Hyperbolic Geometry & Neural Networks

Riley Guyett

July 28, 2025

Outline

1. Overview of the Problem
2. Hyperbolic Geometry
3. Survey of Results
4. Conclusions

Table of Contents

Hierarchical Data Structures

Many real-world networks we would like to train with have an underlying tree-like structure.

- Social networks, taxonomies, knowledge graphs, financial networks, single-cell RNA sequencing, etc.

Hierarchical Data Structures

Many real-world networks we would like to train with have an underlying tree-like structure.

- Social networks, taxonomies, knowledge graphs, financial networks, single-cell RNA sequencing, etc.
- These networks are usually quite large

Hierarchical Data Structures In Euclidean Space

Trees exist without a larger ambient space. But to visualize and analyze these graphs, we would like to embed them in some Euclidean space.

Hierarchical Data Structures In Euclidean Space

Trees exist without a larger ambient space. But to visualize and analyze these graphs, we would like to embed them in some Euclidean space. Unfortunately there are some immediate issues with this embedding.

- To visualize the graph, we would like to embed in a low-dimensional space (2-3 dimensions)
- For better analysis we would like to embed in a higher-dimensional space (hundreds of dimensions)

Hierarchical Data Structures In Euclidean Space

Trees exist without a larger ambient space. But to visualize and analyze these graphs, we would like to embed them in some Euclidean space. Unfortunately there are some immediate issues with this embedding.

- To visualize the graph, we would like to embed in a low-dimensional space (2-3 dimensions)
- For better analysis we would like to embed in a higher-dimensional space (hundreds of dimensions)
- Distances in the graph are not accurately reflected in their embedded Euclidean distance (high distortion) regradless of dimension
- Since distance is usally a measure of similarity, that means that similarity is being distorted in our analysis/visualizations

Observed Networks v.s. Random Networks

The networks that we would like to study/train on have notable differences from random networks of the same size:

- Real networks are usually highly clustered (neighbors of a random node are very likely to be neighbors of each other)
- Further, the amount of clustering is independent of the number of nodes

Observed Networks v.s. Random Networks

The networks that we would like to study/train on have notable differences from random networks of the same size:

- Real networks are usually highly clustered (neighbors of a random node are very likely to be neighbors of each other)
- Further, the amount of clustering is independent of the number of nodes
- Real networks are usually scale-free (the probability that a node has k links is approximately $k^{-\gamma}$, where γ is usually between 2 and 3)

Observed Networks v.s. Random Networks

The networks that we would like to study/train on have notable differences from random networks of the same size:

- Real networks are usually highly clustered (neighbors of a random node are very likely to be neighbors of each other)
- Further, the amount of clustering is independent of the number of nodes
- Real networks are usually scale-free (the probability that a node has k links is approximately $k^{-\gamma}$, where γ is usually between 2 and 3)

Most models to describe the topology of complex networks have difficulty capturing these observations (either individually or together).
So for best results, we need to use models that are optimal for scale-free, clustered graphs, rather than arbitrary graphs.

Why Signs Point to Hyperbolic Geometry

Why might hyperbolic geometry help with these issues?

Why Signs Point to Hyperbolic Geometry

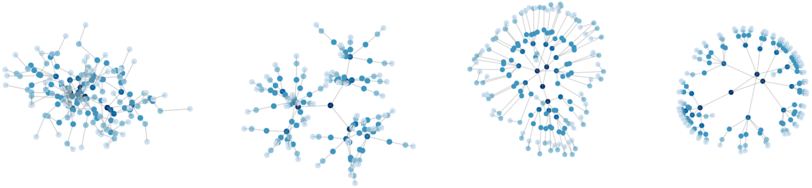
Why might hyperbolic geometry help with these issues?

- We can embed trees into hyperbolic space with arbitrarily small distortion
- This lack of distortion even holds in 2-dimenionsal hyperbolic space, so we may be able to do similar analysis with fewer dimensions
- We can naturally observe hierarchies in the data

Why Signs Point to Hyperbolic Geometry

Why might hyperbolic geometry help with these issues?

- We can embed trees into hyperbolic space with arbitrarily small distortion
- This lack of distortion even holds in 2-dimenionsal hyperbolic space, so we may be able to do similar analysis with fewer dimensions
- We can naturally observe hierarchies in the data



(Left two) first and last layers of a graph convolutional network and (right two) first and last layers of a hyperbolic graph convolutional network. Depth indicated by color.

Table of Contents

What is Hyperbolic Geometry?

Definition-Hyperbolic Space

Hyperbolic space is the unique, complete, simply connected Riemannian manifold with constant negative sectional curvature.

In contrast, Euclidean geometry has constant zero sectional curvature.

What is Hyperbolic Geometry?

Definition-Hyperbolic Space

Hyperbolic space is the unique, complete, simply connected Riemannian manifold with constant negative sectional curvature.

In contrast, Euclidean geometry has constant zero sectional curvature. There are many equivalent models for hyperbolic space, but we will focus one two for the puporses of this talk: the Poincaré model and the Lorentz model.

Poincaré Model

Definition-Poincaré Model

The Poincaré model is the Riemannian manifold $\mathcal{P}^n = (\mathcal{B}^n, g_p)$, where $\mathcal{B}^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$ is the open unit Euclidean n-ball and

$$g_p(\mathbf{x}) = \left(\frac{2}{1 - \|\mathbf{x}\|^2}\right)^2 g_e.$$

Poincaré Model

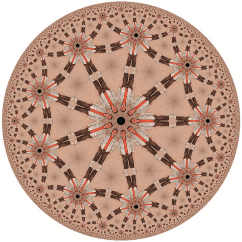
Definition-Poincaré Model

The Poincaré model is the Riemannian manifold $\mathcal{P}^n = (\mathcal{B}^n, g_p)$, where $\mathcal{B}^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$ is the open unit Euclidean n-ball and

$$g_p(\mathbf{x}) = \left(\frac{2}{1 - \|\mathbf{x}\|^2}\right)^2 g_e.$$

The distance function on $\mathcal{P}^n$ is

$$d_p(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}\left(1 + 2\frac{\|\mathbf{x} - \mathbf{y}\|^2}{\left(1 - \|\mathbf{x}\|^2\right)\left(1 - \|\mathbf{y}\|^2\right)}\right)$$



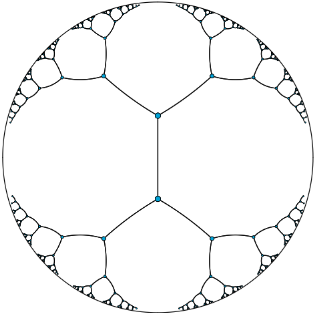
Visualizing Trees in the Poincaré Model

A regular tree with branching factor b has $(b+1)\,b^{\ell-1}$ nodes at level ℓ and $\dfrac{(b+1)\,b^{\ell}-2}{b-1}$ nodes on a level $\leq \ell$. So, the number of children grows exponentially with their distance to the root of the tree.

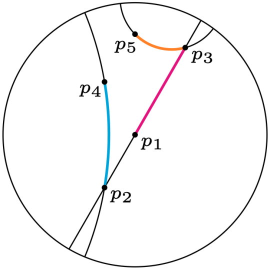
Visualizing Trees in the Poincaré Model

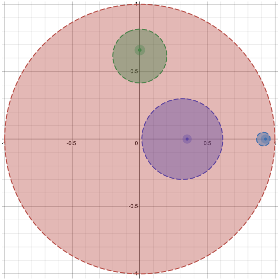
A regular tree with branching factor b has $(b+1)\,b^{\ell-1}$ nodes at level ℓ and $\dfrac{(b+1)\,b^{\ell}-2}{b-1}$ nodes on a level $\leq \ell$. So, the number of children grows exponentially with their distance to the root of the tree.

In the Poincaré model, distance also scales exponentially. So, putting nodes ℓ levels below the root on the hyperbolic sphere of radius $r \propto \ell$ produces a viewable embedding that preserves distances.

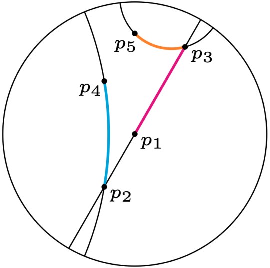


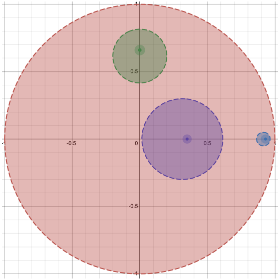
Distance in Poincaré model





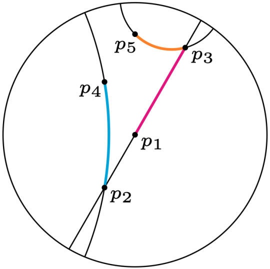
The lines are geodesics in the Poincaré model. Circles are Euclidean circles but with a different center and radius (shifted towards the origin).

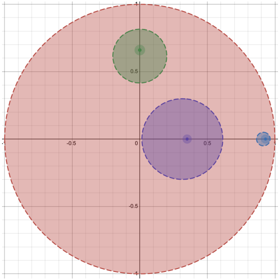




The lines are geodesics in the Poincaré model. Circles are Euclidean circles but with a different center and radius (shifted towards the origin). In the bottom figure, all circles are the same size. So for **x** to be close to a point **y**, the points could either be close in the Euclidean sense or one could be closer to the origin.

Distance in Poincaré model





The lines are geodesics in the Poincaré model. Circles are Euclidean circles but with a different center and radius (shifted towards the origin).

In the bottom figure, all circles are the same size. So for **x** to be close to a point **y**, the points could either be close in the Euclidean sense or one could be closer to the origin.

So to be close to a lot of points, **x** must be close to the origin.

Lorentz Model

Definition-Lorentz Model

For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n+1}$, let $\langle \mathbf{x}, \mathbf{y} \rangle_L = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i$ denote the *Lorentz scalar product*. Then the Lorentz model is $\mathcal{L}^n = (\mathcal{H}^n, g_L)$, where $\mathcal{H}^n = \left\{ \mathbf{x} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x} \rangle_L = -1, x_0 > 0 \right\}$ and $g_L(\mathbf{x}) = \operatorname{diag}(-1, 1, \ldots, 1)$.

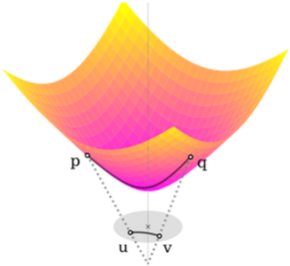
Lorentz Model

Definition-Lorentz Model

For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n+1}$, let $\langle \mathbf{x}, \mathbf{y} \rangle_L = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i$ denote the *Lorentz scalar product*. Then the Lorentz model is $\mathcal{L}^n = (\mathcal{H}^n, g_L)$, where $\mathcal{H}^n = \left\{ \mathbf{x} \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x} \rangle_L = -1, x_0 > 0 \right\}$ and $g_L(\mathbf{x}) = \mathrm{diag}(-1, 1, \ldots, 1)$.

The distance function on $\mathcal{L}^n$ is

$$d_L(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}\left(-\langle \mathbf{x}, \mathbf{y} \rangle_L\right)$$



Hyperbolic Optimzation

In optimization on Riemannian manifolds, we would like to solve problems of the form $\min_{\theta \in \mathcal{M}} f(\theta)$ for some smooth function $f : \mathcal{M} \to \mathbb{R}$ over parameters $\theta \in \mathcal{M}$. We do so using the following algorithm.

Algorithm 1 Riemannian Stochastic Gradient Descent (RSGD)

Input Learning rate η, number of epochs T.

for $t = 1, \dots, T$ **do**
    $\mathbf{h}_t \leftarrow g_{\theta_t}^{-1} \nabla f(\theta_t)$
    $\operatorname{grad} f(\theta_t) \leftarrow \operatorname{proj}_{\theta_t}(\mathbf{h}_t)$
    $\theta_{t+1} \leftarrow \exp_{\theta_t}(-\eta \operatorname{grad} f(\theta_t))$
end for

Hyperbolic Optimzation

In optimization on Riemannian manifolds, we would like to solve problems of the form $\min_{\theta \in \mathcal{M}} f(\theta)$ for some smooth function $f : \mathcal{M} \to \mathbb{R}$ over parameters $\theta \in \mathcal{M}$. We do so using the following algorithm.

Algorithm 2 Riemannian Stochastic Gradient Descent (RSGD)

Input Learning rate η, number of epochs T.

for $t = 1, \ldots, T$ **do**
  $\mathbf{h}_t \leftarrow g_{\theta_t}^{-1} \nabla f(\theta_t)$
  $\operatorname{grad} f(\theta_t) \leftarrow \operatorname{proj}_{\theta_t}(\mathbf{h}_t)$
  $\theta_{t+1} \leftarrow \exp_{\theta_t}(-\eta \operatorname{grad} f(\theta_t))$
end for

For the Lorentz model, g_{θ_t} is trivial to invert, $\operatorname{proj}_{\theta_t}(\mathbf{h}_t) = \mathbf{h}_t + \langle \theta_t, \mathbf{h}_t \rangle_L \theta_t$, and $\exp_{\theta_t}(\mathbf{v}) = \cosh\left(\|\mathbf{v}\|_L\right) \theta_t + \sinh\left(\|\mathbf{v}\|_L\right) \frac{\mathbf{v}}{\|\mathbf{v}_L\|}$.

Immediate Benefits and Downsides of Hyperbolic Space

Benefits

- Poincare model is very intuitive for visualization and interpreting embeddings
- Lorentz model is well-suited for Riemannian optimization

Immediate Benefits and Downsides of Hyperbolic Space

Benefits

- Poincare model is very intuitive for visualization and interpreting embeddings
- Lorentz model is well-suited for Riemannian optimization

Downsides

- Potential for loss in exponential map

Equivalence of Hyperbolic Spaces

Using the diffeomorphisms

$$\pi\colon \mathcal{L}^n \to \mathcal{P}^n, \qquad (x_0, x_1, \ldots, x_n) \mapsto \frac{(x_1, \ldots, x_n)}{x_0 + 1}$$

$$\pi^{-1}\colon \mathcal{P}^n \to \mathcal{L}^n, \qquad (x_1, \ldots, x_n) \mapsto \frac{\left(1 + \|\mathbf{x}\|^2, 2x_1, \ldots, 2x_n\right)}{1 - \|\mathbf{x}\|^2}$$

we can smoothly map points between the Poincaré and Lorentz models.

Equivalence of Hyperbolic Spaces

Using the diffeomorphisms

$$\pi\colon \mathcal{L}^n \to \mathcal{P}^n, \qquad (x_0, x_1, \ldots, x_n) \mapsto \frac{(x_1, \ldots, x_n)}{x_0 + 1}$$

$$\pi^{-1}\colon \mathcal{P}^n \to \mathcal{L}^n, \qquad (x_1, \ldots, x_n) \mapsto \frac{\left(1 + \|\mathbf{x}\|^2, 2x_1, \ldots, 2x_n\right)}{1 - \|\mathbf{x}\|^2}$$

we can smoothly map points between the Poincaré and Lorentz models. Further, these maps preserve all geometric properties (including isometry). So we can do all of our optimization in the Lorentz model, and all of the visualization in the Poincaré model.

Hyperbolicity of a Graph

Definition-Graph Hyperbolicity

Let a, b, c, d be vertices in a graph G and define

$$M_1 = \max\left\{d(a,b) + d(c,d), d(a,c) + d(b,d), d(a,d) + d(b,c)\right\}$$
$$M_2 = \max\left\{d(a,b) + d(c,d), d(a,c) + d(b,d), d(a,d) + d(b,c)\right\} \setminus M_1.$$

We then define $\operatorname{hyp}(a,b,c,d) = M_1 - M_2$. The *hyperbolicity* of the graph G is

$$\delta(G) = 2 \max_{a,b,c,d \in G} \operatorname{hyp}(a,b,c,d).$$

Hyperbolicity of a Graph

Definition-Graph Hyperbolicity

Let a, b, c, d be vertices in a graph G and define

$$M_1 = \max\{d(a,b) + d(c,d), d(a,c) + d(b,d), d(a,d) + d(b,c)\}$$
$$M_2 = \max\{d(a,b) + d(c,d), d(a,c) + d(b,d), d(a,d) + d(b,c)\} \setminus M_1.$$

We then define $\operatorname{hyp}(a,b,c,d) = M_1 - M_2$. The *hyperbolicity* of the graph G is

$$\delta(G) = 2 \max_{a,b,c,d \in G} \operatorname{hyp}(a,b,c,d).$$

A graph has high hyperbolicity if δ is small. For example, trees have $\delta = 0$ and $n \times n$ grids have $\delta = n - 1$.

Table of Contents

Recovering Hierarchies From Embeddings

Let $\mathcal{C} = \{c_i\}_{i=1}^m$ be a set of concepts and $K \in \mathbb{R}^{m \times m}$ be a dataset of pairwise similarity scores between these concepts. Further assume that the concepts have an unobserved hierarchy reflected by a partial order $(\mathcal{C}, \preceq)$.

Recovering Hierarchies From Embeddings

Let $\mathcal{C} = \{c_i\}_{i=1}^{m}$ be a set of concepts and $K \in \mathbb{R}^{m \times m}$ be a dataset of pairwise similarity scores between these concepts. Further assume that the concepts have an unobserved hierarchy reflected by a partial order $(\mathcal{C}, \preceq)$. We would like to recover the partial order $(\mathcal{C}, \preceq)$ from K.

Recovering Hierarchies From Embeddings

Let $\mathcal{C} = \{c_i\}_{i=1}^m$ be a set of concepts and $K \in \mathbb{R}^{m \times m}$ be a dataset of pairwise similarity scores between these concepts. Further assume that the concepts have an unobserved hierarchy reflected by a partial order $(\mathcal{C}, \preceq)$. We would like to recover the partial order $(\mathcal{C}, \preceq)$ from K.

We make two assumptions

A1 Similarity scores describe concepts that are organized in a latent hierarchy

A2 $K_{ij} \geq K_{ik}$ if $c_i \preceq c_j$ and $c_i \not\preceq c_k$.

Our assumptions tell us that we want to preserve the similarity ordering in order to predict comparability.

Recovering Hierarchies From Embeddings, Continued

For a concept c_i, let $\mathcal{N}(i,j) = \{k : K_{ik} < K_{ij}\} \cup \{j\}$ be the set of concepts that are less similar to c_i than c_j is (including c_j).

Recovering Hierarchies From Embeddings, Continued

For a concept c_i, let $\mathcal{N}(i,j) = \{k : K_{ik} < K_{ij}\} \cup \{j\}$ be the set of concepts that are less similar to c_i than c_j is (including c_j).
While we cannot easily decide if c_i and c_j are comparable as a global problem, we can infer that among all $c_k \in \mathcal{N}(i,j)$, c_i is most likely to be comparable to c_j (local).

Recovering Hierarchies From Embeddings, Continued

For a concept c_i, let $\mathcal{N}(i,j) = \{k : K_{ik} < K_{ij}\} \cup \{j\}$ be the set of concepts that are less similar to c_i than c_j is (including c_j).

While we cannot easily decide if c_i and c_j are comparable as a global problem, we can infer that among all $c_k \in \mathcal{N}(i,j)$, c_i is most likely to be comparable to c_j (local).

Let $\phi(i,j)$ be the nearest neighbor of c_i in the set $\mathcal{N}(i,j)$. Then we would like to learn embeddings $\Theta = \{\mathbf{u}_i\}_{i=1}^m$ by optimizing

$$\max_{\Theta} \sum_{i,j} \log \Pr(\phi(i,j) = j \mid \Theta) = \max_{\Theta} \sum_{i,j} \log \frac{e^{-d(\mathbf{u}_i, \mathbf{u}_j)}}{\sum_{k \in \mathcal{N}(i,j)} e^{-d(\mathbf{u}_i, \mathbf{u}_k)}}$$

Recovering Hierarchies From Embeddings, Continued

For a concept c_i, let $\mathcal{N}(i,j) = \{k : K_{ik} < K_{ij}\} \cup \{j\}$ be the set of concepts that are less similar to c_i than c_j is (including c_j).

While we cannot easily decide if c_i and c_j are comparable as a global problem, we can infer that among all $c_k \in \mathcal{N}(i,j)$, c_i is most likely to be comparable to c_j (local).

Let $\phi(i,j)$ be the nearest neighbor of c_i in the set $\mathcal{N}(i,j)$. Then we would like to learn embeddings $\Theta = \{\mathbf{u}_i\}_{i=1}^m$ by optimizing

$$\max_{\Theta} \sum_{i,j} \log \Pr(\phi(i,j) = j \mid \Theta) = \max_{\Theta} \sum_{i,j} \log \frac{e^{-d\left(\mathbf{u}_i, \mathbf{u}_j\right)}}{\sum_{k \in \mathcal{N}(i,j)} e^{-d(\mathbf{u}_i, \mathbf{u}_k)}}$$

This loss produces embeddings where $d(\mathbf{u}_i, \mathbf{u}_j) < d(\mathbf{u}_i, \mathbf{u}_k)$ if $K_{ij} > K_{ik}$. Therefore, we can infer comparability from $d(\mathbf{u}_i, \mathbf{u}_j)$.

Recovering Hierarchies From Embeddings, Continued

For a concept c_i, let $\mathcal{N}(i,j) = \{k : K_{ik} < K_{ij}\} \cup \{j\}$ be the set of concepts that are less similar to c_i than c_j is (including c_j).

While we cannot easily decide if c_i and c_j are comparable as a global problem, we can infer that among all $c_k \in \mathcal{N}(i,j)$, c_i is most likely to be comparable to c_j (local).

Let $\phi(i,j)$ be the nearest neighbor of c_i in the set $\mathcal{N}(i,j)$. Then we would like to learn embeddings $\Theta = \{\mathbf{u}_i\}_{i=1}^m$ by optimizing

$$\max_{\Theta} \sum_{i,j} \log \Pr\left(\phi(i,j) = j \,|\Theta\right) = \max_{\Theta} \sum_{i,j} \log \frac{e^{-d\left(\mathbf{u}_i,\mathbf{u}_j\right)}}{\sum_{k\in\mathcal{N}(i,j)} e^{-d(\mathbf{u}_i,\mathbf{u}_k)}}$$

This loss produces embeddings where $d(\mathbf{u}_i, \mathbf{u}_j) < d(\mathbf{u}_i, \mathbf{u}_k)$ if $K_{ij} > K_{ik}$.

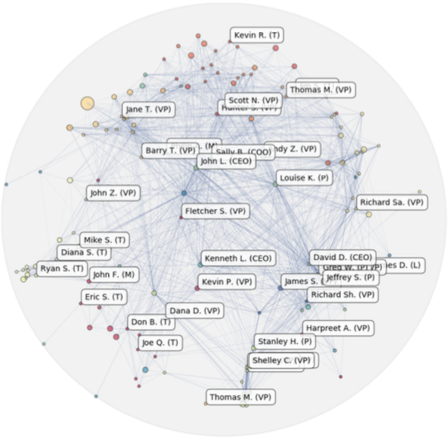
Therefore, we can infer comparability from $d(\mathbf{u}_i, \mathbf{u}_j)$.

Further, more general concepts will be near a lot of points. Thus, they will be pushed towards the center. In other words, we can infer the levels of the hierarchy from the distance from the center (i.e. $\|\mathbf{x}\|$).

Recovering Hierarchies From Embeddings, Continued

Let's look at some visual representations:

Embedding of transitive closure of WORDNET mammals subtree



Embedding of Enron email communications

Recovering Hierarchies From Embeddings, Continued

How do these embeddings compare to Euclidean embeddings?
In terms of reconstruction and link predictions, hyperbolic embedding tends to outperform Euclidean embedding

Recovering Hierarchies From Embeddings, Continued

How do these embeddings compare to Euclidean embeddings?
In terms of reconstruction and link predictions, hyperbolic embedding tends to outperform Euclidean embedding

- Related concepts are more likely to be amongst the closest nodes.
- More accurate in determining if there exists a link between two nodes
- Hyperbolic outperfroms Euclidean significantly in low dimensions

Hyperbolic Graph Neural Networks

Graph neural networks (GNNs) can be interpreted as passing messages between nodes.

Hyperbolic Graph Neural Networks

Graph neural networks (GNNs) can be interpreted as passing messages between nodes.

Mathematically, the message from a node v to a neighbor u is $\mathbf{m}_v^{k+1} = \widetilde{A}_{uv} W^k \mathbf{h}_v^k$, where $\mathbf{h}_v^k$ is the representation of the node v at step k, $W^k \in \mathbb{R}^{h \times h}$ constitute the trainable parameters for step k, and $\widetilde{A}$ captures the connectivity of the graph.

Graph neural networks (GNNs) can be interpreted as passing messages between nodes.

Mathematically, the message from a node v to a neighbor u is $\mathbf{m}_v^{k+1} = \widetilde{A}_{uv} W^k \mathbf{h}_v^k$, where $\mathbf{h}_v^k$ is the representation of the node v at step k, $W^k \in \mathbb{R}^{h \times h}$ constitute the trainable parameters for step k, and $\widetilde{A}$ captures the connectivity of the graph.

To obtain the new representation of u at step k, we add up all messages from its neighbors and then apply an activation function σ. Thus in a more compact way,

$$\mathbf{h}_u^{k+1} = \sigma \left(\sum_{v \in \mathcal{N}(u)} \widetilde{A}_{uv} W^k \mathbf{h}_v^k \right)$$

Hyperbolic Graph Neural Networks, Continued

In a hyperbolic graph neural network, our node representations lie in some hyperbolic space. Thus, we cannot just multiply our representations against some matrix in a meaningful way.

Hyperbolic Graph Neural Networks, Continued

In a hyperbolic graph neural network, our node representations lie in some hyperbolic space. Thus, we cannot just multiply our representations against some matrix in a meaningful way.

Despite this, the tangent space to any given point in hyperbolic space is Euclidean, and we can perform our propogation there. Thus,

$$\mathbf{h}_u^{k+1} = \sigma\left(\exp_{\mathbf{x}}\left(\sum_{v \in \mathcal{N}(u)} \widetilde{A}_{uv} W^k \log_{\mathbf{x}}\left(\mathbf{h}_v^k\right)\right)\right)$$

Hyperbolic Graph Neural Networks, Continued

In a hyperbolic graph neural network, our node representations lie in some hyperbolic space. Thus, we cannot just multiply our representations against some matrix in a meaningful way.

Despite this, the tangent space to any given point in hyperbolic space is Euclidean, and we can perform our propogation there. Thus,

$$\mathbf{h}_u^{k+1} = \sigma\left(\exp_{\mathbf{x}}\left(\sum_{v \in \mathcal{N}(u)} \widetilde{A}_{uv} W^k \log_{\mathbf{x}}\left(\mathbf{h}_v^k\right)\right)\right)$$

Note that we must perform the activation function after the exponential map since $\log_{\mathbf{x}}(\exp_{\mathbf{x}}(\sigma(\cdot))) = \sigma(\cdot)$. Thus in this other setting, the neural network would send $\log_{\mathbf{x}} \mathbf{h}_v^0$ through a standard Euclidean graph neural network for k steps and then exponentiate back to hyperbolic space.

Hyperbolic Graph Neural Networks, Continued

How do these perform against Euclidean GNNs?

Hyperbolic Graph Neural Networks, Continued

How do these perform against Euclidean GNNs?

Below is a table of mean absolute error of predicting certain moleculuar properties of compounds based on data from a dataset of commercially-available drug-like compounds called ZINC. The table compares a Euclidean GNN, a Poincaré-based hyperbolic GNN, and a Lorentz-based hyperbolic GNN of dimensions 3, 5, 10, 20, and 256.

	logP				
	3	5	10	20	256
Euclidean	6.7 ± 0.07	4.7 ± 0.03	4.7 ± 0.02	3.6 ± 0.00	3.3 ± 0.00
Poincare	5.7 ± 0.00	4.6 ± 0.03	3.6 ± 0.02	3.2 ± 0.01	3.1 ± 0.01
Lorentz	5.5 ± 0.02	4.5 ± 0.03	3.3 ± 0.03	2.9 ± 0.01	2.4 ± 0.02
	QED				
	3	5	10	20	256
Euclidean	22.4 ± 0.21	15.9 ± 0.14	14.5 ± 0.09	10.2 ± 0.08	6.4 ± 0.06
Poincare	22.1 ± 0.01	14.9 ± 0.13	10.2 ± 0.02	6.9 ± 0.02	6.0 ± 0.04
Lorentz	21.9 ± 0.12	14.3 ± 0.12	8.7 ± 0.04	6.7 ± 0.06	4.7 ± 0.00
	SAS				
	3	5	10	20	256
Euclidean	20.5 ± 0.04	16.8 ± 0.07	14.5 ± 0.11	9.6 ± 0.05	9.2 ± 0.08
Poincare	18.8 ± 0.03	16.1 ± 0.08	12.9 ± 0.04	9.3 ± 0.07	8.6 ± 0.02
Lorentz	18.0 ± 0.15	16.0 ± 0.15	12.5 ± 0.07	9.1 ± 0.08	7.7 ± 0.06

Table 2: Mean absolute error of predicting molecular properties: the water-octanal partition coefficient (logP); qualitative estimate of drug-likeness (QED); and synthetic accessibility score (SAS). Scaled by 100 for table formatting (low is good).

To the right is a table showing the accuracy of predicting price fluctuations for the Etherium/US$ market based on graph dynamics. Node2vec reflects a control of inputting averaged 128-dimensional node2vex features into an MLP classifier, ARIMA is the autoregressive integrated moving average, and Euclidean, Poincaré, and Lorentz refer to the respective GNNs.

	Dev
Node2vec	54.10 ± 1.63
ARIMA	54.50 ± 0.16
Euclidean	56.15 ± 0.30
Poincare	57.03 ± 0.28
Lorentz	57.52 ± 0.35

Hyperbolic Graph Neural Networks, Continued

We can also modify our propogation to reflect the curvature of the hyperbolic space.

Hyperbolic Graph Neural Networks, Continued

We can also modify our propogation to reflect the curvature of the hyperbolic space.

The Poincaré and Lorentz models previously described reflect hyperbolic space of curvature -1. Adjusting the radius of the Poincaré model and the Lorentz scalar product can modify the curvature of the hyperbolic space to any negative number. We can think of a more-negative curvature as being "more hyperbolic".

Hyperbolic Graph Neural Networks, Continued

We can also modify our propogation to reflect the curvature of the hyperbolic space.

The Poincaré and Lorentz models previously described reflect hyperbolic space of curvature -1. Adjusting the radius of the Poincaré model and the Lorentz scalar product can modify the curvature of the hyperbolic space to any negative number. We can think of a more-negative curvature as being "more hyperbolic".

If we change our propogation to

$$\mathbf{h}_u^{k+1} = \sigma\left(\exp_{\mathbf{x}}^{K_{k+1}}\left(\sum_{v\in\mathcal{N}(u)} \widetilde{A}_{uv} W^k \log_{\mathbf{x}}^{K_k}\left(\mathbf{h}_v^k\right)\right)\right)$$

then we can make the curvature a trainable paramater.

Hyperbolic Graph Neural Networks, Continued

How does a hyperbolic GNN with trainable curvature compare to Euclidean GNNs?

Hyperbolic Graph Neural Networks, Continued

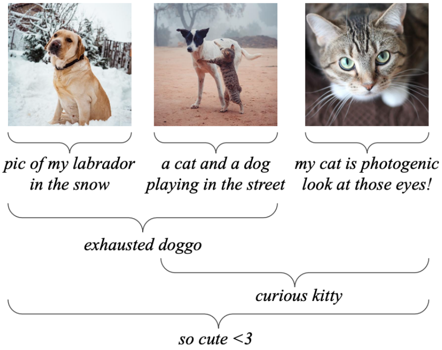
How does a hyperbolic GNN with trainable curvature compare to Euclidean GNNs?

Below is a table of area under the receiver-operating characteristic curve (ROC AUC) for link prediction (LP) and F1 scores for node classification (NC) for various models trained on various datasets.

	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
	Hyp [29]	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
	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
	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
	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[21]	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
	GAT [41]	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	$\mathbf{93.7} \pm 0.1$	$\mathbf{83.0} \pm 0.7$
	SAGE [15]	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
	SGC [44]	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	HGCN	$\mathbf{90.8} \pm 0.3$	$\mathbf{74.5} \pm 0.9$	$\mathbf{78.1} \pm 0.4$	$\mathbf{72.2} \pm 0.5$	$\mathbf{84.5} \pm 0.4$	$\mathbf{74.6} \pm 0.3$	$\mathbf{96.4} \pm 0.1$	$\mathbf{90.6} \pm 0.2$	$\mathbf{96.3} \pm 0.0$	$\mathbf{80.3} \pm 0.3$	92.9 ± 0.1	79.9 ± 0.2
	(%) 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%

As we can see, hyperbolic GNNs perform better than Euclidean GNNs for graphs with high hyperbolicity.

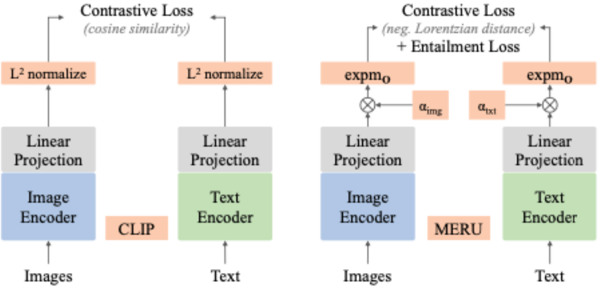
Hyperbolic Computer Vision



Images and texts can be jointly viewed in a visual-semantic hierarchy. So, we might expect hyperbolic representations of text/images may be better than usual methods.

Hyperbolic Computer Vision, Continued

Let's first examine MERU, a hyperbolic version of CLIP



Hyperbolic Computer Vision, Continued

How does MERU compare to CLIP?

Hyperbolic Computer Vision, Continued

How does MERU compare to CLIP?
MERU performed better than CLIP in zero-shot image and text retrieval and zero-shot image classification.

Hyperbolic Computer Vision, Continued

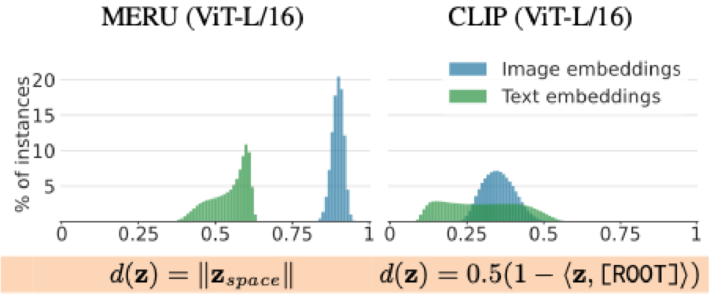
How does MERU compare to CLIP?
MERU performed better than CLIP in zero-shot image and text retrieval and zero-shot image classification.
MERU consistently performs better than CLIP at low embedding widths.
So for resource-constrained applications, hyperbolic embeddings may be more appealing than Euclidean embeddings.

How does MERU compare to CLIP?

MERU performed better than CLIP in zero-shot image and text retrieval and zero-shot image classification.

MERU consistently performs better than CLIP at low embedding widths.

So for resource-constrained applications, hyperbolic embeddings may be more appealing than Euclidean embeddings.



Hyperbolic Computer Vision, Continued

MERU	CLIP
a bengal cat sitting beside wheatgrass on a white surface	*a bengal cat sitting beside wheatgrass on a white surface*
bengal	↓
cat	↓
domestic	↓
[ROOT]	[ROOT]

MERU	CLIP
white horse	*white horse*
equine	↓
equestrian	↓
beauty	↓
female	↓
fluffy	↓
[ROOT]	[ROOT]

MERU	CLIP
photography of rainbow during cloudy sky	*phenomenon*
rainbow	↓
phenomenon	↓
rural	↓
[ROOT]	[ROOT]

MERU	CLIP
retro photo camera on table	↓
fujinomiya	↓
vintage	↓
style	↓
[ROOT]	[ROOT]

MERU	CLIP
avocado toast	*avocado toast*
healthy breakfast	*delicious*
delicious	↓
homemade	↓
fresh	↓
[ROOT]	[ROOT]

MERU	CLIP
brooklyn bridge	*photo of brooklyn bridge, new york*
new york city	*new york city*
city	*new york*
outdoors	↓
day	↓
[ROOT]	[ROOT]

MERU	CLIP
taj mahal	*taj mahal through an arch*
monument	*travel*
architecture	*inspiration*
travel	↓
day	↓
[ROOT]	[ROOT]

MERU	CLIP
sydney opera house	*sydney opera house*
opera house	*opera house*
holiday	*gift*
day	*beauty*
[ROOT]	[ROOT]

Hyperbolic Computer Vision, Continued

Khrulkov et al. have developed Hyperbolic ProtoNet to classify images in few-shot learning. After embedding images into the Poincaré ball, they compare the embedding of the image to the hyperbolic mean of a class of images to classify an image.

Hyperbolic Computer Vision, Continued

Khrulkov et al. have developed Hyperbolic ProtoNet to classify images in few-shot learning. After embedding images into the Poincaré ball, they compare the embedding of the image to the hyperbolic mean of a class of images to classify an image.
Hyperbolic ProtoNet significantly outperforms Euclidean ProtoNet, especially in one-shot learning.

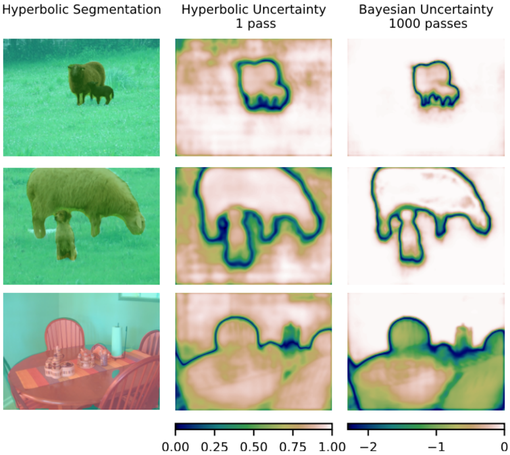
Hyperbolic Computer Vision, Continued

Khrulkov et al. have developed Hyperbolic ProtoNet to classify images in few-shot learning. After embedding images into the Poincaré ball, they compare the embedding of the image to the hyperbolic mean of a class of images to classify an image.

Hyperbolic ProtoNet significantly outperforms Euclidean ProtoNet, especially in one-shot learning.

Hyperbolic ProtoNet outperforms more advanced methods as well.

Image segmentation can also be viewed in a hierarchical manner. As such, GhadimiAtigh et al. have looked into hyperbolic embeddings of images for image segmentation.



Hyperbolic Computer Vision, Continued

While in medium (10) to high (256) dimensions, hyperbolic segmentation is comparable to Euclidean segmentation, hyperbolic segmentation again outperforms at low dimensions, which is ideal for low-resource situation (such as on-device segmentation)

Hyperbolic Computer Vision, Continued

Many of the projects we have looked at required some kind of transformer to turn the graph into meaningful data.

Hyperbolic Computer Vision, Continued

Many of the projects we have looked at required some kind of transformer to turn the graph into meaningful data.
In these previous results, a transformer turns the media into a Euclidean vector, which is then translated to hyperbolic space using an exponential map. But what if the transformer could cut out the middle-man and convert the media directly into hyperbolic representations?

Hyperbolic Computer Vision, Continued

Many of the projects we have looked at required some kind of transformer to turn the graph into meaningful data.

In these previous results, a transformer turns the media into a Euclidean vector, which is then translated to hyperbolic space using an exponential map. But what if the transformer could cut out the middle-man and convert the media directly into hyperbolic representations?

Yang et al. have developed Hypformer, a hyperbolic transformer that works fully in hyperbolic space. This new transformer outperforms GNN and HGNN models and all graph sizes. It also showed significant improvement in low-hyperbolicity graphs.

Hyperbolic Computer Vision, Continued

Many of the projects we have looked at required some kind of transformer to turn the graph into meaningful data.
In these previous results, a transformer turns the media into a Euclidean vector, which is then translated to hyperbolic space using an exponential map. But what if the transformer could cut out the middle-man and convert the media directly into hyperbolic representations?
Yang et al. have developed Hypformer, a hyperbolic transformer that works fully in hyperbolic space. This new transformer outperforms GNN and HGNN models and all graph sizes. It also showed significant improvement in low-hyperbolicity graphs.
However, Hypformer is more sensitive to the graph structure in some data sets than others. For example, when training on CORA, a citation network, removing the graph structure resulted in a performance drop, but for 20news, a dataset of news-based text documents, Hypformer performed best without the graph structure.

Other Applications

Researchers have use Spatio-Temporal Graph Convolutional Networks to encode body movements into graphs. An application of such networks is automatic sign language translation.

Other Applications

Researchers have use Spatio-Temporal Graph Convolutional Networks to encode body movements into graphs. An application of such networks is automatic sign language translation.

When embedding into Euclidean space, there is a lot of distortion. For example, the ASL sign for water (forming a “W” shape with the fingers and tapping the chin/lips twice) requires the arm to move down after. The movement of the arm/wrist dominate the Euclidean representation.

Similarly, two signs that differ in timing or precision may have nearly indistinguishable Euclidean representations.

Other Applications

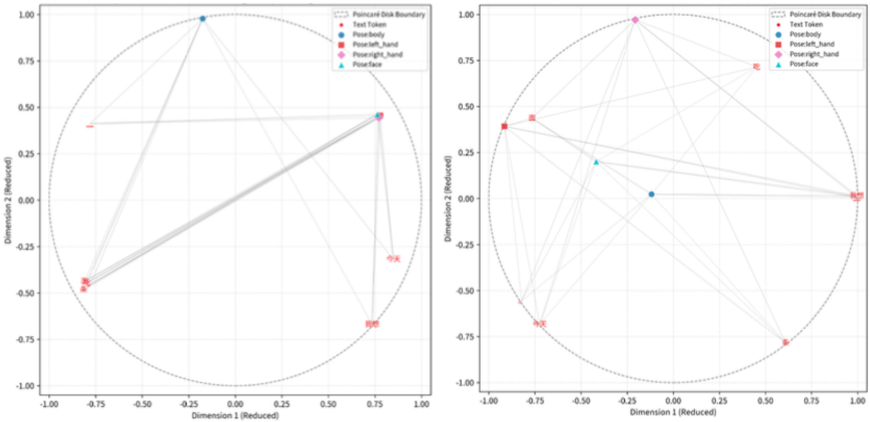
Researchers have use Spatio-Temporal Graph Convolutional Networks to encode body movements into graphs. An application of such networks is automatic sign language translation.

When embedding into Euclidean space, there is a lot of distortion. For example, the ASL sign for water (forming a "W" shape with the fingers and tapping the chin/lips twice) requires the arm to move down after. The movement of the arm/wrist dominate the Euclidean representation.

Similarly, two signs that differ in timing or precision may have nearly indistinguishable Euclidean representations.

Fish and Bowden have developed Geo-Sign, a hyperbolic automatic sign language translator.

Other Applications, Continued



Becuase of the improved embedding, Geo-Sign outperforms more conventional automatic sign language translation.

Table of Contents

Advantages of Hyperbolic Geometry in Neural Networks

- Can detect latent hierarchies
- Better results for high-hyperbolicity graphs than Euclidean embeddings
- Better results for most graphs at lower dimensions (ideal for on-device computations)
- Easier visualizations

Advantages of Hyperbolic Geometry in Neural Networks

- Can detect latent hierarchies
- Better results for high-hyperbolicity graphs than Euclidean embeddings
- Better results for most graphs at lower dimensions (ideal for on-device computations)
- Easier visualizations

Disadvantages of Hyperbolic Geometry in Neural Networks

- Not ideal for graphs with low-hyperbolicity
- Potential for computational issues in exponentiation

References I

- A. Barabási, E. Ravasz, & Z. Oltvai - *Hierarchical Organization of Modularity in Complex Networks* (2004).
- M. Boguñá, F. Papadopoulos, & D. Krioukov - *Sustaining the Internet with hyperbolic mapping* (2010). Nature.
- B. Chamberlain, J. Clough, & M. Deisenroth - *Neural Embeddings of Graphs in Hyperbolic Space* (2017). arXiv:1705.10359v1
- I. Chami, R. Ying, C. Ré, & J Leskovec - *Hyperbolic Graph Convolutional Neural Networks* (2019). arXiv:1910.12933v1
- W. Chen , W. Fang , G. Hu, & M. W. Mahoney - *On the Hyperbolicity of Small-World and Treelike Random Graphs* (2013).
- Y. Chen et al - *Modeling Scale-free Graphs with Hyperbolic Geometry for Knowledge-aware Recommendation* (2022). arXiv:2108.06468v3

References II

- K. Desai et al - *Hyperbolic Image-Text Representations* (2023). arXiv:2304.09172v3
- A. Ermolov, L. Mirvakhabova, V. Khrulkov, N. Sebe, & I.Oseledets - *Hyperbolic Vision Transformers: Combining Improvements in Metric Learning* (2022). arXiv:2203.10833
- E. Fish & R. Bowden - *Geo-Sign: Hyperbolic Contrastive Regularisation for Geometrically Aware Sign Language Translation* (2025). arXiv:2506.00129v1
- M. GhadimiAtigh, J. Schoep, E. Acar, N. van Noord, &P. Mettes - *Hyperbolic Image Segmentation* (2022). arXiv:2203.05898
- N. He et al - *HELM: Hyperbolic Large Language Models via Mixture-of-Curvature Experts* (2025). arXiv:2505.24722v1

References III

- V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, & V. Lempitsky - *Hyperbolic Image Embeddings* (2019). arXiv:1904.02239
- A. Klimovskaia, D. Lopez-Paz1, L. Bottou, & M. Nickel - *Poincaré maps for analyzing complex hierarchies in single-cell data* (2020). Nature
- N. Linial, E. London, & Y. Rabinovich - *The Geometry of Grpahs and some of its Algorithmic Applications* (1994). Combinatorica
- Q. Liu, M. Nickel, & D. Kiela - *Hyperbolic Graph Neural Networks* (2019). arXiv:1910.12892v1
- Y. Liu, Z. He, & K. Han - *Hyperbolic Category Discovery* (2025). arXiv:2504.06120

References IV

- P. Mandica, L. Franco, K. Kallidromitis, S. Petryk, & F. Galasso - *Hyperbolic Learning with Multimodal Large Language Models* (2024). arXiv:2408.05097v1
- M. Nickel & D. Kiela - *Poincaré Embeddings for Learning Hierarchical Representations* (2017). arXiv:1705.08039v2
- M. Nickel & D. Kiela - *Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry* (2018). arXiv:1806.03417v2
- F. Papadopoulos, M. Kitsak, M. A. Serrano, M. Boguñá, & D. Krioukov - *Popularity versus similarity in growing networks* (2013). arXiv:1106.0286v3
- W. Peng et al - *Hyperbolic Deep Neural Networks: A Survey* (2021). arXiv:2101.04562v3

References V

- F. Sala, C. De Sa, A. Gu, & C. Ré - *Representation Tradeoffs for Hyperbolic Embeddings* (2018). arXiv:1804.03329v2
- R. Sarkar - *Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane* (2011).
- M. Xu - *Understanding Graph Embedding Methods and Their Applications* (2021). arXiv:2012.08019v1
- M. Yang et al - *Hypformer: Exploring Efficient Hyperbolic Transformer Fully in Hyperbolic Space* (2024).
- Y. Zhao & J. Chen - *A Survey on Differential Privacy for Unstructured Data Content* (2022).