Hyperbolic Deep Learning

Max Kochurov

Samsung, Skoltech

10 April, 2020

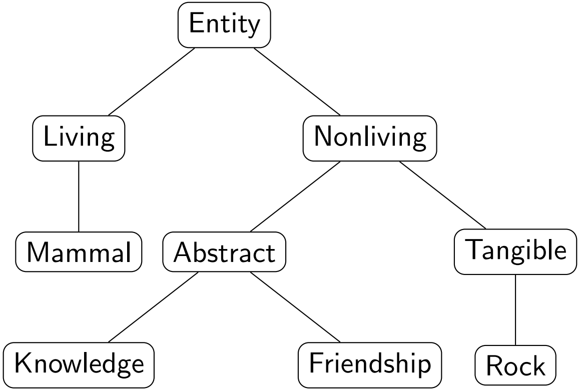
Words

Words contain latent hierarchical (non-unique) structure. Several works learn the word structure

- LDA (D. Blei, 2003)
- Glove (J. Pennington, 2014)
- Word2Vec (T. Mikolov, 2013)

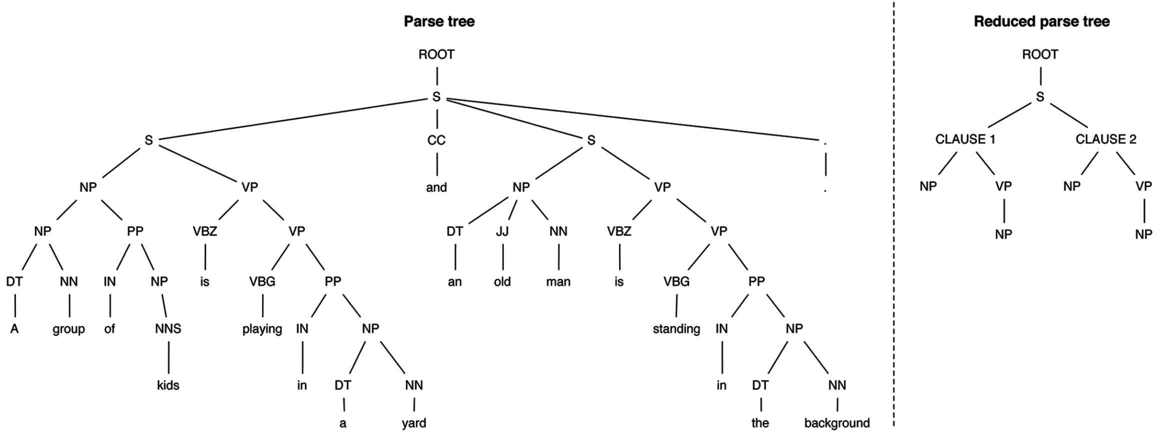
Some share parameters across levels of hierarchy

- F. Morin, Y. Bengio, 2005
- Hierarchical Attention (Z. Yang, 2016)



Sentences

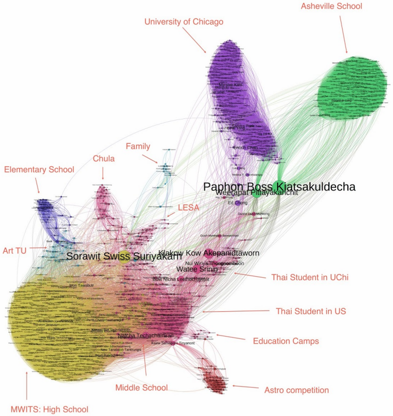
Sentence can also be represented as a tree



Graphs

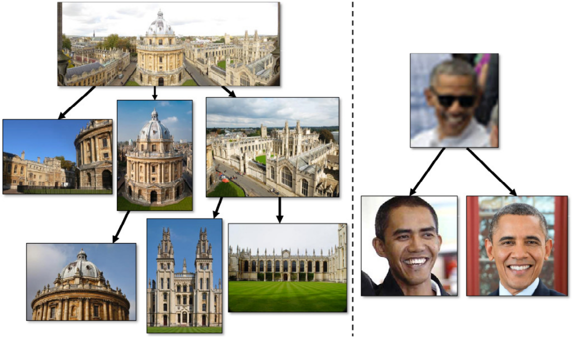
Graphs also may contain hierarchical structure. Empirically they do, node degree distribution usually follows power law in real data. Graphs may also have

- cycles
- sub-trees
- large clusters



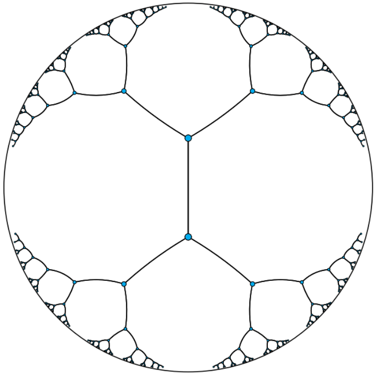
Images

Not obvious how to represent hierarchical structure there at all. However, some works (V. Khrulkov 2019, J. Yang 2011) try do make that assumption



Hyperbolic Space

- Hyperbolic spaces can embed hierarchy. Sarkar's Construction allows to embed K-trees in $\mathbb{H}^K$ space (C. De Sa, 2018)
- Very well studied as a part of special theory of relativity (A. Ungar 2008)
- Can be viewed as Riemannian Manifold (J. Lee, 1991)

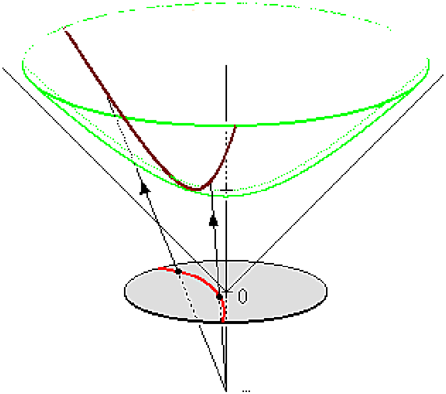


What is Hyperbolic space

Hyperboloid Lorenz model

$$\mathbb{H}^n = \{x \ : \ x_0^2 - x_1^2 - \ldots - x_n^2 = 1\}$$
$$Q(x_0, x_1, \ldots, x_n) = x_0^2 - x_1^2 - \ldots - x_n^2.$$

Lorenz model is a n-dimensional Hyperboloid $\mathbb{H}^n$, embedded in $\mathbb{R}^{n+1}$ Euclidean space.



What is Hyperbolic space

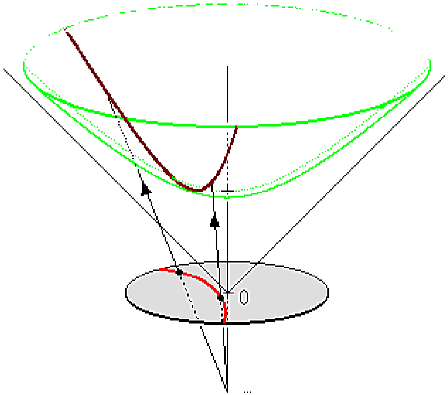
Hyperboloid Lorenz model

$$\mathbb{H}^n = \{x \ : \ x_0^2 - x_1^2 - \ldots - x_n^2 = 1\}$$
$$Q(x_0, x_1, \ldots, x_n) = x_0^2 - x_1^2 - \ldots - x_n^2.$$

Lorenz model is a n-dimensional Hyperboloid $\mathbb{H}^n$, embedded in $\mathbb{R}^{n+1}$ Euclidean space.

Stereographic projection to Poincare Ball model

$$\Pi_{\mathbb{H}^n \to \mathbb{D}^n (x_0, \ldots, x_n)} = \frac{(x_1, \ldots, x_n)}{x_0 + 1}$$

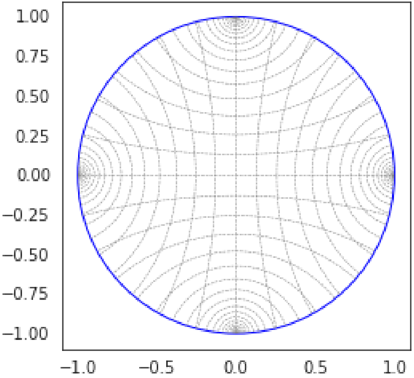


Poincare Ball

Hyperboloid is projected into a compact space bounded by radius $R = 1/\sqrt{-\kappa}$, where κ is constant sectional curvature.

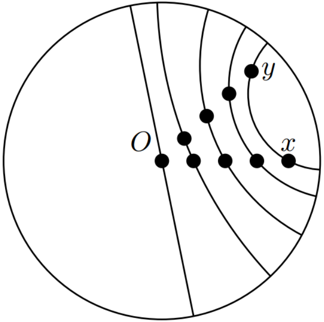
$$\mathbb{D}^n = \{x \in \mathbb{R} : \ \|x\|_2 < R\}$$

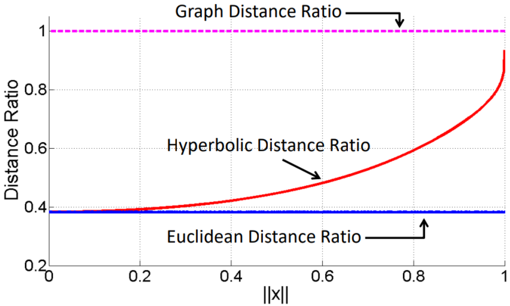
- A Riemannian manifold with conformal metric
- Geometry becomes Euclidean once $\kappa \to 0$



Capacity

Some more intuition, why we mention Hyperbolic space capacity

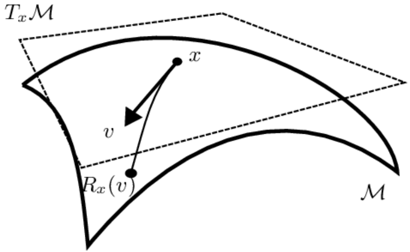


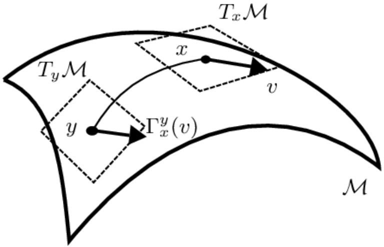


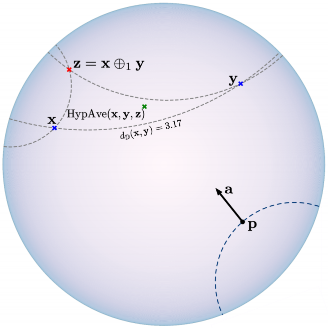
Distance $d(x, y) \to d(x, 0) + x(y, 0)$ in only Hyperbolic space[1]

[1] https://arxiv.org/abs/1804.03329

Geometry


$T_x\mathcal{M}$
x
v
$R_x(v)$
$\mathcal{M}$


$T_x\mathcal{M}$
$T_y\mathcal{M}$
x
v
y
$\Gamma_x^y(v)$
$\mathcal{M}$


$\mathbf{z} = \mathbf{x} \oplus_1 \mathbf{y}$
$\mathbf{y}$
$\mathbf{x}$
$\mathrm{HypAve}(\mathbf{x}, \mathbf{y}, \mathbf{z})$
$d_{\mathbb{D}}(\mathbf{x}, \mathbf{y}) = 3.17$
$\mathbf{a}$
$\mathbf{p}$

Two views

Riemannian Manifold view

- Comes from metric analysis
- Defines
  - $\exp_x(v) \ : \ \mathcal{T}_x\mathcal{M} \to \mathcal{M}$
  - $\log_x(y) \ : \ \mathcal{M} \to \mathcal{T}_x\mathcal{M}$
  - $\phi_{x\to y}(v) \ : \ \mathcal{T}_x \to \mathcal{T}_y\mathcal{M}$

  allowing Riemannian optimization

Gyrovector Space view

- Comes from Special Relativity analysis
- Generalizes vector spaces to hyperbolic geometry
- Useful math concepts
  - $x \oplus y$ – Möbius addition
  - $M \otimes x$ – Möbius matrix vector product
  - $r \odot x$ – Möbius scalar multiplication
  - others

Two views

Riemannian Manifold view

- Comes from metric analysis
- Defines
  - $\exp_x(v) \ : \ \mathcal{T}_x\mathcal{M} \to \mathcal{M}$
  - $\log_x(y) \ : \ \mathcal{M} \to \mathcal{T}_x\mathcal{M}$
  - $\phi_{x\to y}(v) \ : \ \mathcal{T}_x \to \mathcal{T}_y\mathcal{M}$

  allowing Riemannian optimization

Gyrovector Space view

- Comes from Special Relativity analysis
- Generalizes vector spaces to hyperbolic geometry
- Useful math concepts
  - $x \oplus y$ – Möbius addition
  - $M \otimes x$ – Möbius matrix vector product
  - $r \odot x$ – Möbius scalar multiplication
  - others

Connections between

$$x \oplus y = \exp_x(\phi_{\bar{0}\to x}(\log_0(y)))$$
$$M \otimes x = \exp_0(M \log_0(x))$$
$$r \odot x = \exp_0(r \log_0(x))$$

Curvature limits

We've viewed Gyrovector space away from Euclidean geometry.
Starting from here we need to take negative curvature ($c > 0$) in account

Gyrovector space **Euclidean space:** $c \to +0$

Curvature limits

We've viewed Gyrovector space away from Euclidean geometry.
Starting from here we need to take negative curvature ($c > 0$) in account

Gyrovector space

- $x \oplus_c y$ – Möbius addition

Euclidean space: $c \to +0$

- $x \oplus_0 y = x + y$ – addition

Definition

$$x \oplus_c y = \frac{(1 + 2c\langle x, y\rangle + c\|y\|_2^2)x + (1 - c\|x\|_2^2)y}{1 + 2c\langle x, y\rangle + c^2\|x\|_2^2\|y\|_2^2}$$

Non commitative $x \oplus_c y \neq y \oplus_c x$

Curvature limits

We've viewed Gyrovector space away from Euclidean geometry.
Starting from here we need to take negative curvature ($c > 0$) in account

Gyrovector space

- $x \oplus_c y$ – Möbius addition
- $M \otimes_c x$ – Möbius matrix vector product

Euclidean space: $c \to +0$

- $x \oplus_0 y = x + y$ – addition
- $M \otimes_0 x = Mx$ – matrix vector product

Definition

$$M \otimes_c x = (1/\sqrt{c}) \tanh\left(\frac{\|Mx\|_2}{\|x\|_2} \tanh^{-1}(\sqrt{c}\|x\|_2)\right) \frac{Mx}{\|Mx\|_2}$$

Curvature limits

We've viewed Gyrovector space away from Euclidean geometry.
Starting from here we need to take negative curvature ($c > 0$) in account

Gyrovector space

- $x \oplus_c y$ – Möbius addition
- $M \otimes_c x$ – Möbius matrix vector product
- $r \odot_c x$ – Möbius scalar multiplication

Euclidean space: $c \to +0$

- $x \oplus_0 y = x + y$ – addition
- $M \otimes_0 x = Mx$ – matrix vector product
- $r \odot_0 x$ – scalar multiplication

Definition

$$r \otimes_c x = (1/\sqrt{c}) \tanh(r \tanh^{-1}(\sqrt{c}\|x\|_2)) \frac{x}{\|x\|_2}$$

Building analogies

To bring concepts of hyperbolic geometry to machine learning models we need a grounded bottom up approach

- Euclidean models, something we understand, are important for inspiration

Building analogies

To bring concepts of hyperbolic geometry to machine learning models we need a grounded bottom up approach

- Euclidean models, something we understand, are important for inspiration
- Hyperbolic space properties contradict to Euclidean intuition (will be covered next)

Building analogies

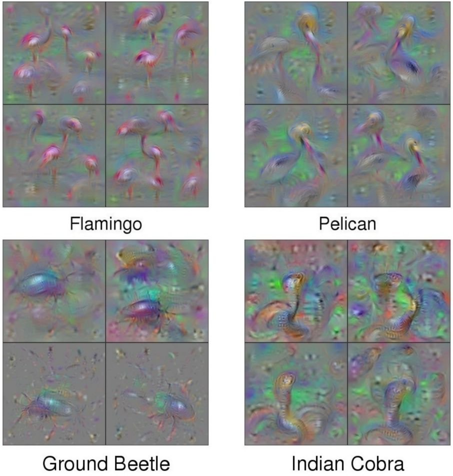
To bring concepts of hyperbolic geometry to machine learning models we need a grounded bottom up approach

- Euclidean models, something we understand, are important for inspiration
- Hyperbolic space properties contradict to Euclidean intuition (will be covered next)
- A model should behave as Euclidean in case of zero curvature ($c = 0$)

Intuition: Vector

$$x = (x_0, \dots, x_n) = r \cdot (x_0, \dots, x_n)/\|x\|_2$$

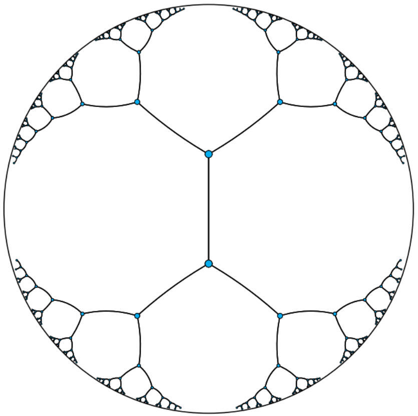
- In Euclidean case channels are important, we can visualize this (J. Yosinski, 2015)



Intuition: Vector

$$x = (x_0, \ldots, x_n) = r \cdot (x_0, \ldots, x_n)/\|x\|_2$$

- In Euclidean case channels are important, we can visualize this (J. Yosinski, 2015)
- In Hyperbolic case a vector defines position in hierarchy, distance from zero and direction are both important

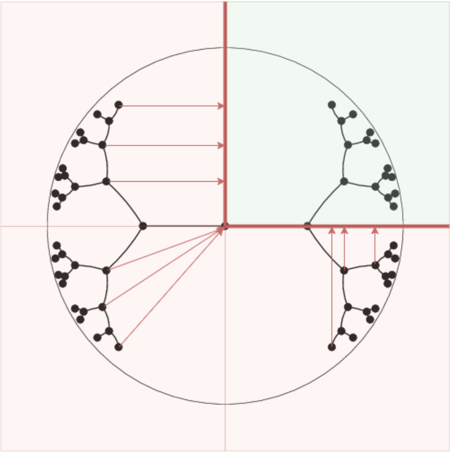


Intuition: Non-linearity

- Conventional Euclidean non-linearities are point-wise and make sense
- Hyperbolic analogy breaks the intuition about Hierarchy

Intuition: Non-linearity

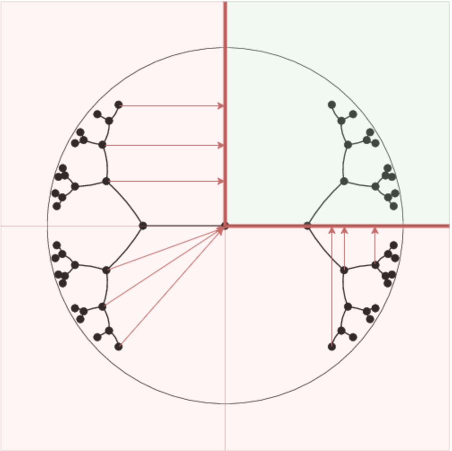
- Conventional Euclidean non-linearities are point-wise and make sense
- Hyperbolic analogy breaks the intuition about Hierarchy



Intuition: Non-linearity

- Conventional Euclidean non-linearities are point-wise and make sense
- Hyperbolic analogy breaks the intuition about Hierarchy

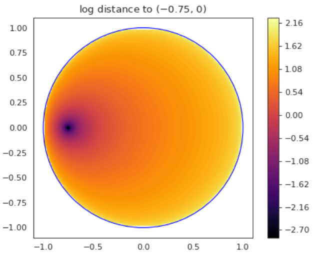
Non-linearity is still uncovered question in hyperbolic deep learning



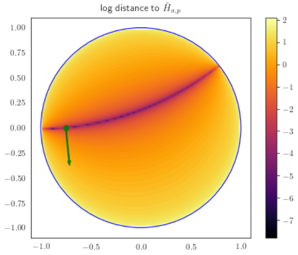
Other non-linearities

Yet not solving all the cases some operations are still useful:

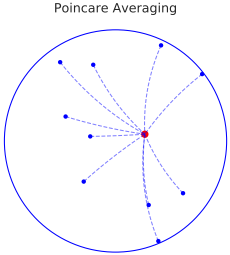
- Distance to a point
- Distance to hyperplane
- Einstein midpoint



Distance to a point



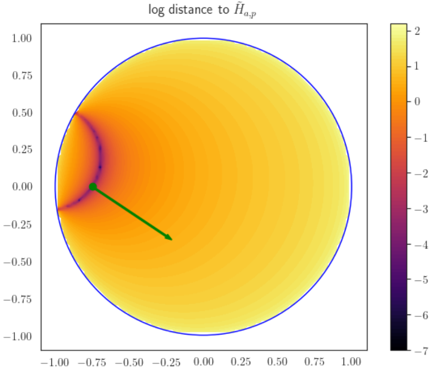
Distance to a hyperplane



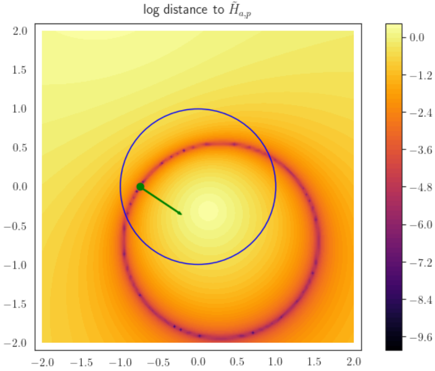
Einstein midpoint

Spoiler: we are not limited to negative curvature

Gyrovector formalism extends to positive curvature manifolds, spheres



Negative Curvature



Positive Curvature

Log Distance to a hyperplane

Where it actually works?

Graph Embeddings

Graph topology is complex, we may desire to numerically represent nodes. It is a good baseline to estimate intristic dimensionality.

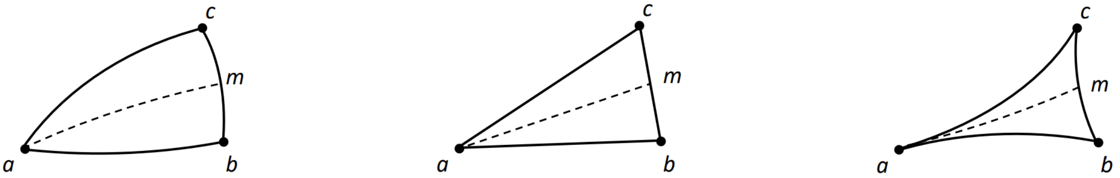
Algorithm

1. Run Dijkstra to get graph distances $\{d_{ij}\}$ between nodes
2. Initialize nodes with random embeddigns $\{e_i\} \in \mathcal{M}$
3. Minimize average distortion $\mathbb{E}_{e_i, e_j \sim G}(\frac{d_{ij}}{d(e_i, e_j)} - 1)^2$

$\mathcal{M}$ is some smooth manifold with defined distance $x, y \in \mathcal{M} \to d(x, y)$. $\mathcal{M}$ may be simply euclidean

Estimating Curvature ξ

Curvature of data may be estimated using triangle



$$\xi_M(a, b, c) := d_M(a, m)^2 + d_M(b, c)^2/4 - (d_M(a, b)^2 + d_M(a, c)^2)/2,$$

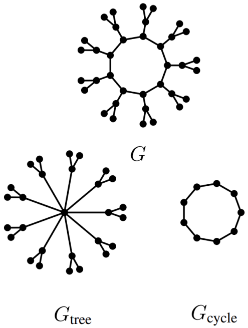
where m is geodesic midpoint, For graphs we can't compute m and use $\xi_G(m; b, c; a) = \frac{1}{d_G(a,m)}\xi(a, b, c)$ given some reference node a. Finally

$$\xi_G(m; b, c) = \frac{1}{|V| - 1} \sum_{a \neq m} \xi_G(m; b, c; a)$$

Estimating Curvature

$$\xi_G(m; b, c) = \frac{1}{|V| - 1} \sum_{a \neq m} \xi_G(m; b, c; a)$$

- Correctly detects cycles, trees,
- The curvature is zero for lines, positive for cycles, and negative for tree
- Easy to calculate



Experimental Results

A. Gu 2019

	Cities	CS PhDs		Power		Facebook	
	$\lvert V\rvert=312$	$\lvert V\rvert=1025, \lvert E\rvert=1043$		$\lvert V\rvert=4941, \lvert E\rvert=6594$		$\lvert V\rvert=4039, \lvert E\rvert=88234$	
	D_{avg}	D_{avg}	mAP	D_{avg}	mAP	D_{avg}	mAP
$\mathbb{E}^{10}$	0.0735	0.0543	0.8691	0.0917	0.8860	0.0653	0.5801
$\mathbb{H}^{10}$	0.0932	0.0502	0.9310	0.0388	0.8442	0.0596	0.7824
$\mathbb{S}^{10}$	0.0598	0.0569	0.8329	0.0500	0.7952	0.0661	0.5562
$(\mathbb{H}^5)^2$	0.0756	0.0382	0.9628	0.0365	0.8605	0.0430	0.7742
$(\mathbb{S}^5)^2$	**0.0593**	0.0579	0.7940	0.0471	0.8059	0.0658	0.5728
$\mathbb{H}^5 \times \mathbb{S}^5$	0.0622	0.0509	0.9141	**0.0323**	0.8850	**0.0402**	0.7414
$(\mathbb{H}^2)^5$	0.0687	**0.0357**	0.9694	0.0396	0.8739	0.0525	0.7519
$(\mathbb{S}^2)^5$	0.0638	0.0570	0.8334	0.0483	0.8818	0.0631	0.5808
$(\mathbb{H}^2)^2 \times \mathbb{E}^2 \times (\mathbb{S}^2)^2$	0.0765	0.0391	0.8672	0.0380	0.8152	0.0474	0.5951
Best model	$\mathbb{S}^5_{1.0} \times \mathbb{S}^5_{1.1}$	$\mathbb{H}^2_{.3} \times \mathbb{H}^2_{.6} \times \mathbb{H}^2_{1.5} \times (\mathbb{H}^2_{1.2})^2$		$\mathbb{H}^5_{3.4} \times \mathbb{S}^5_{12.6}$		$\mathbb{H}^5_{0.3} \times \mathbb{S}^5_{3.5}$	
D_{avg} **improvement over single space**	0.8%	28.89%		16.75%		32.55%	

Reproduced Results

These results we possible to reproduce.

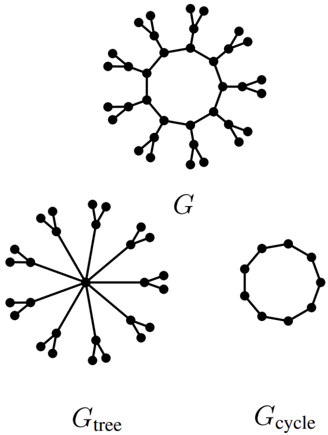
- Stereographic model was used
- Curvature sign was learn and not predefined
- Several approaches for optimization were used

Optimization way	$(\mathbb{U}^2) \times 5$	$(\mathbb{U}^5) \times 2$
Euclidean (20k)	0.0066	0.00715
Euclidean (20k) + Hyperbolic (20k)	0.00564	0.00539
Hyperbolic (20k)	0.00850	0.02255
Tangent Hyperbolic (20k)	0.00747	**0.00246**

Results for Facebook dataset D_{avg}

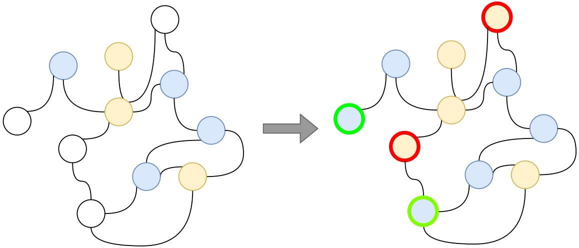
Takeouts

- Curvature may be inferred from data
- Mixed curvature is experimentally always better in graph representations
- Results appeared to be reproducible on real data
- **Nonlinear geometry captures global interactions well**

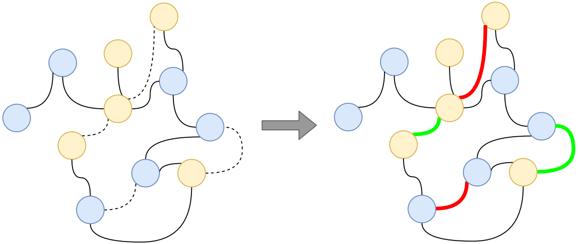


Node classification & Link prediction

These tasks are fundamentally different



Node Classification



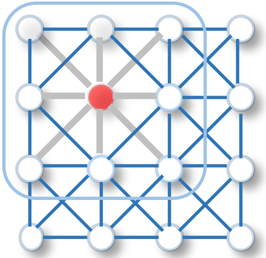
Link Prediction

Node information is important

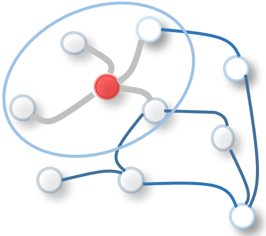
Global structure is important

Global structure is the thing where geometry might be important

Graph Neural Networks



(a) 2D Convolution. Analogous to a graph, each pixel in an image is taken as a node where neighbors are determined by the filter size. The 2D convolution takes the weighted average of pixel values of the red node along with its neighbors. The neighbors of a node are ordered and have a fixed size.



(b) Graph Convolution. To get a hidden representation of the red node, one simple solution of the graph convolutional operation is to take the average value of the node features of the red node along with its neighbors. Different from image data, the neighbors of a node are unordered and variable in size.

Figure: 2D Convolution vs. Graph Convolution. arxiv:1901.00596

Hyperbolic Graph CNNs

Following I. Chami, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and input Euclidean features $(\mathbf{x}^{0,E})_{i\in\mathcal{V}}$, the first layer of HGCN maps from Euclidean to hyperbolic space.

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

$$\mathbf{y}_i^{\ell,H} = \mathrm{AGG}_c(\mathbf{h}^{\ell,H})_i \qquad \text{(attention-based neighborhood aggregation)} \tag{2}$$

$$\mathbf{x}_i^{\ell,H} = \sigma^{\otimes_c}(\mathbf{y}_i^{\ell,H}) \qquad \text{(non-linear activation with different curvatures)} \tag{3}$$

where c is negative hyperbolic curvatures. As $c \to 0$ HGCN recovers Euclidean GCN

Like Euclidean GCN, at each layer HGCN transforms and aggregates neighbour's embeddings applying some transform σ in tangent space.

Hyperbolic Graph CNNs: Results

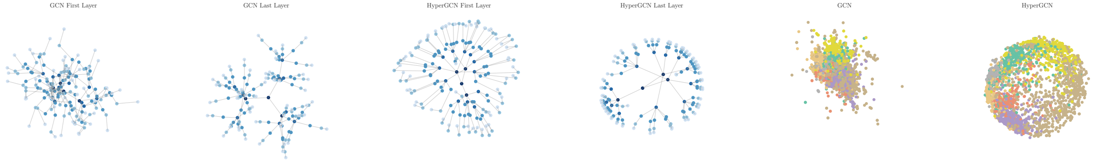


Figure: Left to Right:
1) GCN First Layer; **2)** GCN Last Layer;
3) HyperGCN First Layer, **4)** HyperGCN Last Layer;
5) GCN; **6)** HyperGCN

The embeddings for HyperGCN make more sense. However, for node classification they are **marginally above baselines** and not state-of-the art on real datasets.

Relative Improvements

Several papers present works "improving" graph related tasks. But the improvement is hard to say significant.

Reference	Node classification	Link Prediction
G. Bachmann	Rare ↑	NA
I. Chami	Rare ↑	Common ↑
A. Lou	NA	Common ↑

The result go along with estimated graph hyperbolicity ξ and improvement is correleted with ξ. Some tasks like graph regression and classification are not yet well covered.

This supports the hypothesis that exotic manifolds are useful to capture nontrivial relations.

Language Models

Language models a priori have hierarchical structure. Sentence structure and word relations are good arguments for that.

How does non-Euclidean geometry help here? Where global relations are important?

GloVe model

Given

- $P(j|i) = P_{ij} = X_{ij}/X_i$ – co-occurrence matrix
- w_i, b_i – learnable word embeddings
- $\tilde{w}_i, \tilde{b}_i$ – learnable context embeddings

$$J = \sum_{i,j} f(X_{ij})(-h(d(w_i,\tilde{w}_j)) - b_i - \tilde{b}_j - \log X_{ij})^2 \to \min_{w,b,\tilde{w},\tilde{b}}$$

- $h(x) = x^2/2,\ w_i, \tilde{w}_i \in \mathbb{R}^d$ – GloVe
- $h(x) = \cosh^2(x),\ w_i, \tilde{w}_i \in \mathbb{D}^d$ – Poincare GloVe

Poincaré GloVe: Word Similarity

Experiment name	RareWord	WordSim	SimLex	SimVerb	MC	RG
100D Vanilla GloVe	0.3798	0.5901	0.2963	0.1675	0.6524	0.6894
100D Vanilla GloVe w/ init trick	0.3787	0.5668	0.2964	0.1639	0.6562	0.6757
100D Poincaré GloVe $h(x) = \cosh^2(x)$, w/ init trick	0.4187	0.6209	**0.3208**	**0.1915**	0.7833	0.7578
50x2D Poincaré GloVe $h(x) = \cosh^2(x)$, w/ init trick	**0.4276**	**0.6234**	0.3181	0.189	**0.8046**	**0.7597**
50x2D Poincaré GloVe $h(x) = x^2$, w/ init trick	0.4104	0.5782	0.3022	0.1685	0.7655	0.728

Global structure is important in finding relations, the success is explainable

How to make it work?

- Embeddigns are constructed based on co-occurence graph
- Co-occurences are transformed with $h = cosh^2(\cdot)$ to obtain better metric space
- Isomtries should be fixed via init as only distances matter

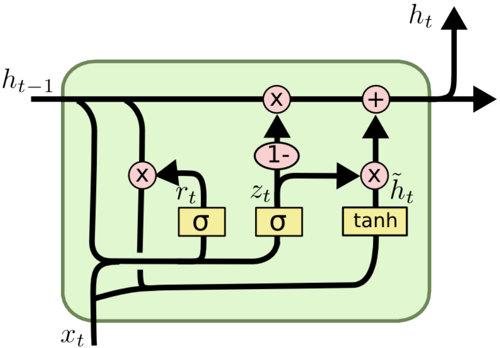
Learned hierarchies for words. Lower is more specific

Hyperbolic (Recurrent) Neural Networks

We have word embeddings, can we use them for language tasks?

Hyperbolic (Recurrent) Neural Networks

Euclidean GRU is a recurrent neural network with time-warping invariance property (C. Tallec - 2018). **In hyperbolic version operations are lifted into hyperbolic space.**



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Hyperbolic (Recurrent) Neural Networks: Results

Sequence classification task on pretrained Hyperbolic Embeddings, best (F1 score) marked in **bold**

WordNet subtree	Model	D = 2	D = 3	D = 5	D = 10
animal.n.01 3218 / 798	Hyp	**47.43 ± 1.07**	**91.92 ± 0.61**	**98.07 ± 0.55**	**99.26 ± 0.59**
	Eucl	41.69 ± 0.19	68.43 ± 3.90	95.59 ± 1.18	**99.36 ± 0.18**
	$\log_0$	38.89 ± 0.01	62.57 ± 0.61	89.21 ± 1.34	98.27 ± 0.70
group.n.01 6649 / 1727	Hyp	**81.72 ± 0.17**	**89.87 ± 2.73**	**87.89 ± 0.80**	**91.91 ± 3.07**
	Eucl	61.13 ± 0.42	63.56 ± 1.22	67.82 ± 0.81	**91.38 ± 1.19**
	$\log_0$	60.75 ± 0.24	61.98 ± 0.57	67.92 ± 0.74	**91.41 ± 0.18**
worker.n.01 861 / 254	Hyp	**12.68 ± 0.82**	**24.09 ± 1.49**	**55.46 ± 5.49**	**66.83 ± 11.38**
	Eucl	10.86 ± 0.01	22.39 ± 0.04	35.23 ± 3.16	47.29 ± 3.93
	$\log_0$	9.04 ± 0.06	22.57 ± 0.20	26.47 ± 0.78	36.66 ± 2.74
mammal.n.01 953 / 228	Hyp	**32.01 ± 17.14**	**87.54 ± 4.55**	**88.73 ± 3.22**	**91.37 ± 6.09**
	Eucl	**15.58 ± 0.04**	44.68 ± 1.87	59.35 ± 1.31	77.76 ± 5.08
	$\log_0$	13.10 ± 0.13	44.89 ± 1.18	52.51 ± 0.85	56.11 ± 2.21

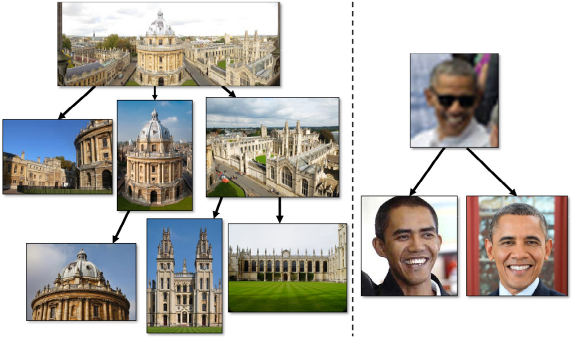
Takeouts so far

1. If we need global structure, we better use non-euclidean space
2. If we use non-euclidean space we should remove isometries
3. It is possible to use hierarchical embeddings in neural networks (caution!)

Images

V. Khrulkov et al. show results for few shot classification task. They show significant improvements for:

- Few shot learning
  (Mini ImageNet, CaltechBirds, etc.)
- Person reidentification
  (market-1501, DukeMTMC-reID)



Poincare VAE

E. Mathieu, 2019

Encoder

$$x \mapsto z \sim \mathcal{N}_{\mathcal{P}}(\mu(x), \sigma^2(x))$$

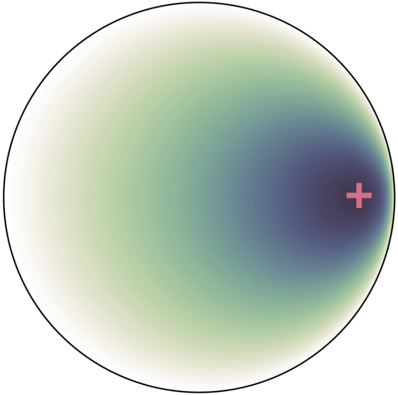
Regular network, reparametrization is implicit (Figurnov, 2018)

Decoder

$$z \mapsto \hat{x} \sim \mathcal{N}(\mu(z'), \sigma^2(z'))$$

$$z' = (H_{a_1,b_1}(z), \dots, H_{a_k,b_k}(z))$$

The first layer of neural network is distance to hyperplanes to respect hierarchy

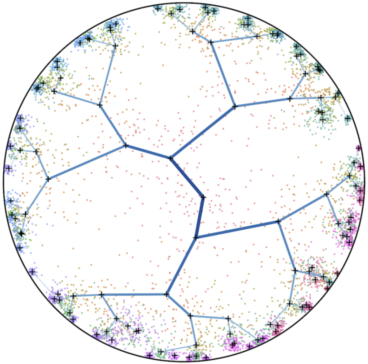


Poincare Normal distribution density

Poincare VAE

Another positive sign it works for images was presented in Poincare VAE model. Datasets are toy: MNIST, Synthetic

Dataset/NLL	VAE	$\mathcal{P}$-VAE
Synthetic	$56.6_{\pm 0.2}$	$\mathbf{55.6}_{\pm \mathbf{0.1}}$
MNIST	$97.6_{\pm 0.2}$	$\mathbf{97.0}_{\pm \mathbf{0.1}}$



Toy dataset embeddigns and hierarchy

Conclusions

1. If we need global structure, we better use non-euclidean space
2. If we use non-euclidean space we should remove isometries
3. It is possible to use hierarchical embeddings in neural networks (with caution!)
4. Tools to work with hyperbolic neural networks are yet limited