{\mathbf{h}_i\} \subset \mathbb{H}^{d,K}$ of a graph G, we can find $H' \subset \mathbb{H}^{d,K'}$, $H' = \{\mathbf{h}'_i | \mathbf{h}'_i = \sqrt{\frac{K'}{K}}\mathbf{h}_i\}$, such that the reconstructed graph from H' via the Fermi-Dirac decoder is the same as the reconstructed graph from H, with different decoder parameters (r, t) and (r', t').*

Proof. The Fermi-Dirac decoder predicts that there exists a link between node i and j *iif* $\left[e^{(d^K_{\mathcal{L}}(\mathbf{h}_i,\mathbf{h}_j)-r)/t}+1\right]^{-1} \geq b$, where $b \in (0,1)$ is the threshold for determining existence of links. The criterion is equivalent to $d^K_{\mathcal{L}}(\mathbf{h}_i, \mathbf{h}_j) \leq r + t\log(\frac{1-b}{b})$.

Given $H = \{\mathbf{h_1}, \ldots, \mathbf{h_n}\}$, the graph G_H reconstructed with the Fermi-Dirac decoder has the edge set $E_H = \left\{(i,j) | d^K_{\mathcal{L}}(\mathbf{h}_i, \mathbf{h}_j) \leq r + t\log(\frac{1-b}{b})\right\}$. Consider the mapping to $\mathbb{H}^{d,K'}$, $\phi(\mathbf{x}) := \sqrt{\frac{K'}{K}}\mathbf{x}$. Let $H' = \{\phi(\mathbf{h_1}), \ldots, \phi(\mathbf{h_n})\}$. By Lemma 1,

$$d^{K'}_{\mathcal{L}}(\phi(\mathbf{h}_i), \phi(\mathbf{h}_j)) = \sqrt{K'}\text{arcosh}\left(-\frac{K'}{K}\langle \mathbf{h}_i, \mathbf{h}_j\rangle_{\mathcal{L}}/K'\right) = \sqrt{\frac{K'}{K}}d^K_{\mathcal{L}}(\mathbf{h}_i, \mathbf{h}_j). \tag{21}$$

Due to linearity, we can find decoder parameter, r' and t' that satisfy $r' + t'\log(\frac{1-b}{b}) = \sqrt{\frac{K'}{K}}(r + t\log(\frac{1-b}{b}))$. With such r', t', the criterion $d^K_{\mathcal{L}}(\mathbf{h}_i, \mathbf{h}_j) \leq r + t\log(\frac{1-b}{b})$ is equivalent to $d^{K'}_{\mathcal{L}}(\phi(\mathbf{h}_i), \phi(\mathbf{h}_j)) \leq r' + t'\log(\frac{1-b}{b})$. Therefore, the reconstructed graph $G_{H'}$ based on the set of embeddings H' is identical to G_H. □

C Experimental Details

C.1 Dataset statistics

We detail the dataset statistics in Table 3. We will release datasets download links at https://github.com/ines-chami/hypergcn.

C.2 Training details

Here we present details of HYPERGCN's training pipeline, with optimization and incorporation of DropConnect [40].
Parameter optimization. Recall that linear transformations and attention are defined on the tangent space of points. Therefore the linear layer and attention parameters are Euclidean. For bias, there are two options: one can either define parameters in hyperbolic space, and use hyperbolic addition operation [10], or define parameters in Euclidean space, and use Euclidean addition after transforming the points into the tangent space. Through experiments we find that Euclidean optimization is much more stable, and gives slightly better test performance compared to Riemannian optimization, if we define parameters such as bias in hyperbolic space. Hence different from shallow hyperbolic embeddings, although our model and embeddings are hyperbolic, the learnable graph convolution parameters can be optimized via Euclidean optimization (Adam Optimizer [18]), thanks to exponential and logarithmic maps. Note that to train shallow Poincaré embeddings, we use Riemannian Stochastic Gradient Descent [4, 46], since its model parameters are hyperbolic. We use early stopping based on validation set performance with a patience of 100 epochs.

Drop connection. Since rescaling vectors in hyperbolic space requires exponential and logarithmic maps, and is conceptually not tied to the inverse dropout rate in terms of re-normalizing L1 norm, Dropout cannot be directly applied in HYPERGCN. However, as a result of using Euclidean parameters in HYPERGCN, DropConnect [40], the generalization of Dropout, can be used as a regularization. DropConnect randomly zeros out the neural network connections, *i.e.* elements of the Euclidean parameters during training time, improving the generalization of HYPERGCN.
Projections. Finally, we apply projections similar to Equations 16 and 17 for the hyperboloid model $\mathbb{H}^{d,K}$ after each feature transform and $\log$ or $\exp$ map, to constrain embeddings and tangent vectors to remain on the manifold and tangent spaces.