Hyperbolic image embeddings

November 27, 2019

Overview

Motivation for hyperbolic embeddings

- Currently, in computer vision, many methods employ either Euclidean or spherical embeddings
- In our work, we propose to extend it to hyperbolic embeddings
- Hyperbolic spaces are especially suitable for embeddings of hierarchies
- We hypothesise that there can be hidden hierarchies in visual data.

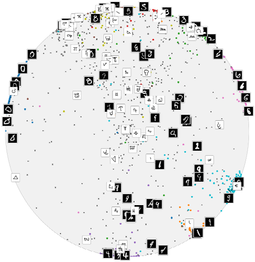


Figure 1: An example of two–dimensional Poincaré embeddings computed by a hyperbolic neural network trained on MNIST, and evaluated additionally on Omniglot.

Hyperbolic space

- Hyperbolic space is a space with constant **negative** curvature
- Euclidean space has constant **zero** curvature
- Spherical spaces have constant **positive** curvature

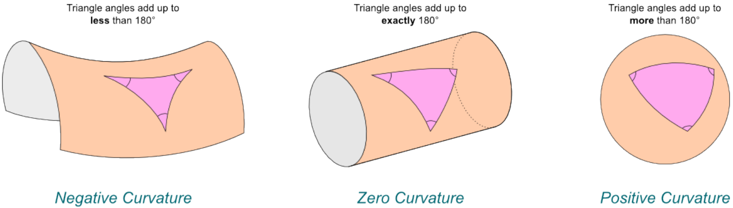


Figure 2: Triangles in spaces of different curvature. Source: `http://www.science4all.org/article/brazuca/`

Poincaré ball model

- In our work, we use Poincaré model of hyperbolic geometry
- Poincaré ball model $(\mathbb{D}^n, g^{\mathbb{D}})$ is manifold $\mathbb{D}^n = \{x \in \mathbb{R}^n : \|x\| < 1\}$ equipped with the following Riemannian metric $g_x^{\mathbb{D}} = \lambda_x^2 g^E$, where $\lambda_x = \frac{2}{1-\|x\|^2}$ – conformal factor, $g^E = I_n$ – Euclidean metric tensor
- Poincaré ball is conformal to Euclidean space
- In this model the *geodesic distance* between two points is given by the following expression:

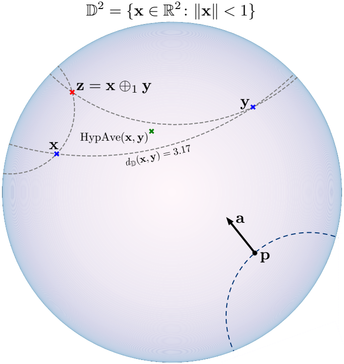
$$d_{\mathbb{D}}(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}\left(1 + 2\frac{\|\mathbf{x}-\mathbf{y}\|^2}{(1-\|\mathbf{x}\|^2)(1-\|\mathbf{y}\|^2)}\right). \quad (1)$$

- Hyperbolic spaces are **gyrovector** spaces
- This framework of gyrovector spaces allows to define operations as sum, product etc. in hyperbolic spaces

For a pair $\mathbf{x}, \mathbf{y} \in \mathbb{D}_c^n$, the **Möbius addition** is defined as follows:

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

c – curvature parameter; for $c = 0$ we recover Euclidean sum.



- **Möbius scalar multiplication** is defined as:

$$r \otimes_c x = (1/\sqrt{c}) \tanh(r \operatorname{arctanh}(\sqrt{c}\|x\|)) \frac{x}{\|x\|}. \tag{3}$$

- **Möbius matrix by vector** product is as follows:

$$\mathrm{M}^{\otimes_c}(\mathbf{x}) = \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\mathrm{M}\mathbf{x}\|}{\|\mathbf{x}\|} \operatorname{arctanh}(\sqrt{c}\|\mathbf{x}\|)\right) \frac{\mathrm{M}\mathbf{x}}{\|\mathrm{M}\mathbf{x}\|}, \tag{4}$$

- **Linear hyperbolic layer** represented by mapping $\mathrm{M}\mathbf{x} + \mathbf{b}$ is then generalized as $\mathrm{M}^{\otimes_c}(\mathbf{x}) \oplus_c \mathbf{b}$.

- The *exponential* map $\exp_{\mathbf{x}}^c$ is a function from $T_{\mathbf{x}}\mathbb{D}_c^n \cong \mathbb{R}^n$ to $\mathbb{D}_c^n$, which is given by

$$\exp_{\mathbf{x}}^c(\mathbf{v}) = \mathbf{x} \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_{\mathbf{x}}^c\|\mathbf{v}\|}{2} \right) \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \right). \tag{5}$$

- The inverse *logarithmic* map is defined as

$$\log_{\mathbf{x}}^c(\mathbf{y}) = \frac{2}{\sqrt{c}\lambda_{\mathbf{x}}^c} \mathrm{arctanh}(\sqrt{c}\| -\mathbf{x} \oplus_c \mathbf{y}\|) \frac{-\mathbf{x} \oplus_c \mathbf{y}}{\| -\mathbf{x} \oplus_c \mathbf{y}\|}. \tag{6}$$

- *Hyperbolic Averaging*, a substitute for averaging which is widely used in many algorithms, is defined as

$$\mathrm{HypAve}(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \sum_{i=1}^{N} \gamma_i \mathbf{x}_i / \sum_{i=1}^{N} \gamma_i, \tag{7}$$

where $\gamma_i = \frac{1}{\sqrt{1-c\|\mathbf{x}_i\|^2}}$ are the Lorentz factors.

Delta-hyperbolicity

- We can estimate a *degree of hyperbolicity* of a dataset – δ (defined on next slide)
- This is needed to approximate a suitable radius of a ball $r = \frac{1}{\sqrt{c}}$
- E.g., for Euclidean case, i.e., $c = 0$, the corresponding radius would be equal to infinity
-
$$c(X) = \left(\frac{\delta_P}{\delta_X}\right)^2, \tag{8}$$

  where $\delta_P = \log(1 + \sqrt{2}) \sim 0.88$ – δ for Poincaré ball of radius 1

Delta-hyperbolicity

- We need to define *Gromovproduct* for points $x, y, z \in X$:

$$(y, z)_x = \frac{1}{2}(d(x, y) + d(x, z) - d(y, z)). \tag{9}$$

- Then δ is the minimal value such that the following four-point condition holds for all points $x, y, z, w \in X$:

$$(x, z)_w \geq \min((x, y)_w, (y, z)_w) - \delta. \tag{10}$$

- It suffices to find the δ for some fixed point w_0
- A more computational friendly way:
  1. we first compute the matrix A of pairwise Gromov products
  2. then, δ value is simply the largest coefficient in the matrix $(A \otimes A) - A$, where $\otimes$ denotes the min-max matrix product

$$A \otimes B = \max_k \min\{A_{ik}, B_{kj}\}.$$

Delta-hyperbolicity

- To make it scale-invariant, we can compute relative delta value, which lies in $[0, 1]$ and specifies how close is the dataset to a perfect hyperbolic space $\delta_{rel}(X) = \frac{2\delta(X)}{\mathrm{diam}(X)}$
- We measure δ values for a set of features extracted from datasets using VGG16 network

	Tree	Omniglot	CUB	*mini*ImageNet	S_2	$S_2, z > 0$
$2\delta(X)/\mathrm{diam}(X)$	0	0.31	0.23	0.14	0.99	0.94
c	-	0.036	0.005	0.007	-	-

Figure 3: The relative delta $2\delta(X)\mathrm{diam}(X)$ and curvature parameter values calculated for different datasets. S_2 and $S_2, z > 0$ denote the two–dimensional unit sphere and upper hemisphere correspondingly (1700 points were sampled from each one)

Few-shot learning

- We focused on the problem of **few-shot learning**
- The concept of few-shot learning is to train the network to generalize to unseen samples
- The task is formulated as m shot n way classification problem, where m is the number of labeled samples per class, and n is the number of classes to classify among

Experimental setup

- As a baseline, we took ProtoNet where one uses a so-called *prototype representation* of a class, which is defined as a mean of the embedded support set of a class
- Generalizing this concept to hyperbolic space, we substitute the Euclidean mean operation by HypAve defined earlier
- We map extracted features to hyperbolic space, compute pairwise hyperbolic distances and use HypAve operation.

Experiment results

Dataset	Model	c	1-shot 5-way	5-shot 5-way
*Mini*ImageNet	MatchNet [43]	-	43.56 ± 0.84	55.31 ± 0.73
	ProtoNet	-	48.29 ± 0.19	66.11 ± 0.16
	RelationNet [39]	-	50.44 ± 0.82	65.32 ± 0.70
	Hyperbolic ProtoNet	0.05	$\mathbf{51.57 \pm 0.2}$	66.27 ± 0.17
	Hyperbolic ProtoNet	0.007	47.97 ± 0.19	$\mathbf{68.92 \pm 0.16}$
CUB	ProtoNet	-	54.58 ± 0.24	68.04 ± 0.19
	Hyperbolic ProtoNet	0.05	$\mathbf{60.52 \pm 0.25}$	72.22 ± 0.19
	Hyperbolic ProtoNet	0.005	58.03 ± 0.24	$\mathbf{75.80 \pm 0.17}$

Figure 4: Experimental results on two datasets: MiniImageNet and CUB averaged over 10,000 test episodes and are reportedwith 95% confidence intervals.

THAT'S IT