Hyperbolic Machine Learning

Group Meeting Presentation

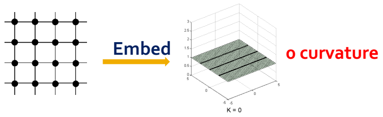
Eric Qu
zq32@duke.edu
Duke Kunshan University

February 11, 2022

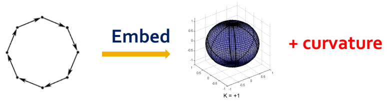
Outline

- Hyperbolic Geometry
- Hyperbolic Neural Networks
- Applications in Molecular Generation

- Why do we care about non-Euclidean embedding space?
  - The underlying geometric structure of embeddings space is beneficial for representation.
- Example: Graphs
  - Grid-like graphs

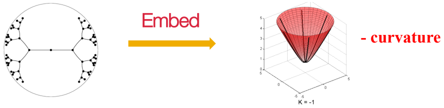


  - Graphs with many large cycles

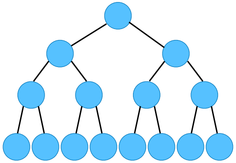


  - Hierarchical, tree-like graphs?

- We could use hyperbolic space!



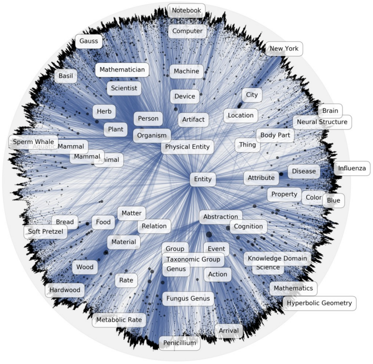
- Why not Euclidean? Consider the following tree data.
  - # of nodes grow exponentially with depth



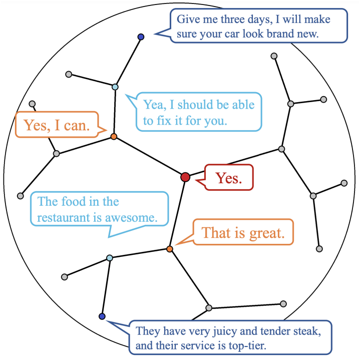
- Euclidean embedding is too clustered.
- Where can we find hierarchical data?

Hyperbolic Geometry
Hyperbolic Embeddings

- Hierarchical structure in NLP

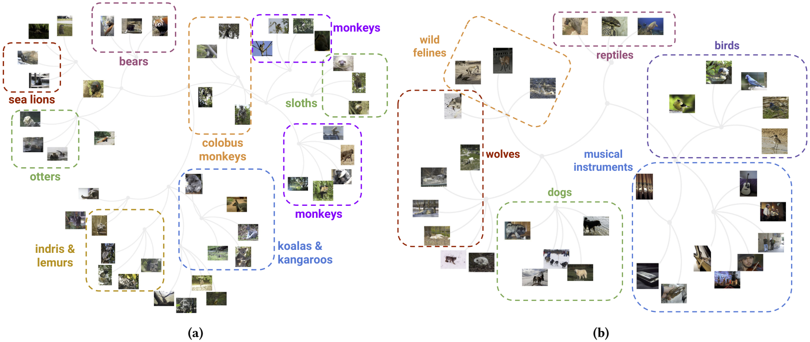


Nickel and Kiela (2017)


Give me three days, I will make sure your car look brand new.
Yea, I should be able to fix it for you.
Yes, I can.
Yes.
The food in the restaurant is awesome.
That is great.
They have very juicy and tender steak, and their service is top-tier.

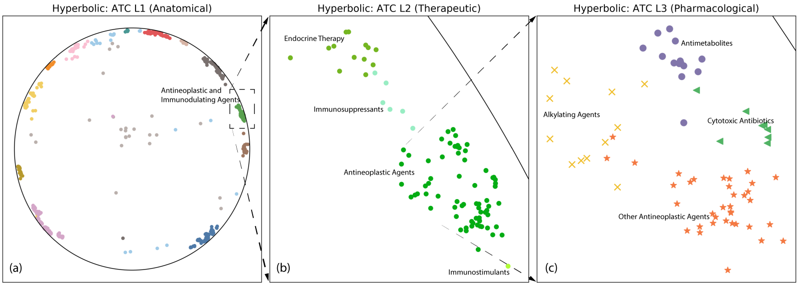
Dai et al. (2021)

- Hierarchical structure in CV



Monath et al. (2019)

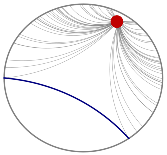
- Hierarchical structure in Drugs
  - "Anatomical Therapeutic Chemical Classification System (ATC), groups drugs that are similar in terms of their mechanism of action and therapeutic, pharmacological and chemical characteristics." (Yu et al., 2020)



Yu et al. (2020)

Hyperbolic Geometry
Introduction

- Hyperbolic geometry differs from Euclidean geometry by the parallel postulate (5th axiom)
  1. A straight line may be drawn between any two points.
  2. Any terminated straight line may be extended indefinitely.
  3. A circle may be drawn with any given point and radius.
  4. All right angles are equal.
  5. *In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point.*
- In hyperbolic geometry, there are infinite number of lines parallel to the given line through the point.

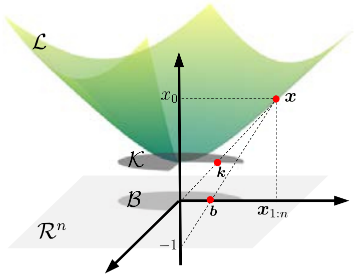


Set of geodesic lines from the red point to boundary of the Poincare ball that are parallel to the blue line

A good video illustration: *https://www.youtube.com/watch?v=zQo_S3yNa2w*

Hyperbolic Geometry
Introduction

- We represent points in hyperbolic space by geometry model (coordinate system).
- Five classic models
  - *Lorentz model*
  - *Poincaré ball model*
  - Hemisphere model
  - Klein model
  - Poincaré half-space model

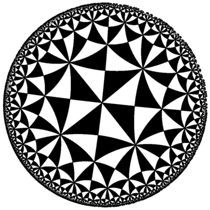


- Two commonly used ones are Lorentz model and Poincaré ball model (Poincare model in short).

Hyperbolic Geometry

Introduction

- Poincaré Model
  - Radius proportional to $\sqrt{1/K}$ (K: curvature)
  - Open ball (exclude boundary)
  - Each triangle in the figure has the same area
  - Needs to represent exponentially many embedding points close to the boundary of the Poincare ball
  - Suitable for visualization
- Lorentz model
  - Upper sheet of 2-sheet hyperboloid
  - Numerically more stable and sufficient space for optimization (Nickel and Kiela, 2018)
  - We will use this model from now on


2D Poincaré ball

2D Lorentz model

- The Lorentz model $\mathbb{L}_K^n = (\mathcal{L}, \mathfrak{g}^K)$
  - K: Constant negative curvature (The more negative is the curvature, the more curved the space is)
  - $\mathcal{L}$: n-dimensional manifold embedded in the $(n+1)$-dimensional Minkowski space
  - $\mathfrak{g}^K = \mathrm{diag}([-1, \mathbf{1}_n^\top])$: Riemannian metric tensor
- Every point $\boldsymbol{x} \in \mathbb{L}_K^n$ is represented by $\boldsymbol{x} = \begin{bmatrix} x_t \\ \boldsymbol{x}_s \end{bmatrix}$, $x_t > 0$, $\boldsymbol{x}_s \in \mathbb{R}^n$ and satisfies $\langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\mathcal{L}} = 1/K$
  - $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ is the Lorentz inner product induced by $\mathfrak{g}^K$:

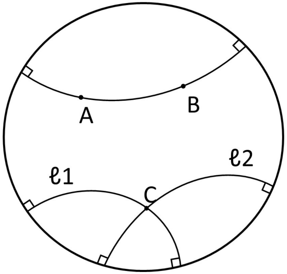
$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} := \boldsymbol{x}^\top \mathfrak{g}^K \boldsymbol{y} = -x_t y_t + \boldsymbol{x}_s^\top \boldsymbol{y}_s,\ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}_K^n.$$

- Following the convention in special relativity, x_t is the “time axis” and $\boldsymbol{x}_s$ is the “spatial axes”.

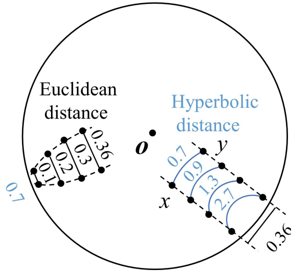
Hyperbolic Geometry
Geodesics and Distances

- Geodesics are shortest paths in a manifold. (generalization of "straight lines" in Euclidean geometry)
- Distance: length of geodesic. Given by,

$$d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{\sqrt{-K}} \cosh^{-1}(K\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}).$$



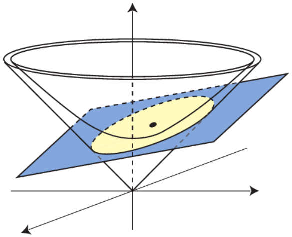
Geodesics



Distances

Hyperbolic Geometry
Tangent Space

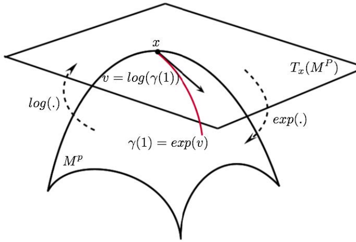
- For each point $\boldsymbol{x} \in \mathbb{L}_K^n$, the tangent space at $\boldsymbol{x}$ is $\mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n := \{\boldsymbol{y} \in \mathbb{R}^{n+1} \mid \langle \boldsymbol{y}, \boldsymbol{x} \rangle_{\mathcal{L}} = 0\}$.
- The first order approximation of hyperbolic manifold around $\boldsymbol{x}$
- It is a subspace of $\mathbb{R}^{n+1}$! We can perform Euclidean operations on tangent space.
- $\|\boldsymbol{v}\|_{\mathcal{L}} = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\mathcal{L}}}$ is the norm of $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n$



Tangent space

Hyperbolic Geometry
Exp & Log map

- For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}_K^n$ and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n$
- Exponential map $\exp_{\boldsymbol{x}}^K(\boldsymbol{v}) : \mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n \to \mathbb{L}_K^n$
  - Map tangent vectors to hyperbolic spaces
  - $\exp_{\boldsymbol{x}}^K(\boldsymbol{v}) := \gamma(1)$, γ is the geodesic s.t. $\gamma(0) = \boldsymbol{x}$ and $\gamma'(0) = \boldsymbol{v}$.
- Logarithmic map $\log_{\boldsymbol{x}}^K(\boldsymbol{y}) : \mathbb{L}_K^n \to \mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n$
  - Inverse of exponential map
  - $\log_{\boldsymbol{x}}^K(\exp_{\boldsymbol{x}}^K(\boldsymbol{v})) = \boldsymbol{v}$



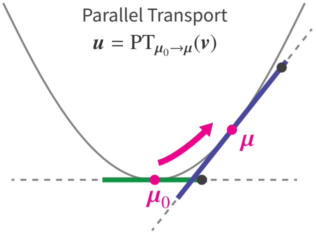
Exp and Log Map

$$\exp_{\boldsymbol{x}}^K(\boldsymbol{v}) = \cosh(\phi)\boldsymbol{x} + \sinh(\phi)\frac{\boldsymbol{v}}{\phi}, \quad \phi = \sqrt{-K}\|\boldsymbol{v}\|_{\mathcal{L}}$$

$$\log_{\boldsymbol{x}}^K(\boldsymbol{y}) = \frac{\cosh^{-1}(\psi)}{\sqrt{\psi^2 - 1}}(\boldsymbol{y} - \psi\boldsymbol{x}), \quad \psi = K\langle\boldsymbol{x}, \boldsymbol{y}\rangle_{\mathcal{L}}$$

Hyperbolic Geometry
Parallel Transport

- For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}_K^n$ and $\boldsymbol{v}, \boldsymbol{w} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n$
- Parallel transport $\mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}^K : \mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n \to \mathcal{T}_{\boldsymbol{y}}\mathbb{L}_K^n$
  - Transport vector from $\mathcal{T}_{\boldsymbol{x}}\mathbb{L}_K^n$ to $\mathcal{T}_{\boldsymbol{y}}\mathbb{L}_K^n$ along the geodesic from $\boldsymbol{x}$ to $\boldsymbol{y}$.
  - It preserves metric, i.e. $\left\langle \mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}^K(\boldsymbol{v}), \mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}^K(\boldsymbol{u}) \right\rangle_{\mathcal{L}} = \langle \boldsymbol{v}, \boldsymbol{u} \rangle_{\mathcal{L}}$



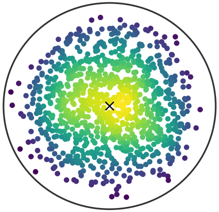
Parallel Transport
$\boldsymbol{u} = \mathrm{PT}_{\mu_0\to\mu}(\boldsymbol{v})$
$\boldsymbol{\mu}$
$\boldsymbol{\mu}_0$

Parallel transport

$$\mathrm{PT}_{\boldsymbol{x}\to\boldsymbol{y}}^K(\boldsymbol{v}) = \frac{\langle \boldsymbol{y}, \boldsymbol{v} \rangle_{\mathcal{L}}}{-1/K - \langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}}}(\boldsymbol{x} + \boldsymbol{y})$$

Hyperbolic Geometry
Wrapped Normal Distribution

- To sample $\boldsymbol{z} \in \mathbb{L}_K^n$ from wrapped normal distribution $\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (Nagano et al., 2019)
  a. Sample $\hat{\boldsymbol{v}}$ from Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ defined over $\mathbb{R}^n$
  b. Rewrite $\hat{\boldsymbol{v}}$ as $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{o}}\mathbb{L}_K^n$ by $\boldsymbol{v} = \begin{bmatrix} 0 \\ \hat{\boldsymbol{v}} \end{bmatrix}$
  c. Parallel transport $\boldsymbol{v}$ to $\boldsymbol{u} = \mathrm{PT}^K_{\boldsymbol{o}\to\boldsymbol{\mu}}(\boldsymbol{v})$
  d. Map $\boldsymbol{u}$ to $\boldsymbol{z} = \exp_{\boldsymbol{\mu}}(\boldsymbol{u}) \in \mathbb{L}_K^n$


Wrapped Normal

Let $\mathrm{proj}_{\boldsymbol{\mu}} := \exp_{\boldsymbol{\mu}} \circ \ \mathrm{PT}^K_{\boldsymbol{o}\to\boldsymbol{\mu}}$. The probability density of $\mathcal{G}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ at $\boldsymbol{z} = \mathrm{proj}_{\boldsymbol{\mu}}(\boldsymbol{v})$ is

$$\begin{aligned}\log p(\boldsymbol{z}) &= \log p(\boldsymbol{v}) - \log\det\left(\frac{\partial \ \mathrm{proj}_{\boldsymbol{\mu}}(\boldsymbol{v})}{\partial \boldsymbol{v}}\right) \\ &= \log p(\boldsymbol{v}) - \log\left(\frac{\sinh \left\|\exp_{\boldsymbol{\mu}}^{-1}(\boldsymbol{z})\right\|_{\mathcal{L}}}{\left\|\exp_{\boldsymbol{\mu}}^{-1}(\boldsymbol{z})\right\|_{\mathcal{L}}}\right)^{n-1}\end{aligned}$$

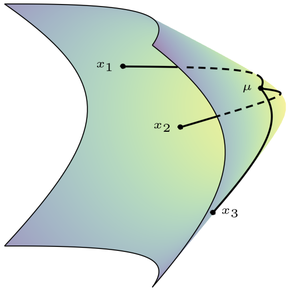
Hyperbolic Geometry
Centroid

- Centroid or center of mass (Law et al., 2019)
  - Useful in self-attention, feature aggregation in GNN, convolution
  - Define squared Lorentzian distance as $d_{\mathcal{L}}^2(\boldsymbol{x}, \boldsymbol{y}) = 2/K - 2\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}},\ \boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}_K^n$
  - The centroid is the minimizer that solves

$$\min_{\boldsymbol{\mu} \in \mathbb{L}_K^n} \sum_{i=1}^{N} \nu_i d_{\mathcal{L}}^2(\boldsymbol{x}_i, \boldsymbol{\mu})$$

subject to $\boldsymbol{x}_i \in \mathbb{L}_K^n$, $\nu_i \geq 0$, $\sum_i \nu_i > 0$.

$$\boldsymbol{\mu} = \mathrm{HCent}(\boldsymbol{X}, \boldsymbol{\nu}) = \frac{\sum_{i=1}^{N} \nu_i \boldsymbol{x}_i}{\sqrt{-K} \left| \| \sum_{i=1}^{N} \nu_i \boldsymbol{x}_i \|_{\mathcal{L}} \right|}, \tag{1}$$



Centroid

- Concatenation & split are not well defined in hyperbolic space
- Poincaré β-concatenation and β-split in the Poincaré model Shimizu et al. (2021).
  - Perform concatenation and split in the tangent space.
  - Use β regularization to keep it in the ball.
- Similarly, we could define "Lorentz Tangent Concatenation"
  - Suppose $\{\boldsymbol{x}_i\}_{i=1}^{N}$ and $\boldsymbol{x}_i \in \mathbb{L}_K^{n_i}$
  - Lift each $\boldsymbol{x}_i$ to $\boldsymbol{v}_i = \log_{\boldsymbol{o}}^{K}(\boldsymbol{x}_i) = \begin{bmatrix} v_{i_t} \\ \boldsymbol{v}_{i_s} \end{bmatrix} \in \mathbb{R}^{n_i+1}$
  - Perform Euclidean concatenation $\boldsymbol{v} \coloneqq \left(0, \boldsymbol{v}_{1_s}^{\top}, \ldots, \boldsymbol{v}_{N_s}^{\top}\right)^{\top}$.
  - Map $\boldsymbol{v}$ back to $\boldsymbol{y} = \exp_{\boldsymbol{o}}^{K}(\boldsymbol{v})$

Hyperbolic Geometry

Concatenation and Split

- We also have "Lorentz Tangent Split"
  - Suppose $\boldsymbol{x} \in \mathbb{L}_K^n$ with split sub-dimensions $\sum_{i=1}^N n_i = n$
  - Map $\boldsymbol{x}$ to tangent space and perform Euclidean split $\boldsymbol{v} = \log_{\boldsymbol{o}}^K(\boldsymbol{x}) = \left(0, \boldsymbol{v}_{1_s}^\top \in \mathbb{R}^{n_1}, \ldots, \boldsymbol{v}_{N_s}^\top \in \mathbb{R}^{n_N}\right)^\top$
  - Rewrite as tangent vector $\boldsymbol{v}_i = \left[\begin{smallmatrix} 0 \\ \boldsymbol{v}_{i_s} \end{smallmatrix}\right] \in \mathcal{T}_{\boldsymbol{o}}\mathbb{L}_K^{n_i}$
  - Map back to $\boldsymbol{y}_i = \exp_{\boldsymbol{o}}^K(\boldsymbol{v}_i)$.
- This method is numerically unstable
  - Too many Exp and Log maps
  - Numerous concatenation and split between time and space axis
  - (result in NaN Loss)

Hyperbolic Geometry
Concatenation and Split

- We propose "Lorentz Direct Concatenation"
  - Suppose $\{\boldsymbol{x}_i\}_{i=1}^N$ where each $\boldsymbol{x}_i \in \mathbb{L}_K^{n_i}$ and $M = \sum_{i=1}^N n_i$

$$\boldsymbol{y} = \mathrm{HCat}(\{\boldsymbol{x}_i\}_{i=1}^N) = \begin{bmatrix} \sqrt{\sum_{i=1}^N x_{i_t}^2 + \frac{N-1}{K}} \\ \boldsymbol{x}_{1_s} \\ \vdots \\ \boldsymbol{x}_{N_s} \end{bmatrix} \in \mathbb{L}_k^M. \qquad (2)$$

- "Lorentz Direct Split"
  - Suppose $\boldsymbol{x} \in \mathbb{L}_K^n$ with split sub-dimensions $\sum_{i=1}^N n_i = n$
  - Split $\boldsymbol{x}$ in the space dimension and calc the time dimension

$$\boldsymbol{x} = \begin{bmatrix} x_t \\ \boldsymbol{y}_{1_s} \\ \vdots \\ \boldsymbol{y}_{N_s} \end{bmatrix}, \quad \boldsymbol{y}_i = \begin{bmatrix} \sqrt{\|\boldsymbol{y}_{i_s}\|^2 - 1/K} \\ \boldsymbol{y}_{i_s} \end{bmatrix}. \qquad (3)$$

Hyperbolic Neural Networks

- As we have seen, many Euclidean operations are not well defined in hyperbolic spaces.
  - Vector addition, scalar multiplication, matrix multiplication
  - The results of such operations might no longer constraint in the hyperbolic space
- To define neural networks in the hyperbolic space, we need to ensure some constraints or it will lose the geometric properties

Hyperbolic Neural Networks
Hyperbolic Linear Layer

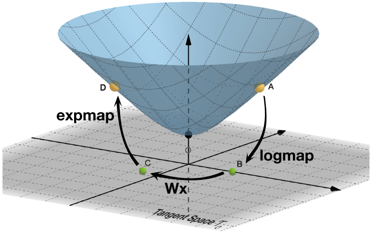
- Conventional HNN (Ganea et al., 2018)
  - Do it on the Tangent Space!
  - Suppose $\boldsymbol{x} \in \mathbb{L}_K^n$, $W \in \mathbb{R}^{(n+1)\times(m+1)}$
  - Hyperbolic matrix multiplication

$$W \otimes^K \boldsymbol{x} := \exp_{\boldsymbol{o}}^K(W \log_{\boldsymbol{o}}^K(\boldsymbol{x}))$$

  - Suppose $\boldsymbol{x} \in \mathbb{L}_K^n$, $\boldsymbol{b} \in \mathcal{T}_{\boldsymbol{o}}\mathbb{L}_K^n$
  - Hyperbolic bias addition

$$\boldsymbol{x} \oplus^K \boldsymbol{b} := \exp_{\boldsymbol{x}}^K(\mathrm{PT}_{\boldsymbol{o}\to\boldsymbol{x}}^K(\log_{\boldsymbol{o}}^K(\boldsymbol{x})))$$

- These operations are not defined natively in the Lorentz model.
- Is there any notion of "linear transformation" in Lorentz model?



Conventional HNN

Hyperbolic Neural Networks

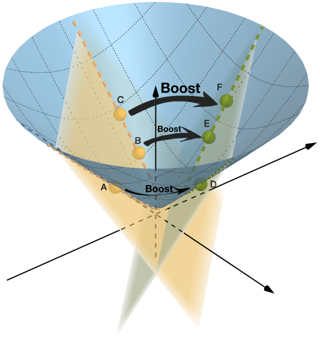
Hyperbolic Linear Layer

Definition 1: Lorentz Boost

Lorentz boost describes relative motion with constant velocity and without rotation of the spatial coordinate axes.
Given a velocity $\boldsymbol{v} \in \mathbb{R}^n$ (ratio to the speed of light), $\|\boldsymbol{v}\| < 1$ and $\gamma = \frac{1}{\sqrt{1-\|\boldsymbol{v}\|^2}}$, the Lorentz boost matrices are given by

$$\boldsymbol{B} = \begin{bmatrix} \gamma & -\gamma\boldsymbol{v}^\top \\ -\gamma\boldsymbol{v} & \boldsymbol{I} + \frac{\gamma^2}{1+\gamma}\boldsymbol{v}\boldsymbol{v}^\top \end{bmatrix}$$

Note that points on the intersection of a plane and the hyperboloid are still coplanar after the Lorentz boost.



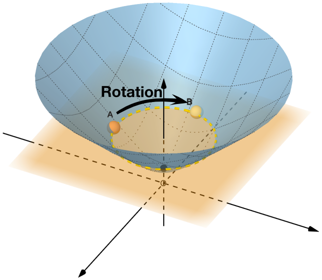
Lorentz Boost
Chen et al. (2021)

Definition 2: Lorentz Rotation

Lorentz rotation is the rotation of the spatial coordinates. The Lorentz rotation matrices are given by

$$\boldsymbol{R} = \begin{bmatrix} 1 & \mathbf{0}^\top \\ \mathbf{0} & \tilde{\boldsymbol{R}} \end{bmatrix},$$

where $\tilde{\boldsymbol{R}}^\top \tilde{\boldsymbol{R}} = \boldsymbol{I}$ and $\det(\tilde{\boldsymbol{R}}) = 1$, i.e., $\boldsymbol{R} \in \boldsymbol{SO}(n)$ is a special orthogonal matrix.



Lorentz Rotation
Chen et al. (2021)

Hyperbolic Neural Networks
Hyperbolic Linear Layer

- Both Lorentz boost and rotation are linear transformations in the Lorentz model. $\forall \boldsymbol{x} \in \mathbb{L}_K^n, \boldsymbol{B}\boldsymbol{x} \in \mathbb{L}_K^n, \boldsymbol{R}\boldsymbol{x} \in \mathbb{L}_K^n$
- How to define a general linear transformation $\mathbb{L}_K^n \to \mathbb{L}_K^m$?
- Consider $\boldsymbol{M} = \left[\begin{smallmatrix} \boldsymbol{v}^\top \\ \boldsymbol{W} \end{smallmatrix}\right]$, $\boldsymbol{v} \in \mathbb{R}^{n+1}$, $\boldsymbol{W} \in \mathbb{R}^{m\times(n+1)}$
- For $\boldsymbol{x} \in \mathbb{L}_K^n$, the linear transformation is $f_{\boldsymbol{x}}(\boldsymbol{M})\boldsymbol{x} \in \mathbb{L}_K^m$, and

$$f_{\boldsymbol{x}}(\boldsymbol{M}) = f_{\boldsymbol{x}}\left(\begin{bmatrix} \boldsymbol{v}^\top \\ \boldsymbol{W} \end{bmatrix}\right) = \begin{bmatrix} \frac{\sqrt{\|\boldsymbol{W}\boldsymbol{x}\|^2 - 1/K}}{\boldsymbol{v}^\top \boldsymbol{x}} \boldsymbol{v}^\top \\ \boldsymbol{W} \end{bmatrix}$$

- We could verify $\langle f_{\boldsymbol{x}}(\boldsymbol{M})\boldsymbol{x}, f_{\boldsymbol{x}}(\boldsymbol{M})\boldsymbol{x}\rangle_{\mathcal{L}} = 1/K$
- How to show the expressibility of such linear transformation?

Hyperbolic Neural Networks

Hyperbolic Linear Layer

Lemma 3

Given $\boldsymbol{x} \in \mathbb{L}_K^n$, we also denote the range of $f_{\boldsymbol{x}}(\boldsymbol{M})$ at $\boldsymbol{x}$ without changing the number of space dimension as $\mathcal{M}_{\boldsymbol{x}} = \left\{ f_{\boldsymbol{x}}(\boldsymbol{M}) \mid \boldsymbol{M} \in \mathbb{R}^{(n+1)\times(n+1)} \right\}$. Then $\mathcal{M}_{\mathrm{x}}$ covers all valid transformations in the Lorentz model.

Proof.

$\mathcal{A} = \{\boldsymbol{A} \in \mathbb{R}^{(n+1)\times(n+1)} | \forall \boldsymbol{x} \in \mathbb{L}_K^n : \langle \boldsymbol{A}\boldsymbol{x}, \boldsymbol{A}\boldsymbol{x}\rangle_{\mathcal{L}} = \frac{1}{K}, (\boldsymbol{A}\boldsymbol{x})_0 > 0\}$ is the set of all valid transformation matrices in the Lorentz model.

Given $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{v}_A^\top \\ \boldsymbol{W}_A \end{bmatrix} \in \mathcal{A}$, there $\exists \boldsymbol{v}^\top \boldsymbol{x} > 0$ and $\|\boldsymbol{W}_A \boldsymbol{x}\|^2 - \left(\boldsymbol{v}_A^\top \boldsymbol{x}\right)^2 = \frac{1}{K}$.

Furthermore, $\forall \boldsymbol{A} \in \mathcal{A}$, we have

$f_{\boldsymbol{x}}(\boldsymbol{A}) = f_{\boldsymbol{x}}\left(\begin{bmatrix} \boldsymbol{v}_A^\top \\ \boldsymbol{W}_A \end{bmatrix}\right) = \begin{bmatrix} \frac{\sqrt{\|\boldsymbol{W}_A \boldsymbol{x}\|^2 - 1/K}}{\boldsymbol{v}_A^\top \boldsymbol{x}} \boldsymbol{v}_A^\top \\ \boldsymbol{W}_A \end{bmatrix} = \boldsymbol{A}.$

Hence, we get $\mathcal{A} \subseteq \mathcal{M}_{\boldsymbol{x}}$. $\square$

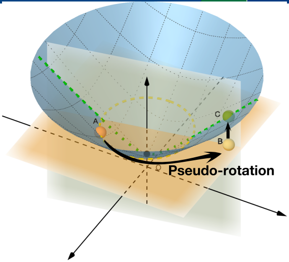
Corollary Both Lorentz boost and rotation can be covered by such linear transformation.

Hyperbolic Linear Layer

- What is the connection between conventional HNN and the new one?
- We further define the Lorentz pseudo rotation
  - At point $p \in \mathbb{L}_K^n$, all matrices for pseudo-rotation are collected by the set

$$\mathcal{P}_{\boldsymbol{x}} = \left\{ f_{\boldsymbol{x}} \left(\left[\begin{smallmatrix} w & 0^\top \\ 0 & \boldsymbol{W} \end{smallmatrix} \right] \right) \mid w \in \mathbb{R}, \boldsymbol{W} \in \mathbb{R}^{n \times n} \right\}$$

  - A relaxation of the Lorentz rotation
- The conventional HNN with input $\boldsymbol{x} \in \mathbb{L}_K^n$ is



Lorentz Pseudo Rotation
Chen et al. (2021)

$$\exp_{\mathbf{0}}\left(\left[\begin{array}{cc} * & 0^\top \\ \mathbf{0} & \boldsymbol{W} \end{array}\right] \log_{\mathbf{0}}\left(\left[\begin{array}{c} x_t \\ \boldsymbol{x}_s \end{array}\right]\right)\right) = \left[\begin{array}{cc} \frac{\cosh(\beta)}{\sqrt{-K x_t}} & \mathbf{0}^\top \\ \mathbf{0} & \frac{\sinh(\beta)}{\sqrt{-K}\|\boldsymbol{W}\boldsymbol{x}_s\|} \end{array}\right] \left[\begin{array}{c} x_t \\ \boldsymbol{x}_s \end{array}\right]$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times n}, \beta = \frac{\sqrt{-K} \cosh^{-1}\left(\sqrt{-K} x_t\right)}{\sqrt{-K x_t^2 - 1}} \|\boldsymbol{W}\boldsymbol{x}_s\|$

Hyperbolic Neural Networks

Hyperbolic Linear Layer

Lemma 4

$\forall \boldsymbol{x} \in \mathbb{L}_K^n, \mathcal{H}_{\boldsymbol{x}} = \left\{ \begin{bmatrix} \frac{\cosh(\beta)}{\sqrt{-Kx_t}} & \mathbf{0}^\top \\ \mathbf{0} & \frac{\sinh(\beta)}{\sqrt{-K}\|\boldsymbol{W}\boldsymbol{x}_s\|} \end{bmatrix}, \boldsymbol{W} \in \mathbb{R}^{n\times n}, \beta = \frac{\sqrt{-K}\cosh^{-1}\left(\sqrt{-K}x_t\right)}{\sqrt{-Kx_t^2-1}} \|\boldsymbol{W}\boldsymbol{x}_s\| \right\}$, we have $\mathcal{H}_{\boldsymbol{x}} \subseteq \mathcal{P}_{\boldsymbol{x}}$ and $\mathcal{H}_{\boldsymbol{x}} \cap \mathcal{B} = \{\boldsymbol{I}\}$

Proof.

For any $\boldsymbol{H} \in \mathcal{H}_{\boldsymbol{x}}, \boldsymbol{H}$ has the form $\begin{bmatrix} w & 0^\top \\ \mathbf{0} & \boldsymbol{W} \end{bmatrix}$, satisfying $\|\boldsymbol{W}\boldsymbol{x}_s\|^2 - (wx_t)^2 = \frac{1}{K}$ and $wx_t > 0$.

We can get $f_{\boldsymbol{x}}(\boldsymbol{H}) = f_{\boldsymbol{x}}\left(\begin{bmatrix} w & 0^\top \\ 0 & \boldsymbol{W} \end{bmatrix}\right) = \begin{bmatrix} \frac{\sqrt{\|\boldsymbol{W_s}\|^2-1/K}}{wx_t} & 0 \\ 0 & \mathbf{0}^\top \end{bmatrix} = \boldsymbol{H}$.

Hence, $\forall \boldsymbol{x} \in \mathbb{L}_K^n, \forall \boldsymbol{H} \in \mathcal{H}_{\boldsymbol{x}}$, then $\boldsymbol{H} = f_{\boldsymbol{x}}(\boldsymbol{H}) \in \mathcal{P}_{\boldsymbol{x}}$, and $\mathcal{H}_{\boldsymbol{x}} \subseteq \mathcal{P}_{\boldsymbol{x}}$ $\square$

- Therefore, a conventional HNN is a special rotation where the time axis is changed to stay in the Lorentz model.
- The new linear layer also have boost, more expressive.
- Based on this linear transformation, a hyperbolic linear layer with activation, bias and normalization is defined to be

$$\boldsymbol{y} = \mathrm{HLinear}_{n,m}(\boldsymbol{x}) = \begin{bmatrix} \sqrt{\|h(\boldsymbol{W}\boldsymbol{x}, \boldsymbol{v})\|^2 - 1/K} \\ h(\boldsymbol{W}\boldsymbol{x}, \boldsymbol{v}) \end{bmatrix}.$$

Here $\boldsymbol{x} \in \mathbb{L}_K^n$ is the input of the layer and $h(\boldsymbol{W}\boldsymbol{x}, \boldsymbol{v}) = \frac{\lambda\sigma\left(\boldsymbol{v}^\top \boldsymbol{x} + b'\right)}{\|\boldsymbol{W}\tau(\boldsymbol{x}) + \boldsymbol{b}\|}(\boldsymbol{W}\tau(\boldsymbol{x}) + \boldsymbol{b})$, where $\boldsymbol{v} \in \mathbb{R}^{n+1}$ and $\boldsymbol{W} \in \mathbb{R}^{m\times(n+1)}$ are trainable weights, $\boldsymbol{b}$ and b' are trainable biases, σ is the sigmoid function, τ is the activation function, and the trainable parameter $\lambda > 0$ scales the range.

- Graph neural networks mainly consists of feature transformation and node aggregation.
- Feature transformation $\rightarrow$ Hyperbolic linear layer
- Node aggregation $\rightarrow$ Centroid point Eq. (1)
- We have

$$\begin{aligned}\boldsymbol{x}_v^{(l)} &= \mathrm{HGCN}(\boldsymbol{X}^{(l-1)})_v \\ &= \mathrm{HCent}(\{\mathrm{HLinear}_{d_{l-1},d_l}(\boldsymbol{x}_u^{(l-1)}) \mid u \in N(v)\}, \mathbf{1})\end{aligned}$$

where $\boldsymbol{x}_v^{(l)}$ is the feature of node v in layer l, d_l denotes the dimensionality of layer l, and $N(v)$ is the set of neighbour point of node v.

Hyperbolic Neural Networks
Hyperbolic Centroid Distance Layer

- Since the input and output space are often in $\mathbb{R}^n$, we need to find a mapping between $\mathbb{L}_K^n$ and $\mathbb{R}^n$.
- Hyperbolic centroid distance layer $\mathbb{L}_K^n \to \mathbb{R}^m$ (Liu et al., 2019)
- It first initializes m trainable centroids $\{\boldsymbol{c}_i\}_{i=1}^m \subset \mathbb{L}_K^n$
- Given an input $\boldsymbol{x} \in \mathbb{L}_K^n$, it produces a distance vector

$$\boldsymbol{y} = \mathrm{HCDist}_{n,m}(\boldsymbol{x}) = \begin{bmatrix} d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{c}_1) & \cdots & d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{c}_m) \end{bmatrix}^\top \in \mathbb{R}^m$$

- This method is more flexible and numerically stable than directly logarithmic map.

Hyperbolic Neural Networks
Hyperbolic Embedding Layer

- No more fancy layers for $\mathbb{R}^m \to \mathbb{L}_K^m$. Just use exp map.

$$\boldsymbol{y} = \mathrm{E2H}_{n,m}(\boldsymbol{t}) = \exp_{\boldsymbol{o}}^K \left(\begin{bmatrix} 0 \\ \boldsymbol{t} \end{bmatrix} \right)$$

  where $\boldsymbol{o} = \left[\sqrt{-1/K}, 0, \ldots, 0\right]^\top$ is the hyperbolic origin.

- To embed an one-hot vector, we first map the input to a hidden embedding $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^m$ with a trainable embedding matrix $\boldsymbol{W} \in \mathbb{R}^{n\times m}$

$$\boldsymbol{y} = \mathrm{HEmbed}_{n,m}(\boldsymbol{x}) = \mathrm{E2H}\left(\boldsymbol{W}\boldsymbol{x}\right).$$

- No more fancy layers for $\mathbb{R}^m \to \mathbb{L}_K^m$. Just use exp map.

$$\boldsymbol{y} = \mathrm{E2H}_{n,m}(\boldsymbol{t}) = \exp_{\boldsymbol{o}}^K \left(\begin{bmatrix} 0 \\ \boldsymbol{t} \end{bmatrix} \right)$$

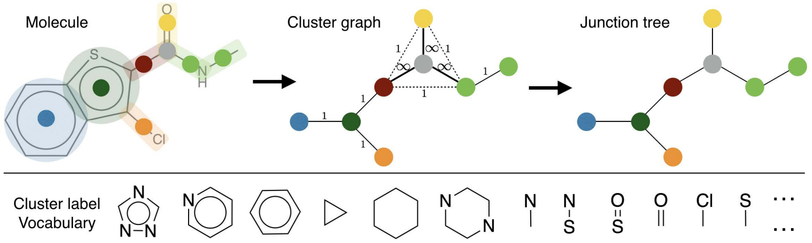
  where $\boldsymbol{o} = \left[\sqrt{-1/K}, 0, \ldots, 0 \right]^\top$ is the hyperbolic origin.

- To embed an one-hot vector, we first map the input to a hidden embedding $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} \in \mathbb{R}^m$ with a trainable embedding matrix $\boldsymbol{W} \in \mathbb{R}^{n \times m}$

$$\boldsymbol{y} = \mathrm{HEmbed}_{n,m}(\boldsymbol{x}) = \mathrm{E2H}\left(\boldsymbol{W}\boldsymbol{x}\right).$$

Applications
Molecular Generation

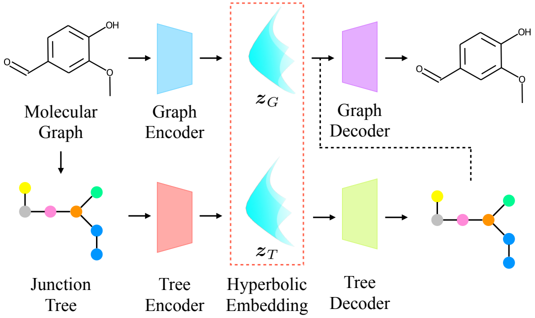
- Molecular generation (drug discovery) is a good playground for hyperbolic neural networks
  - Drugs are hierarchical (ATC Tree) Yu et al. (2020)
  - Molecular graphs are hierarchical
- Each molecular graph could be decomposed into a junction tree (Jin et al., 2018)



Junction tree decomposition (Jin et al., 2018)

Applications
HJTAE

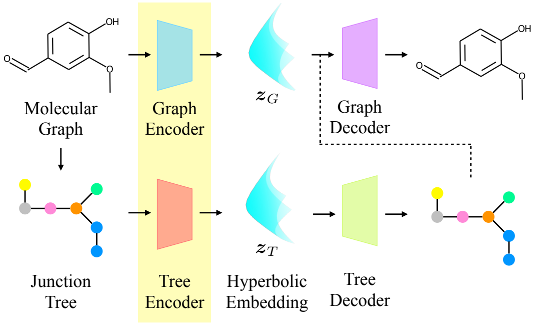
- We proposed an fully hyperbolic encoder-decoder for molecular graphs using the Junction Tree AE framework
  - Fully hyperbolic: no operations on tangent spaces



Hyperbolic Junction Tree Encoder-Decoder

Hyperbolic JT Encoder-Decoder
Encoder

- Graph encoder: Encode molecular graphs to hyperbolic embedding $\boldsymbol{z}_G$
- Tree encoder: Encode junction tree to hyperbolic embedding $\boldsymbol{z}_T$



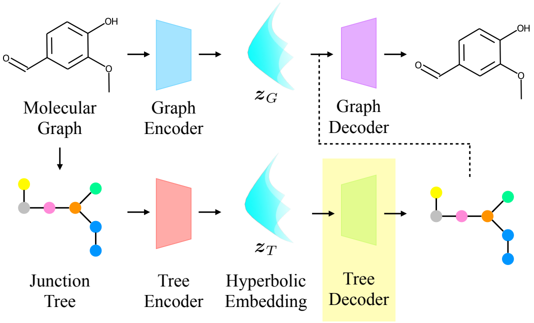
Graph and Tree encoder

Hyperbolic JT Encoder-Decoder
Encoder

- Graph encoder
  - Map node feature $\boldsymbol{x}_v$ (atom type, valence, etc.) to the hyperbolic space
  $$\boldsymbol{x}_v^{(0)} = \mathrm{E2H}_{d_{G_0}, d_G}(\boldsymbol{x}_v)$$
  - Pass to a hyperbolic GCN with l_G layers
  $$\boldsymbol{x}^{(l)} = \mathrm{HGCN}(\boldsymbol{x}^{(l-1)}), \quad l = 1, \cdots, l_G.$$
  - Take the centroid of the embeddings of all vertices to get $\boldsymbol{z}_G$
  $$\boldsymbol{z}_G = \mathrm{HCent}(\boldsymbol{x}^{(l_G)}).$$
- Tree encoder
  - Basically the same, except that the input tree node features are one-hot vectors of cluster type
  $$\begin{aligned} \boldsymbol{x}_i^{(0)} &= \mathrm{HEmbed}_{d_{T_0}, d_T}(\boldsymbol{x}_i), \\ \boldsymbol{x}^{(l)} &= \mathrm{HGCN}(\boldsymbol{x}^{(l-1)}), \quad l = 1, \cdots, l_T, \\ \boldsymbol{z}_T &= \mathrm{HCent}(\boldsymbol{x}^{(l_T)}). \end{aligned}$$

Hyperbolic JT Encoder-Decoder
Tree Decoder

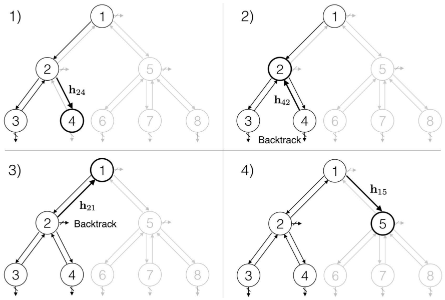
- Tree decoder: Decode a junction tree from $\boldsymbol{z}_T$



Tree decoder

Hyperbolic JT Encoder-Decoder
Tree Decoder

- Tree decoder: Decode the junction tree autoregressively.
- Consider a depth first search of the junction tree
- At each step
  - Massage passing
  - Topological prediction
  - Label prediction



DFS of Junction Tree
Jin et al. (2018)

Hyperbolic JT Encoder-Decoder
Tree Decoder - Massage Passing

- Let $\tilde{E} = \{(i_1, j_1), \ldots, (i_m, j_m)\}$ be the collection of the edges visited in a DFS, $\tilde{E}_t$ be the set of the first t edges in $\tilde{E}$.
- We store a message $\boldsymbol{h}_{i_t,j_t}$ for each edge in $\tilde{E}$
- The $\boldsymbol{h}_{i_t,j_t}$ is updated by the node feature $\boldsymbol{x}_{i_t}$ and inward messages $\boldsymbol{h}_{k,i_t}$
  - Use hyperbolic centroid to gather the inward messages
  $$\boldsymbol{z}_{\text{nei}} = \mathrm{HCent}(\mathrm{HLinear}_{d_T,d_T}(\{\boldsymbol{h}_{k,i_t}\}_{(k,i_t)\in\tilde{E},k\neq j_t})),$$
  - Map the tree node features to the hyperbolic space
  $$\boldsymbol{z}_{\text{cur}} = \mathrm{HEmbed}_{d_{T_0},d_T}(\boldsymbol{x}_{i_t}).$$
  - Combine them using the Lorentz Direct Concatenation and pass them through a hyperbolic linear layer
  $$\boldsymbol{h}_{i_t,j_t} = \mathrm{HLinear}_{2\times d_T,d_T}\left(\mathrm{HCat}(\{\boldsymbol{z}_{\text{cur}}, \boldsymbol{z}_{\text{nei}}\})\right).$$

Hyperbolic JT Encoder-Decoder
Tree Decoder - Topological Prediction

- At each time step t, the model make a decision on whether to generate a child node
- It uses tree embedding $\boldsymbol{z}_T$, node feature $\boldsymbol{x}_{i_t}$, and inward messages $\boldsymbol{h}_{k,i_t}$ in the following way

$$\begin{aligned}
\boldsymbol{z}_{\text{nei}} &= \text{HCent}(\text{HLinear}_{d_T,d_T}(\{\boldsymbol{h}_{k,i_t}\}_{(k,i_t)\in\tilde{E}})),\\
\boldsymbol{z}_{\text{cur}} &= \text{HEmbed}_{d_{T_0},d_T}(\boldsymbol{x}_{i_t}),\\
\boldsymbol{z}_{\text{all}} &= \text{HLinear}_{3\times d_T,d_T}\left(\text{HCat}(\{\boldsymbol{z}_{\text{cur}},\boldsymbol{z}_{\text{nei}},\boldsymbol{z}_T\})\right),\\
\boldsymbol{p}_t &= \text{Softmax}(\text{HCDist}_{d_T,2}(\boldsymbol{z}_{\text{all}})).
\end{aligned}$$

Hyperbolic JT Encoder-Decoder
Tree Decoder - Label Prediction

- If a child node j_t is generated, we use the tree embedding $\boldsymbol{z}_T$ and the outward message $\boldsymbol{h}_{i_t,j_t}$ to predict its label

$$\begin{aligned}\boldsymbol{z}_{\text{all}} &= \text{HLinear}_{2\times d_T, d_T}\left(\text{HCat}(\{\boldsymbol{h}_{i_t,j_t}, \boldsymbol{z}_T\})\right)\\ \boldsymbol{q}_t &= \text{Softmax}(\text{HCDist}_{d_T, d_{T_0}}(\boldsymbol{z}_{\text{all}})).\end{aligned}$$

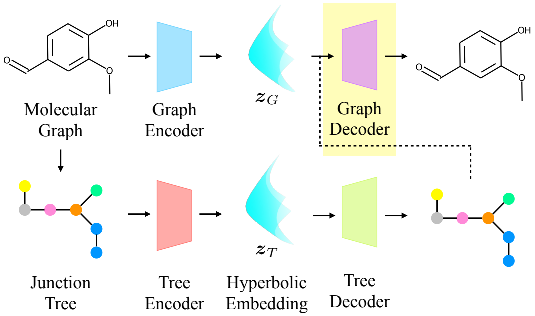
- These two prediction are trained with the cross-entropy loss

$$L_{\text{topo}} = \sum_{t=1}^{m} L_{\text{cross}}(\hat{\boldsymbol{p}}_t, \boldsymbol{p}_t),\ \ L_{\text{label}} = \sum_{t=1}^{m} L_{\text{cross}}(\hat{\boldsymbol{q}}_t, \boldsymbol{q}_t)$$

- In training, we use the teacher forcing

Hyperbolic JT Encoder-Decoder
Graph Decoder

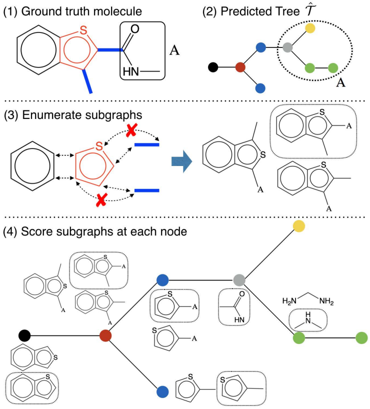
- Graph decoder: Decode a molecular graph from z_G and a junction tree



Graph decoder

Hyperbolic JT Encoder-Decoder
Graph Decoder

- Junction tree and molecular graph is not one-to-one mapping
  - Exists multiple ways of attaching clusters
  - Let $\mathcal{G}_i$ be the set of possible candidate subgraphs around tree node i. We need a scoring function for each candidate subgraph $G_j^{(i)} \in \mathcal{G}_i$
  - We want to use "graph encoder" to get $z_{G_j^{(i)}}$ of each subgraph $G_j^{(i)}$
  - Some nodes do not exist in the subgraph
  - The neighbor of i could have open attachment points
  - We connect these points with virtual node from junction tree, whose features are acquired from the final layer of the Tree Encoder.



(1) Ground truth molecule
(2) Predicted Tree $\hat{\mathcal{T}}$
(3) Enumerate subgraphs
(4) Score subgraphs at each node
S
O
HN
A
H_2N
NH_2

Jin et al. (2018)

Hyperbolic JT Encoder-Decoder
Graph Decoder

- We use "graph encoder" to get $\boldsymbol{z}_{G_j^{(i)}}$ of each subgraph $G_j^{(i)}$

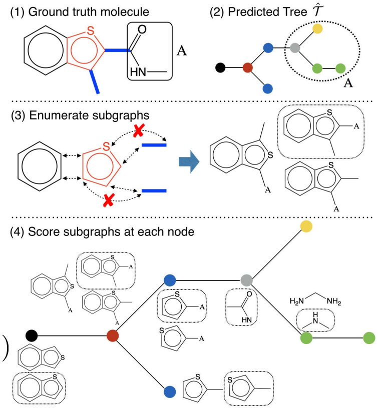
$$\begin{aligned}\boldsymbol{x}_v^{(0)} &= \mathrm{E2H}_{d_{G_0},d_G}(\boldsymbol{x}_v),\\ \boldsymbol{x}^{(l)} &= \mathrm{HGCN}(\boldsymbol{x}^{(l-1)}), \quad l = 1, \cdots, l_G,\\ \boldsymbol{z}_{G_j^{(i)}} &= \mathrm{HCent}(\boldsymbol{x}^{(l_G)}).\end{aligned}$$

- This embeddings is combined with $\boldsymbol{z}_G$ by the Lorentz Direct Concatenation

$$\boldsymbol{z}_{\mathrm{all}} = \mathrm{Hlinear}_{2\times d_G,d_G}(\mathrm{HCat}(\{\boldsymbol{z}_{G_j^{(i)}}, \boldsymbol{z}_G\}))$$

- It is passed to the hyperbolic centroid distance layer to get a score

$$s_j^{(i)} = \mathrm{HCDist}_{d_G,1}(\boldsymbol{z}_{\mathrm{all}}) \in \mathbb{R}.$$


(1) Ground truth molecule
(2) Predicted Tree $\hat{\mathcal{T}}$
(3) Enumerate subgraphs
(4) Score subgraphs at each node
A
S
O
HN

Jin et al. (2018)

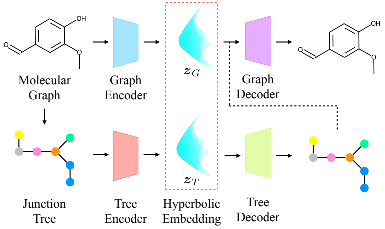
Hyperbolic JT Encoder-Decoder
Graph Decoder

- The loss is the sum of the cross-entropy losses in each $\mathcal{G}_i$, suppose the correct subgraph is $G_c^{(i)}$,

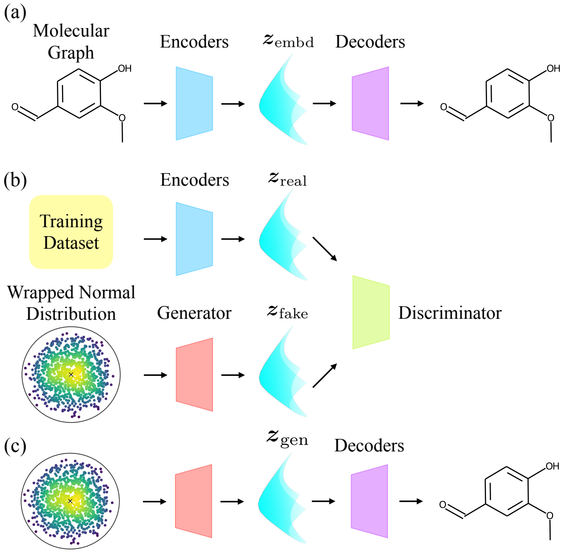
$$L_{\text{assm}} = \sum_i \left(s_c^{(i)} - \log \sum_{G_j^{(i)} \in \mathcal{G}_i} \exp\left(s_j^{(i)}\right) \right)$$

- We also use teacher forcing for training.

- This is all for Hyperbolic Junction Tree Encoder-Decoder!



Hyperbolic GAN
Architecture



Hyperbolic GAN
Architecture

- Generator
  - Map a wrapped normal dist $\mathcal{G}(\boldsymbol{o}, \boldsymbol{I})$ to a hyperbolic distribution
  - We sample $\boldsymbol{z}^{(0)} \sim \mathcal{G}(\boldsymbol{o}, \boldsymbol{I})$ then

$$\begin{aligned} \boldsymbol{z}^{(l)} &= \mathrm{HLinear}_{d_{l-1},d_l}(\boldsymbol{z}^{(l-1)}), \quad l = 1, \cdots, l_{\mathrm{gen}}, \\ \boldsymbol{z}_{\mathrm{fake}} &= \boldsymbol{z}^{(l_{\mathrm{gen}})}. \end{aligned}$$

- Discriminator
  - Distinguish between fake and real data, output is a score in $\mathbb{R}$

$$\begin{aligned} \boldsymbol{z}^{(l)} &= \mathrm{HLinear}_{d_{l-1},d_l}(\boldsymbol{z}^{(l-1)}), \quad l = 1, \cdots, l_{\mathrm{dis}}, \\ s &= \mathrm{HCDist}_{d_{l_{\mathrm{dis}}},1}(\boldsymbol{z}^{(l_{\mathrm{dis}})}). \end{aligned}$$

Hyperbolic GAN
Training

- We use Wasserstein GAN framework Arjovsky et al. (2017)
- It minimize Wasserstein-1 (W_1) distance between the generator distribution $\mathbb{P}_g$ and data distribution $\mathbb{P}_r$

$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(\boldsymbol{x}, \boldsymbol{y}) \sim \gamma}[d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})],$$

- By Kantorovich-Rubinstein duality Villani (2009)

$$W_1(\mathbb{P}_g, \mathbb{P}_r) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[f(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[f(\boldsymbol{x})],$$

  where the sup is over all 1-Lipschitz functions $f : \mathbb{L}_K^n \to \mathbb{R}$.

  - A function f is K-Lipschitz continuous there exists a real constant $K \geq 0$ s. t., for all $\boldsymbol{x}_1$, $\boldsymbol{x}_2 \in \mathbb{L}_K^n$

$$\frac{|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)|}{d_{\mathcal{L}}(\boldsymbol{x}_1, \boldsymbol{x}_2)} \leq K$$

Hyperbolic GAN
Training

- To enforce the 1-Lipschitz constraint, we adopt gradient penalty by Gulrajani et al. (2017). The loss function is

$$L_{\mathrm{WGAN}} = \underset{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}{\mathbb{E}}[D(\tilde{\boldsymbol{x}})] - \underset{\boldsymbol{x}\sim\mathbb{P}_r}{\mathbb{E}}[D(\boldsymbol{x})] + \lambda \underset{\hat{\boldsymbol{x}}\sim\mathbb{P}_{\hat{x}}}{\mathbb{E}}\left[(\|\nabla D(\hat{\boldsymbol{x}})\|_{\mathcal{L}} - 1)^2\right],$$

  where $\nabla D(\hat{\boldsymbol{x}})$ is the Riemannian gradient of $D(\boldsymbol{x})$ at $\hat{\boldsymbol{x}}$, $\mathbb{P}_{\hat{\boldsymbol{x}}}$ samples uniformly along the geodesic between pairs of points sampled from $\mathbb{P}_g$ and $\mathbb{P}_r$.
- This is because the norm of the Riemannian gradient of the optimal discriminator evaluated at the points is equal to 1 almost surely.
- Consider the following proposition

Hyperbolic GAN
Training

Proposition 5

Let $\mathbb{P}_r$ and $\mathbb{P}_g$ be two distribution in the compact space $\mathbb{L}_K^n$. Let f^* be an optimal solution of

$$\max_{\|f\|_L \leq 1} \mathbb{E}_{\boldsymbol{y} \sim \mathbb{P}_r}[f(\boldsymbol{y})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_g}[f(\boldsymbol{x})],$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{L}_K^n$. Let π be the optimal coupling between $\mathbb{P}_r$ and $\mathbb{P}_g$ that minimizes

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\pi \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x, y \sim \pi)}[d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})],$$

Let $\gamma(t), 0 \leq t \leq 1$ be the geodesic between $\boldsymbol{x}$ and $\boldsymbol{y}$, such that

$$\gamma(0) = \boldsymbol{x}, \gamma(1) = \boldsymbol{y}, \gamma'(t) = \boldsymbol{v}_t, \|\boldsymbol{v}_t\|_{\mathcal{L}} = d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})$$

where $\boldsymbol{v}_t \in \mathcal{T}\mathbb{L}_K^n$. Let $\boldsymbol{x}_t = \gamma(t)$. If f^* is differentiable and $\pi(\boldsymbol{x} = \boldsymbol{y}) = 0$, then it holds that

$$\mathbb{P}_{(\boldsymbol{x}, \boldsymbol{y}) \sim \pi}\left[\nabla f^*(\boldsymbol{x}_t) = \frac{\boldsymbol{v}_t}{d_{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{y})}\right] = 1$$

Hyperbolic GAN
Training

Proof. For the optimal solution f^*, we have

$$\mathbb{P}_{(\boldsymbol{x},\boldsymbol{y})\sim\pi}\left[f^*(\boldsymbol{y}) - f^*(\boldsymbol{x}) = d_{\mathcal{L}}(\boldsymbol{y},\boldsymbol{x})\right] = 1.$$

Let $\psi(t) = f^*\left(\boldsymbol{x}_t\right) - f^*(\boldsymbol{x})$, $0 \le t, t' \le 1$. By Gulrajani et al. (2017), it is clear that ψ is $d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})$-Lipschitz, and
$f^*(\boldsymbol{x}_t) - f^*(\boldsymbol{x}) = \psi(t) = t d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})$,
$f^*(\boldsymbol{x}_t) = f^*(\boldsymbol{x}) + t d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y}) = f^*(\boldsymbol{x}) + t\|\boldsymbol{v}_t\|_{\mathcal{L}}$.

Let $\boldsymbol{u}_t = \frac{\boldsymbol{v}_t}{d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})} \in \mathcal{T}\mathbb{L}_K^n$ be the unit speed directional vector of the geodesic at point $\boldsymbol{x}_t$. Let $\alpha : [-1,1] \to \mathbb{L}_K^n$ be a differentiable curve with $\alpha(0) = \boldsymbol{x}_t$ and $\alpha'(0) = \boldsymbol{u}_t$. Note that $\gamma'(t) = d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})\alpha'(0)$.
Therefore,

$$\lim_{h\to 0}\alpha(h) = \lim_{h\to 0}\gamma\left(t + \frac{h}{d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})}\right) = \lim_{h\to 0}\boldsymbol{x}_{t+\frac{h}{d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})}}$$

Hyperbolic GAN (cont.)
Training

The directional derivative can be thus calculated as

$$\begin{aligned}
\nabla_{\boldsymbol{u}_t} f^*\left(\boldsymbol{x}_t\right) &= \left.\frac{d}{d\tau} f^*(\alpha(\tau))\right|_{\tau=0} = \lim_{h \to 0} \frac{f^*\left(\alpha(h)\right) - f^*\left(\alpha(0)\right)}{h} \\
&= \lim_{h \to 0} \frac{f^*\left(\boldsymbol{x}_{t+\frac{h}{d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})}}\right) - f^*\left(\boldsymbol{x}_t\right)}{h} \\
&= \lim_{h \to 0} \frac{f^*\left(\boldsymbol{x}\right) + (t + \frac{h}{d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})}) d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y}) - f^*\left(\boldsymbol{x}\right) - t d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})}{h} = 1.
\end{aligned}$$

Since f^* is 1-Lipschitz, we have $\|\nabla f^*(\boldsymbol{x}_t)\|_{\mathcal{L}} \leq 1$. This implies

$$\begin{aligned}
1 &\geq \|\nabla f^*(\boldsymbol{x})\|_{\mathcal{L}}^2 \\
&= \left\langle \boldsymbol{u}_t, \nabla f^*\left(\boldsymbol{x}_t\right)\right\rangle_{\mathcal{L}}^2 + \left\|\nabla f^*\left(\boldsymbol{x}_t\right) - \left\langle \boldsymbol{u}_t, \nabla f^*\left(\boldsymbol{x}_t\right)\right\rangle \boldsymbol{u}_t\right\|_{\mathcal{L}}^2 \\
&= \left|\nabla_{\boldsymbol{u}_t} f^*\left(\boldsymbol{x}_t\right)\right|^2 + \left\|\nabla f^*\left(\boldsymbol{x}_t\right) - \boldsymbol{u}_t \nabla_{\boldsymbol{u}_t} f^*\left(\boldsymbol{x}_t\right)\right\|_{\mathcal{L}}^2 \\
&= 1 + \left\|\nabla f^*\left(\boldsymbol{x}_t\right) - \boldsymbol{u}_t\right\|_{\mathcal{L}}^2 \geq 1.
\end{aligned}$$

Therefore, we have $1 = 1 + \left\|\nabla f^*\left(\boldsymbol{x}_t\right) - \boldsymbol{u}_t\right\|_{\mathcal{L}}^2$, $\nabla f^*\left(\boldsymbol{x}_t\right) = \boldsymbol{u}_t$. This yields $\nabla f^*\left(\boldsymbol{x}_t\right) = \frac{\boldsymbol{v}_t}{d_{\mathcal{L}}(\boldsymbol{x},\boldsymbol{y})}$.

Experiment

- Dataset
  - MOSES benchmarking platform (Polykovskiy et al., 2020)
  - 1.58M training, 176k test, and 176k scaffold test molecules.
  - scaffold test set $\rightarrow$ different Bemis-Murcko scaffolds
- Metric
  - *Validity* and *Unique*(ness) $\rightarrow$ % of valid and unique molecules
  - Internal diversity (*IntDiv*) $\rightarrow$ chemical diversity (mode collapse)
  - *Filters* $\rightarrow$ % of molecules that passed the filter applied on ZINC
  - *Novelty* $\rightarrow$ % of molecules that are not in training set
  - Fréchet ChemNet Distance (*FCD*) $\rightarrow$ diff. in dist. of ChemNet
  - Similarity to a Nearest Neighbor (*SNN*) $\rightarrow$ avg. similarly between generation and its nearest neighbor in reference set
  - Fragment similarity (*Frag*) and Scaffold similarity (*Scaf*) $\rightarrow$ cosine distances between fragment or scaffold frequency vectors

Experiment

Table: Performance of different models in Valid, Unique, IntDiv, Filters, and Novelty metrics. Reported (mean ± std) over three independent samples.

Model	Valid (↑)	Unique@1k (↑)	Unique@10k (↑)	IntDiv (↑)	IntDiv2 (↑)	Filters (↑)	Novelty (↑)
Train	*1*	*1*	*1*	*0.857*	*0.851*	*1*	*1*
HMM	$0.076_{\pm}0.0322$	$0.623_{\pm}0.1224$	$0.567_{\pm}0.1424$	$0.847_{\pm}0.0403$	$0.810_{\pm}0.0507$	$0.902_{\pm}0.0489$	$\mathbf{0.999_{\pm}0.001}$
NGram	$0.238_{\pm}0.0025$	$0.974_{\pm}0.0108$	$0.922_{\pm}0.0019$	$\mathbf{0.874_{\pm}0.0002}$	$0.864_{\pm}0.0002$	$0.958_{\pm}0.001$	$0.969_{\pm}0.001$
Combinatorial	$\mathbf{1.0_{\pm}0.0}$	$0.998_{\pm}0.0015$	$0.991_{\pm}0.0009$	$0.873_{\pm}0.0002$	$\mathbf{0.867_{\pm}0.0002}$	$0.956_{\pm}0.0018$	$0.988_{\pm}0.0008$
CharRNN	$0.975_{\pm}0.0264$	$\mathbf{1.0_{\pm}0.0}$	$0.999_{\pm}0.0003$	$0.856_{\pm}0.0005$	$0.850_{\pm}0.0005$	$0.994_{\pm}0.0034$	$0.842_{\pm}0.0509$
AAE	$0.937_{\pm}0.0341$	$\mathbf{1.0_{\pm}0.0}$	$0.997_{\pm}0.002$	$0.856_{\pm}0.0031$	$0.850_{\pm}0.003$	$0.996_{\pm}0.0006$	$0.793_{\pm}0.0285$
VAE	$0.977_{\pm}0.0012$	$\mathbf{1.0_{\pm}0.0}$	$0.998_{\pm}0.0005$	$0.856_{\pm}0.0004$	$0.850_{\pm}0.0004$	$\mathbf{0.997_{\pm}0.0002}$	$0.695_{\pm}0.0069$
LatentGAN	$0.897_{\pm}0.0029$	$\mathbf{1.0_{\pm}0.0}$	$0.997_{\pm}0.0002$	$0.857_{\pm}0.0007$	$0.851_{\pm}0.0006$	$0.974_{\pm}0.0006$	$0.914_{\pm}0.0058$
JTVAE	$\mathbf{1.0_{\pm}0.0}$	$\mathbf{1.0_{\pm}0.0}$	$\mathbf{1.0_{\pm}0.0}$	$0.855_{\pm}0.0034$	$0.849_{\pm}0.0035$	$0.976_{\pm}0.0016$	$0.950_{\pm}0.0006$
HJTGAN (Ours)	$\mathbf{1.0_{\pm}0.0}$	$\mathbf{1.0_{\pm}0.0}$	$\mathbf{1.0_{\pm}0.0}$	$0.840_{\pm}0.0012$	$0.833_{\pm}0.0015$	$0.987_{\pm}0.0009$	$0.905_{\pm}0.0063$

Table: Performance of different models in FCD, SNN, Frag, and Scaf metrics. Reported (mean ± std) over three independent samples.

	FCD (↓)		SNN (↑)		Frag (↑)		Scaf (↑)	
Model	Test	TestSF	Test	TestSF	Test	TestSF	Test	TestSF
Train	*0.008*	*0.476*	*0.642*	*0.586*	*1*	*0.999*	*0.991*	*0*
HMM	$24.466_{\pm}2.5251$	$25.431_{\pm}2.5599$	$0.388_{\pm}0.0107$	$0.380_{\pm}0.0107$	$0.575_{\pm}0.1224$	$0.568_{\pm}0.1218$	$0.207_{\pm}0.0481$	$0.049_{\pm}0.018$
NGram	$5.507_{\pm}0.1027$	$6.231_{\pm}0.0966$	$0.521_{\pm}0.001$	$0.500_{\pm}0.0005$	$0.985_{\pm}0.0012$	$0.982_{\pm}0.0012$	$0.530_{\pm}0.0163$	$0.098_{\pm}0.0142$
Combinatorial	$4.238_{\pm}0.037$	$4.511_{\pm}0.0274$	$0.451_{\pm}0.0003$	$0.439_{\pm}0.0002$	$0.991_{\pm}0.0004$	$0.990_{\pm}0.0003$	$0.445_{\pm}0.0056$	$0.087_{\pm}0.0027$
CharRNN	$\mathbf{0.073_{\pm}0.0247}$	$\mathbf{0.520_{\pm}0.0379}$	$0.602_{\pm}0.0206$	$0.565_{\pm}0.0142$	$\mathbf{1.0_{\pm}0.0002}$	$0.998_{\pm}0.0003$	$0.924_{\pm}0.0058$	$0.110_{\pm}0.0081$
AAE	$0.556_{\pm}0.2033$	$1.057_{\pm}0.2375$	$0.608_{\pm}0.0043$	$0.568_{\pm}0.0045$	$0.991_{\pm}0.0051$	$0.991_{\pm}0.0039$	$0.902_{\pm}0.0375$	$0.079_{\pm}0.009$
VAE	$0.099_{\pm}0.0125$	$0.567_{\pm}0.0338$	$0.626_{\pm}0.0005$	$0.578_{\pm}0.0008$	$0.999_{\pm}0.0001$	$\mathbf{0.998_{\pm}0.0003}$	$\mathbf{0.939_{\pm}0.0021}$	$0.059_{\pm}0.0095$
LatentGAN	$0.297_{\pm}0.0087$	$0.828_{\pm}0.0117$	$0.537_{\pm}0.0004$	$0.513_{\pm}0.0002$	$0.999_{\pm}0.0004$	$0.997_{\pm}0.0007$	$0.887_{\pm}0.0009$	$0.107_{\pm}0.0098$
JTVAE	$0.395_{\pm}0.0234$	$0.938_{\pm}0.0531$	$0.548_{\pm}0.0076$	$0.519_{\pm}0.007$	$0.997_{\pm}0.0003$	$0.995_{\pm}0.0002$	$0.896_{\pm}0.0039$	$0.101_{\pm}0.0105$
HJTGAN (Ours)	$0.819_{\pm}0.0317$	$1.343_{\pm}0.0454$	$\mathbf{0.631_{\pm}0.0037}$	$\mathbf{0.593_{\pm}0.0023}$	$0.996_{\pm}0.0004$	$0.995_{\pm}0.0005$	$0.874_{\pm}0.0024$	$\mathbf{0.113_{\pm}0.0073}$

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR.

Chen, W., Han, X., Lin, Y., Zhao, H., Liu, Z., Li, P., Sun, M., and Zhou, J. (2021). Fully hyperbolic neural networks. *arXiv preprint arXiv:2105.14686*.

Dai, S., Gan, Z., Cheng, Y., Tao, C., Carin, L., and Liu, J. (2021). Apo-vae: Text generation in hyperbolic space. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 416–431.

Ganea, O., Bécigneul, G., and Hofmann, T. (2018). Hyperbolic neural networks. *Advances in neural information processing systems*, 31:5345–5355.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jin, W., Barzilay, R., and Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*, pages 2323–2332. PMLR.

References (cont.)

Law, M., Liao, R., Snell, J., and Zemel, R. (2019). Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pages 3672–3681. PMLR.

Liu, Q., Nickel, M., and Kiela, D. (2019). Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32:8230–8241.

Monath, N., Zaheer, M., Silva, D., McCallum, A., and Ahmed, A. (2019). Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 714–722.

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. (2019). A wrapped normal distribution on hyperbolic space for gradient-based learning. In *International Conference on Machine Learning*, pages 4693–4702. PMLR.

Nickel, M. and Kiela, D. (2017). Poincaré embeddings for learning hierarchical representations. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6341–6350. Curran Associates, Inc.

Nickel, M. and Kiela, D. (2018). Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR.

References (cont.)

Polykovskiy, D., Zhebrak, A., Sanchez-Lengeling, B., Golovanov, S., Tatanov, O., Belyaev, S., Kurbanov, R., Artamonov, A., Aladinskiy, V., Veselov, M., et al. (2020). Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:1931.

Shimizu, R., Mukuta, Y., and Harada, T. (2021). Hyperbolic neural networks++. In *International Conference on Learning Representations*.

Villani, C. (2009). *Optimal transport: old and new*, volume 338. Springer.

Yu, K., Visweswaran, S., and Batmanghelich, K. (2020). Semi-supervised hierarchical drug embedding in hyperbolic space. *Journal of chemical information and modeling*, 60(12):5647–5657.