Hyperbolic Neural Networks

Octavian-Eugen Ganea*
Department of Computer Science
ETH Zürich
Zurich, Switzerland
`octavian.ganea@inf.ethz.ch`

Gary Bécigneul*
Department of Computer Science
ETH Zürich
Zurich, Switzerland
`gary.becigneul@inf.ethz.ch`

Thomas Hofmann
Department of Computer Science
ETH Zürich
Zurich, Switzerland
`thomas.hofmann@inf.ethz.ch`

Use **hyperbolic space** instead of **Euclidean space**

for embedding data with a latent **hierarchical** structure

Use **hyperbolic space** instead of **Euclidean space**

for embedding data with a latent **hierarchical** structure

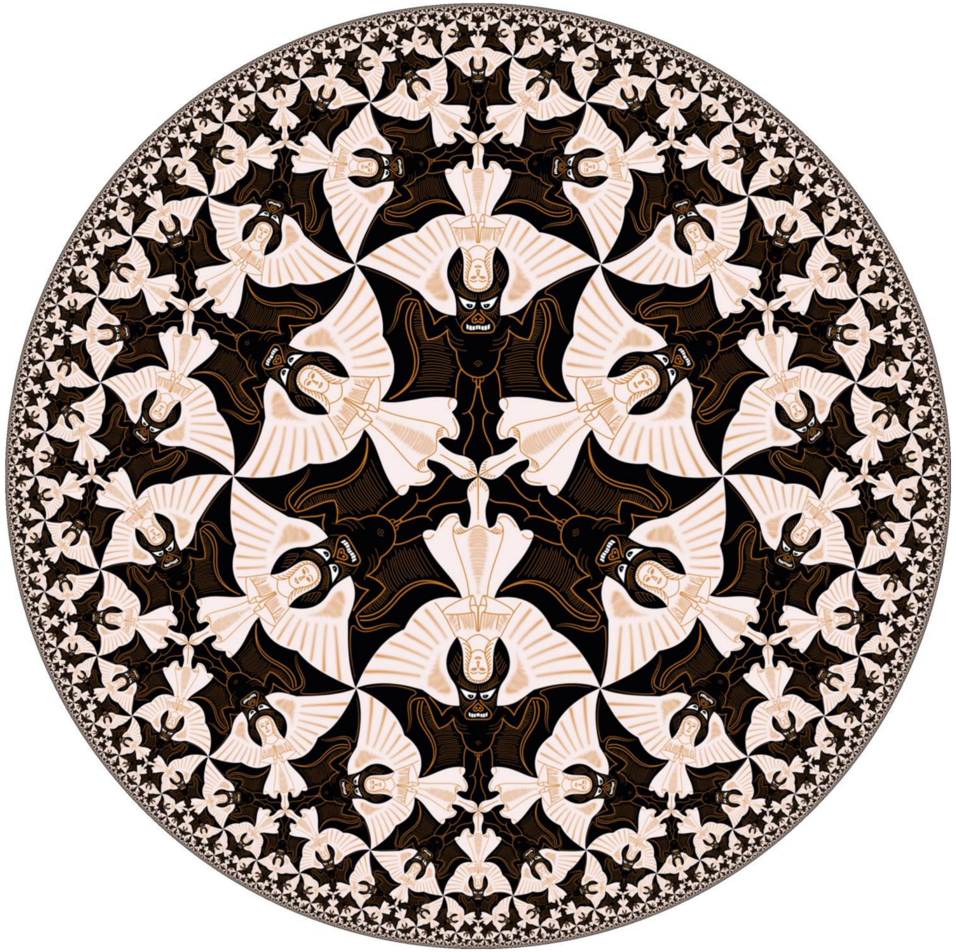


image source: http://inspirehep.net/record/1355197/plots

The volume of a ball grows **exponentially** with its radius!

Use **hyperbolic space** instead of **Euclidean space**

for embedding data with a latent **hierarchical** structure

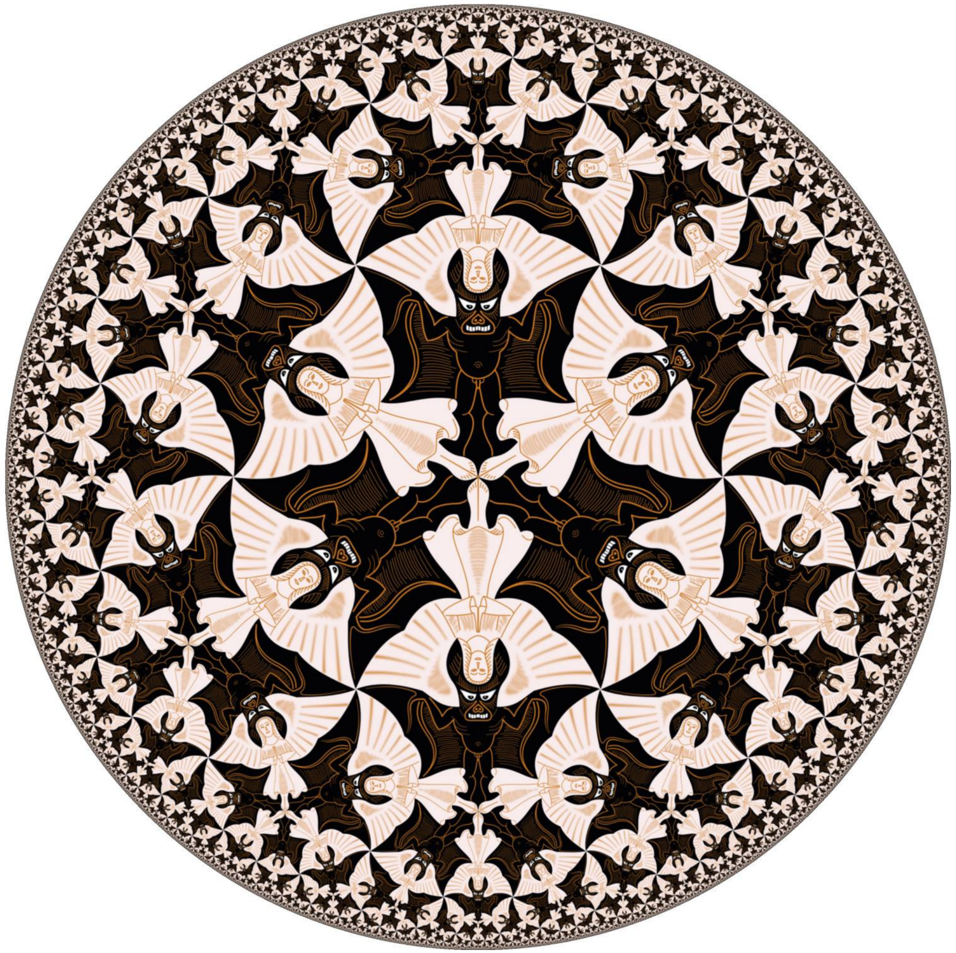


image source: http://inspirehep.net/record/1355197/plots

The volume of a ball grows **exponentially** with its radius!

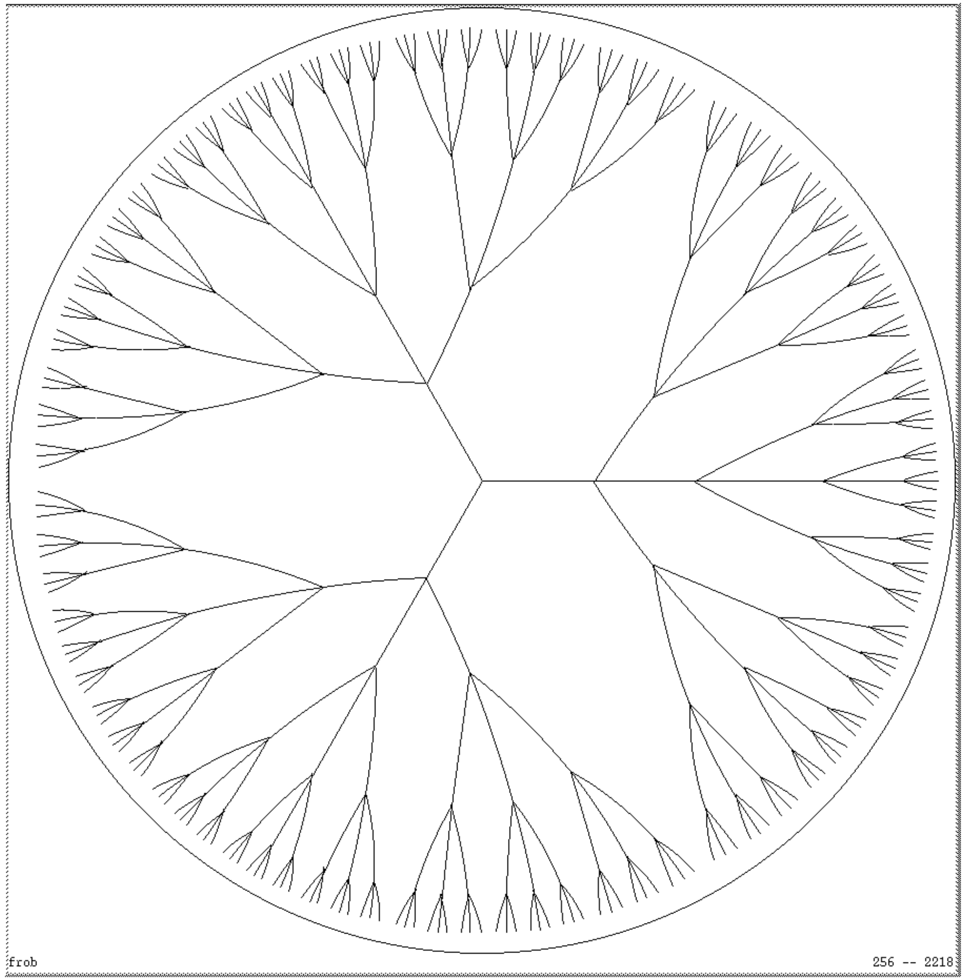


Image source: http://prior.sigchi.org

Similarly as for a tree: the number of nodes grows **exponentially** with the tree depth!

Use **hyperbolic space** instead of **Euclidean space** for embedding data with a latent **hierarchical** structure

Hot topic in ML since

Poincaré Embeddings for Learning Hierarchical Representations,

Nickel & Kiela, (NIPS 2017)

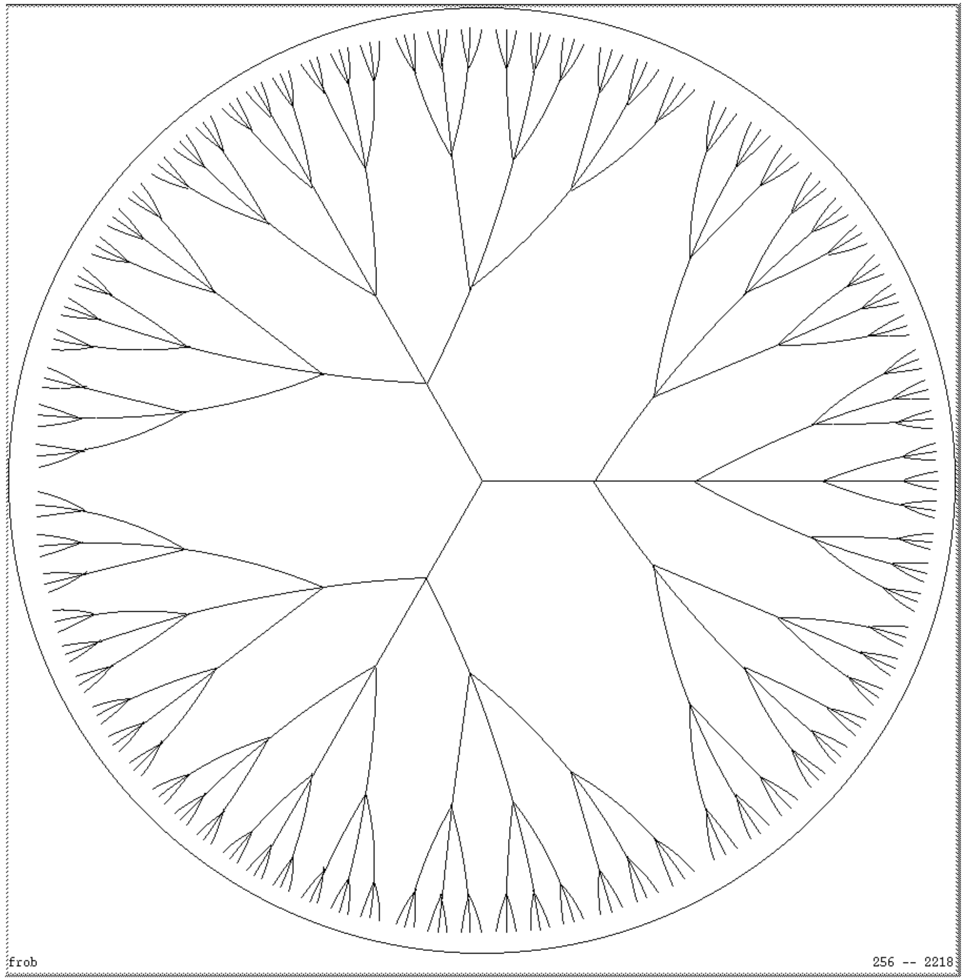


Image source: http://prior.sigchi.org

Difficulties

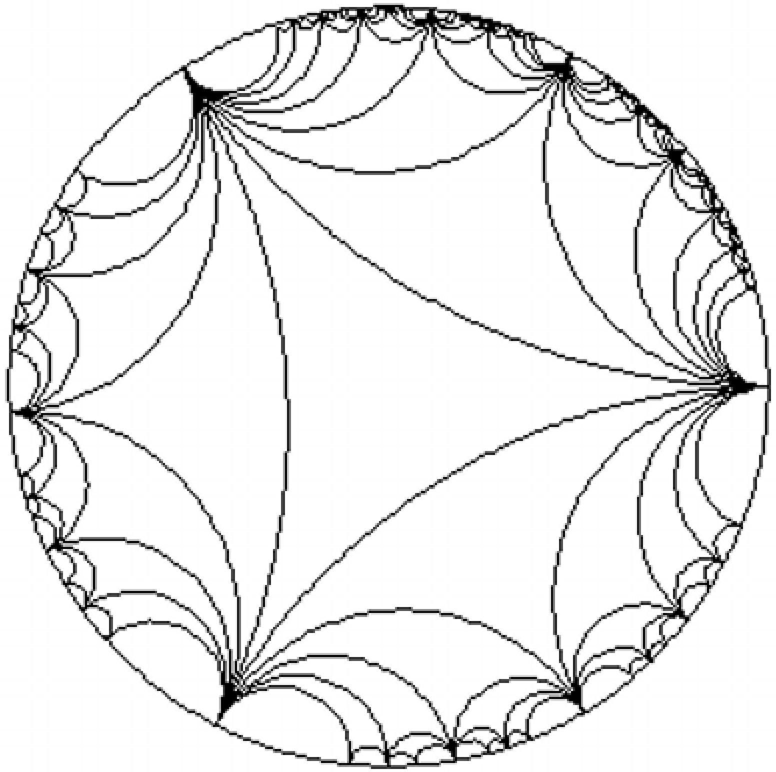
- HOW TO USE HYPERBOLIC EMBEDDINGS IN **DOWNSTREAM TASKS** ?
- HOW TO FEED HYPERBOLIC EMBEDDINGS TO **NEURAL NETS** ?

Difficulties

- HOW TO USE HYPERBOLIC EMBEDDINGS IN **DOWNSTREAM TASKS** ?

- HOW TO FEED HYPERBOLIC EMBEDDINGS TO **NEURAL NETS** ?

- basic Euclidean operations **not defined** in the hyperbolic space! *e.g. vector addition should follow hyperbolic "straight-lines", i.e. geodesics*



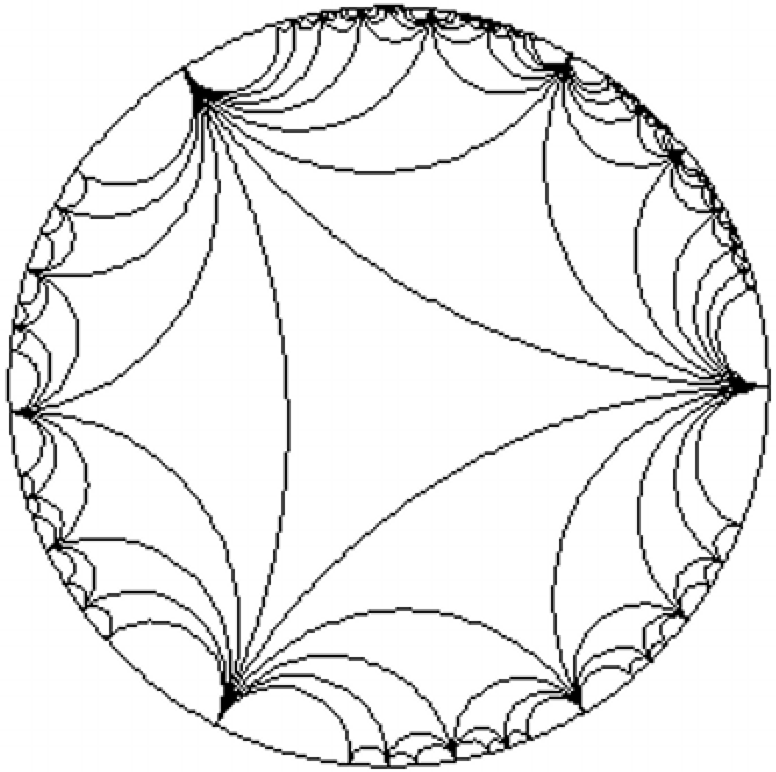
Difficulties

- HOW TO USE HYPERBOLIC EMBEDDINGS IN **DOWNSTREAM TASKS** ?

- HOW TO FEED HYPERBOLIC EMBEDDINGS TO **NEURAL NETS** ?

- basic Euclidean operations **not defined** in the hyperbolic space! *e.g. vector addition should follow hyperbolic "straight-lines", i.e. geodesics*

- neural networks **should not ignore** the hyperbolic geometry (e.g. how to enforce hyperbolicity of an RNN's hidden states ?)

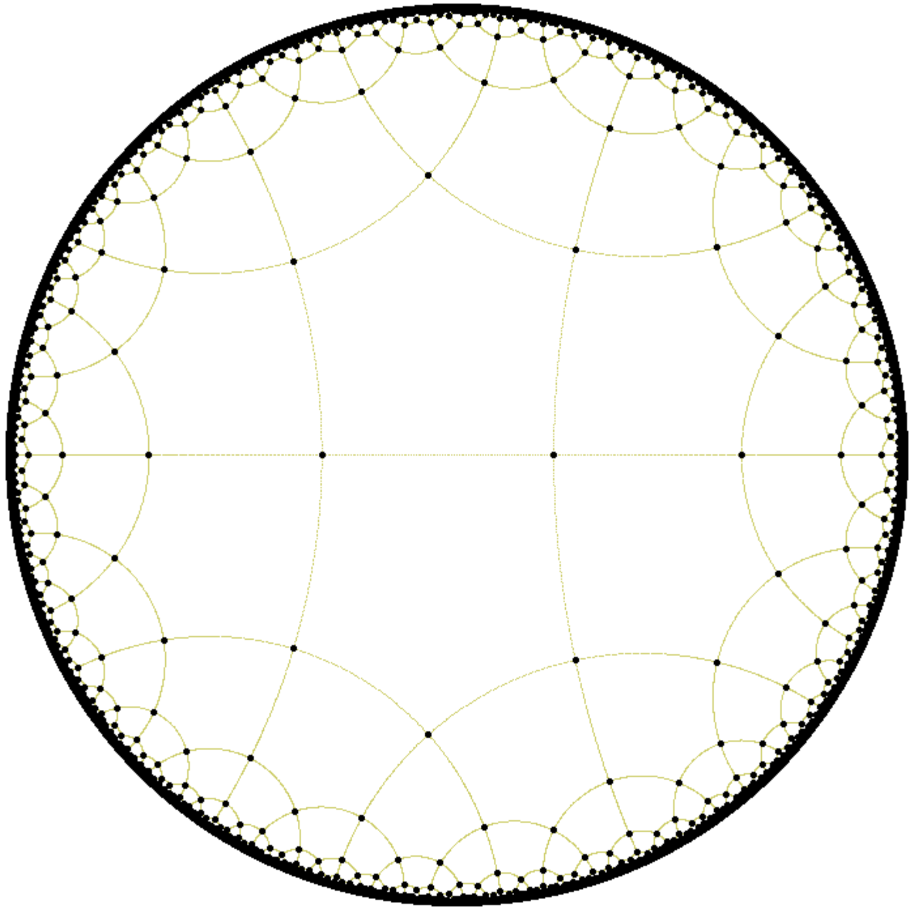


Poincaré Ball

$$\mathbb{D}_c^n = \{x \in \mathbb{R}^n, c\|x\|_2^2 < 1\}$$

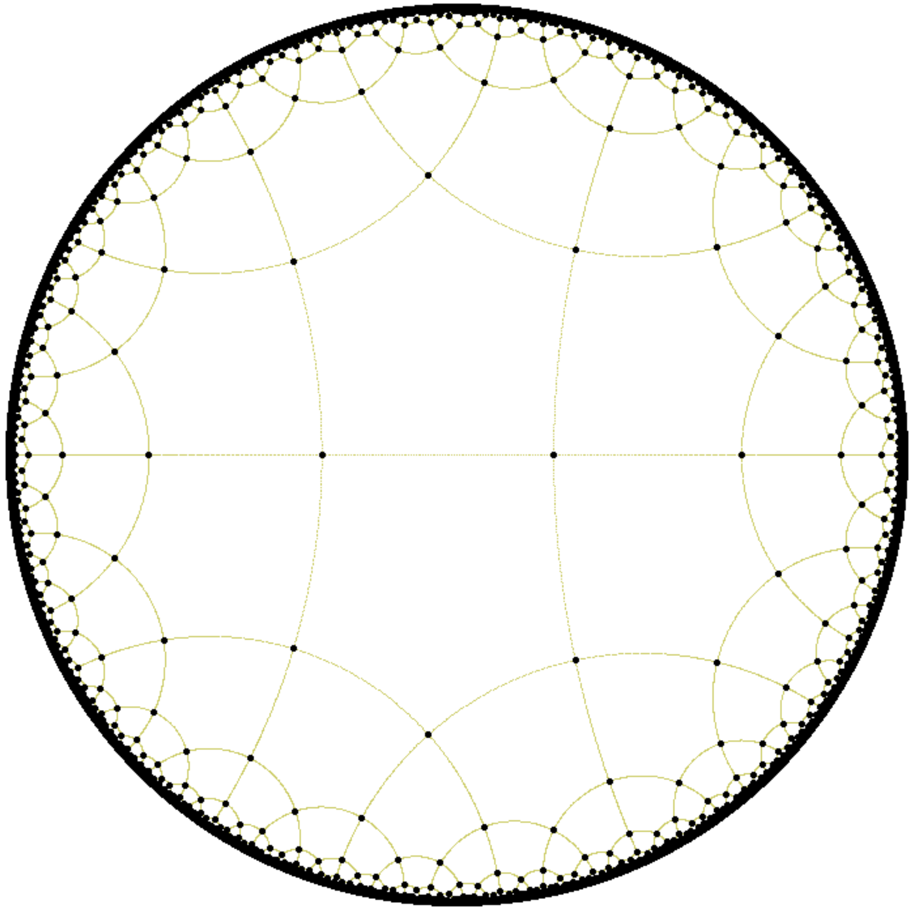
Poincaré Ball

$$\mathbb{D}_c^n = \{x \in \mathbb{R}^n, c\|x\|_2^2 < 1\}$$



Poincaré Ball

$$\mathbb{D}_c^n = \{x \in \mathbb{R}^n, c\|x\|_2^2 < 1\}$$



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

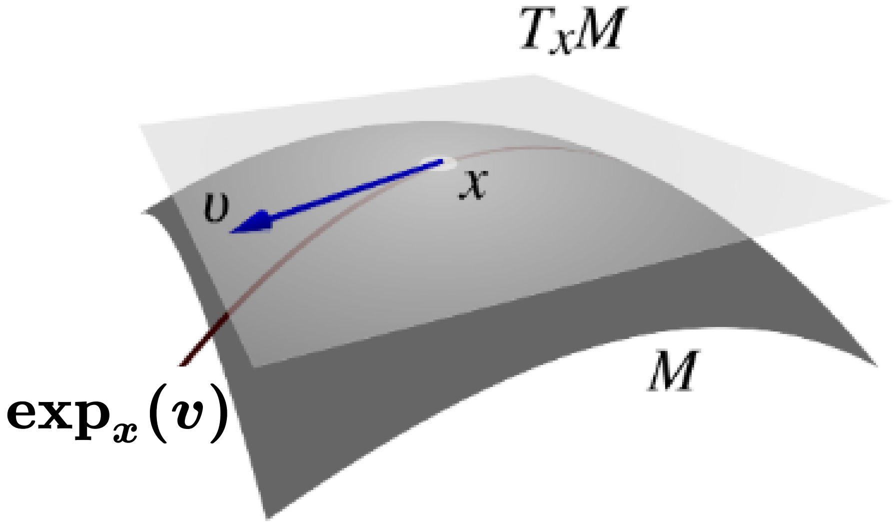
Use **Gyro-vector spaces** to *generalize basic operations* and *neural networks* from Euclidean to hyperbolic spaces:

- Gyro-vs: - analogue of Euclidean vector spaces
  - used in relativity theory (speeds of particles are hyperbolic)
- Vector addition $x + y \iff x \oplus_c y$
- Scalar multiplication $rx \iff r \otimes_c x$
  - Closed form distance $d_c(x, y) = (2/\sqrt{c}) \tanh^{-1}\left(\sqrt{c}\|-x \oplus_c y\|\right)$
  - Closed form geodesics: $\gamma_{x \to y}(t) := x \oplus_c \left(-x \oplus_c y\right) \otimes_c t$

Our contributions

1) We *connect Gyro-vs and Riemannian hyperbolic geometry*

- Closed form $\exp_x(v)$, $\log_x(y)$
- Closed form parallel transport (move across tangent spaces)



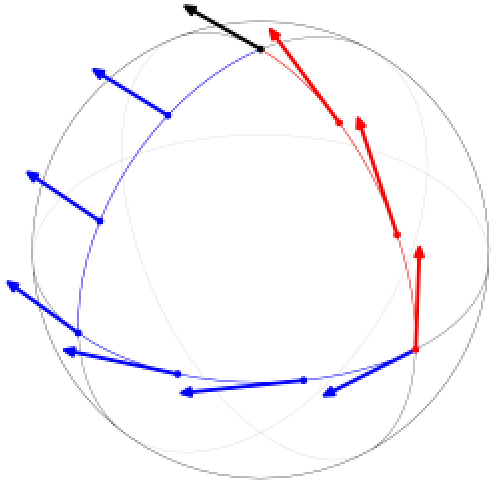


Image sources: stackexchange.com , wikipedia.org

Our contributions

2) Hyperbolic Feed-forward Neural Networks

- *Möbius version of* $f : \mathbb{R}^n \to \mathbb{R}^m$ (e.g. pointwise non-linearity):

$$f^{\otimes_c} : \mathbb{D}_c^n \to \mathbb{D}_c^m, \qquad f^{\otimes_c}(x) := \exp_{\mathbf{0}}^c(f(\log_{\mathbf{0}}^c(x)))$$

Our contributions

2) Hyperbolic Feed-forward Neural Networks

- *Möbius version of* $f : \mathbb{R}^n \to \mathbb{R}^m$ (e.g. pointwise non-linearity):

$$f^{\otimes_c} : \mathbb{D}_c^n \to \mathbb{D}_c^m, \qquad f^{\otimes_c}(x) := \exp_{\mathbf{0}}^c(f(\log_{\mathbf{0}}^c(x)))$$

- Matrix - vector multiplication:

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

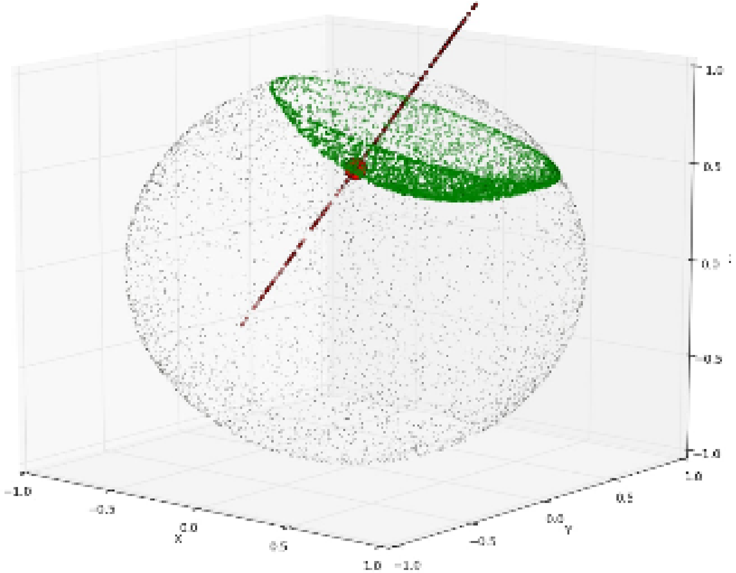
  - *Properties:* matrix associativity, scalar-matrix associativity, preserved rotations

Our contributions

3) Hyperbolic Softmax layer - Multiclass Logistic Regression

- Hyperbolic hyperplane:

$$\tilde{H}^c_{a,p} = \{x \in \mathbb{D}^n_c : \langle -p \oplus_c x, a \rangle = 0\}.$$



- **Theorem:** closed form of $d_c(x, \tilde{H}^c_{a,p})$

- Final MLR formula (based on Lebanon and Lafferty,2004):

$$p(y = k|x) \propto \exp\left(\frac{\lambda^c_{p_k}\|a_k\|}{\sqrt{c}} \sinh^{-1}\left(\frac{2\sqrt{c}\langle -p_k \oplus_c x, a_k\rangle}{(1 - c\| - p_k \oplus_c x\|^2)\|a_k\|}\right)\right)$$

Our contributions

4) Hyperbolic Recurrent Networks, e.g. hGRU

$$\text{hyp-GRU} \leftarrow \begin{cases} r_t = \sigma \log_{\mathbf{0}}^{c}(W^r \otimes_c h_{t-1} \oplus_c U^r \otimes_c x_t \oplus_c b^r) \\ \tilde{h}_t = \varphi^{\otimes_c}((W \operatorname{diag}(r_t)) \otimes_c h_{t-1} \oplus_c U \otimes_c x_t \oplus b) \\ h_t = h_{t-1} \oplus_c \operatorname{diag}(z_t) \otimes_c (-h_{t-1} \oplus_c \tilde{h}_t) \end{cases}$$

- Hyperbolic hidden states
- **Theorem:** update-gate mechanism derived from *time-warping invariance* principle (via gyro-derivative and gyro-chain-rule)

Property: *All our models recover their Euclidean variants when curvature* $c \to 0$.

Riemannian Optimization

- Both Euclidean and hyperbolic parameters
- Riemannian SGD:

$$x \leftarrow \exp_x^c(-\eta \nabla_x^R \mathcal{L}), \quad x \in \mathbb{D}_c^n$$

- Riemannian gradient:

$$\nabla_x^R \mathcal{L} = (1/\lambda_x^c)^2 \nabla_x \mathcal{L}, \quad \text{conformal factor } \lambda_x^c = \frac{2}{1 - c\|x\|^2}$$

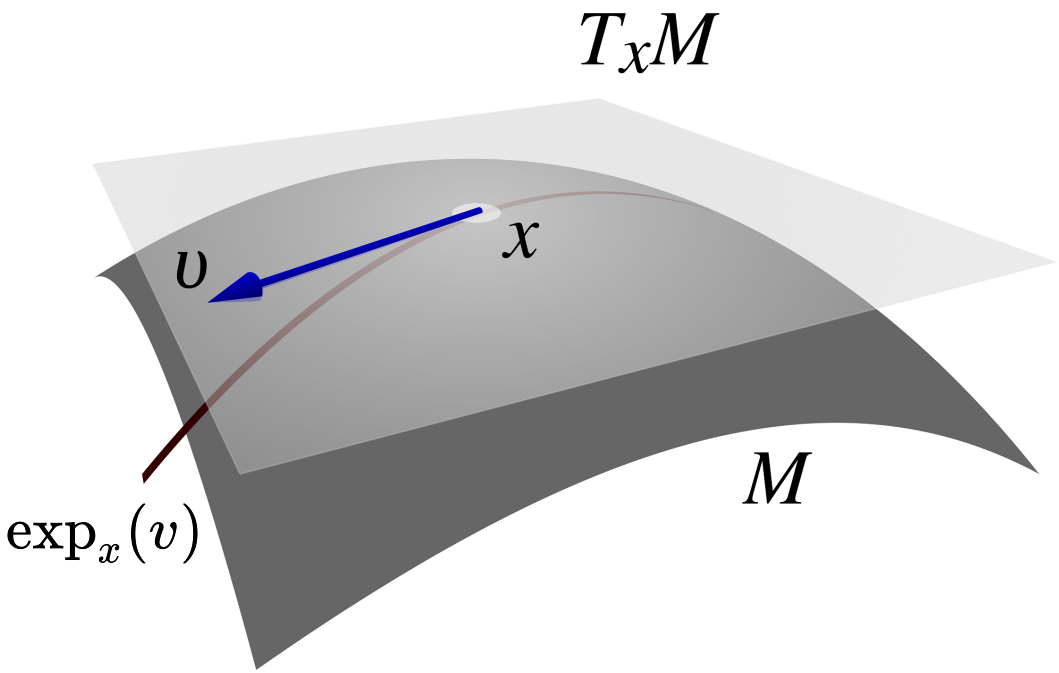


Image source: stackexchange.com

Experiments

Experiments

1) Textual Entailment tasks (semantic + syntactic).

TEST ACCURACY	SNLI	PREFIX-10%	PREFIX-30%	PREFIX-50%
FULLY EUCLIDEAN RNN	**79.34** %	89.62 %	81.71 %	72.10 %
HYP RNN+FFNN, EUCL MLR	**79.18** %	96.36 %	**87.83** %	**76.50** %
FULLY HYPERBOLIC RNN	78.21 %	**96.91** %	87.25 %	62.94 %
FULLY EUCLIDEAN GRU	**81.52** %	95.96 %	86.47 %	75.04 %
HYP GRU+FFNN, EUCL MLR	79.76 %	**97.36** %	**88.47** %	**76.87** %
FULLY HYPERBOLIC GRU	**81.19** %	**97.14** %	**88.26** %	**76.44** %

All word and sentence embeddings have dimension 5.

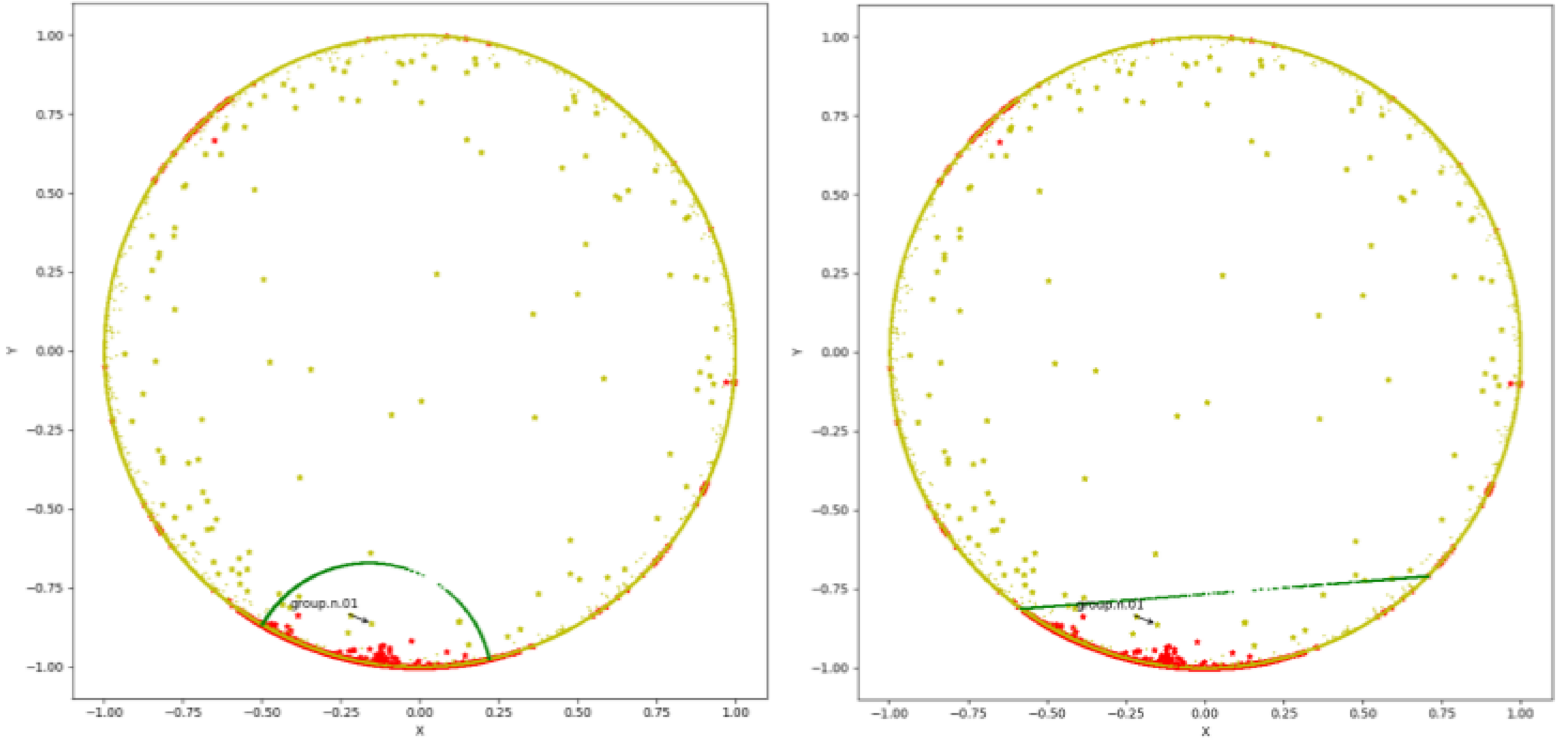
Experiments

2) MLR experiments.

Test F1 classification scores (%) for 4 subtrees of the WordNet tree.

WordNet subtree	Model	d = 2	d = 3	d = 5	d = 10
animal.n.01 3218 / 798	Hyp	$\mathbf{47.43 \pm 1.07}$	$\mathbf{91.92 \pm 0.61}$	$\mathbf{98.07 \pm 0.55}$	$\mathbf{99.26 \pm 0.59}$
	Eucl	41.69 ± 0.19	68.43 ± 3.90	95.59 ± 1.18	$\mathbf{99.36 \pm 0.18}$
	$\log_{\mathbf{0}}$	38.89 ± 0.01	62.57 ± 0.61	89.21 ± 1.34	98.27 ± 0.70
group.n.01 6649 / 1727	Hyp	$\mathbf{81.72 \pm 0.17}$	$\mathbf{89.87 \pm 2.73}$	$\mathbf{87.89 \pm 0.80}$	$\mathbf{91.91 \pm 3.07}$
	Eucl	61.13 ± 0.42	63.56 ± 1.22	67.82 ± 0.81	$\mathbf{91.38 \pm 1.19}$
	$\log_{\mathbf{0}}$	60.75 ± 0.24	61.98 ± 0.57	67.92 ± 0.74	$\mathbf{91.41 \pm 0.18}$
worker.n.01 861 / 254	Hyp	$\mathbf{12.68 \pm 0.82}$	$\mathbf{24.09 \pm 1.49}$	$\mathbf{55.46 \pm 5.49}$	$\mathbf{66.83 \pm 11.38}$
	Eucl	10.86 ± 0.01	22.39 ± 0.04	35.23 ± 3.16	47.29 ± 3.93
	$\log_{\mathbf{0}}$	9.04 ± 0.06	22.57 ± 0.20	26.47 ± 0.78	36.66 ± 2.74
mammal.n.01 953 / 228	Hyp	$\mathbf{32.01 \pm 17.14}$	$\mathbf{87.54 \pm 4.55}$	$\mathbf{88.73 \pm 3.22}$	$\mathbf{91.37 \pm 6.09}$
	Eucl	$\mathbf{15.58 \pm 0.04}$	44.68 ± 1.87	59.35 ± 1.31	77.76 ± 5.08
	$\log_{\mathbf{0}}$	13.10 ± 0.13	44.89 ± 1.18	52.51 ± 0.85	56.11 ± 2.21

Experiments



Hyperbolic (left) vs Direct Euclidean (right) binary MLR used to classify nodes as being part in the GROUP.N.01 *subtree of the WordNet noun hierarchy solely based on their Poincaré embeddings.*

THANK YOU!

Please visit our website:

hyperbolicdeeplearning.com

HDL

HYPERBOLIC DEEP LEARNING

A nascent and promising field

Octavian Ganea

is currently looking for postdoctoral positions!

Matrix-vector multiplication

We define:

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

Nice properties:

- Matrix associativity: $M\otimes(N\otimes x) = (MN)\otimes x$
- Compatibility with scalar multiplication: $M\otimes(r\otimes x) = (rM)\otimes x = r\otimes(M\otimes x)$
- Directions are preserved: $M\otimes x/\|M\otimes x\| = Mx/\|Mx\|$ for $Mx\neq 0$
- Rotations are preserved: $M\otimes x = Mx$ for $M\in\mathcal{O}_n(\mathbb{R})$

Matrix-vector multiplication

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

When the curvature c goes to zero, it recovers the usual matrix multiplication!

$$\lim_{c \to 0} M^{\otimes_c}(x) = Mx$$