Hyperbolic Space in Machine Learning

Andreea Muşat [1]

Abstract

The linear structure of the Euclidean space has commonly been used for learning data representations, often as a mathematical commodity and without matching the real properties of the data. This paper reviews the recent machine learning techniques using the hyperbolic space, a manifold of constant negative curvature with tree-like properties which is well-suited for handling hierarchical and graph-structured data.

1. Introduction

Due to its convenient vectorial structure and intuitive properties, the Euclidean space is the implicit choice in machine learning applications. For signals such as video, audio or images, this assumption matches the underlying data structure and performance on several tasks such as image classification [18], image generation [30], speech generation [27] and reinforcement learning [23] has been particularly improved using standard Euclidean deep learning techniques.

However, certain data types are either non-Euclidean or their inherent characteristics cannot be properly captured when assuming a flat geometry [7], [17]. For example, the latent hierarchical structure often found in symbolic datasets is not accounted for [25] and embedding tree-like data with low distortion in a Euclidean space requires an untractable embedding dimensionality [10] (Fig. 1). Moreover, properties such as clustering and power law distributed degrees, which often characterize real world networks, must be explicitly modelled [17]. By contrast, all these properties arise naturally when assuming an underying hyperbolic geometry.

This paper aims to provide a comprehensive introduction to the use of hyperbolic space in machine learning. In Section 2, we informally review some fundamental differential geometry concepts and present the two hyperbolic space models commonly used in machine learning, namely the Poincaré and Lorentz models. Different algorithm adaptations are presented in Section 3 and their benefits and applications are discussed in Section 4. Finally, Section 5 presents conclusions and future directions of work.

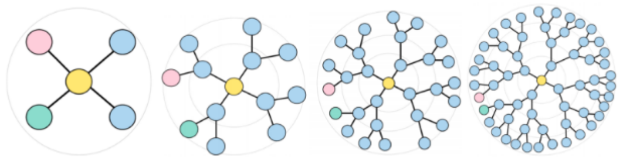

Figure 1. Perfect tree embeddings should preserve the shortest path distance. As the depth increases, the Euclidean embeddings of the green and pink nodes get closer, although their shortest path increases. Source [3]

2. Background

2.1. Rudiments of Riemannian Geometry

A k-dimensional *manifold* $\mathcal{M}$ is a space that locally behaves like $\mathbb{R}^k$ (intuitively, a generalization of a surface in $\mathbb{R}^3$). The *tangent space* at a point $x \in \mathcal{M}$, denoted by $T_x\mathcal{M}$, is the linear subspace spanned by the tangent vectors at x. The disjoint union of tangent spaces of $\mathcal{M}$ is called the *tangent bundle* of $\mathcal{M}$ and is denoted by $T\mathcal{M} = \bigsqcup_{x \in \mathcal{M}} T_x M$.

A *Riemannian manifold* $(\mathcal{M}, g)$ is a real, smooth manifold $\mathcal{M}$ equipped with a positive definite inner product g_x on each tangent space, $g_x : T_xM \times T_xM \mapsto \mathbb{R}$ varying smoothly with x, where $g_x(u,v) = \langle u, v\rangle_x = u^T g_x v$. The family g_x of inner products is called a *Riemannian metric*. It induces a *norm* $\|\cdot\|_x : T_x\mathcal{M} \mapsto \mathbb{R}$ with $\|v\|_x = \sqrt{g_x(v,v)}$ and naturally, it allows for the notion of angle to be defined, $\theta(u,v) = \arccos \frac{g_x(u,v)}{\|u\|_x \|v\|_x}, \forall u, v \in T_x\mathcal{M}$. Two Riemannian metrics g and $\bar{g}$ are called conformal if they induce the same angles, i.e. $\exists \lambda_x > 0$ on $\mathcal{M}$ s.t. $g_x = \lambda_x \bar{g}_x, \forall x \in \mathcal{M}$.

Furthermore, the *length of a smooth curve* on manifold, $\gamma : [0,1] \mapsto \mathcal{M}$, is defined as $L(\gamma) = \int_0^1 \sqrt{g_{\gamma(t)}(\dot{\gamma}(t), \dot{\gamma}(t))}dt$, where $\dot{\gamma}$ represents the derivative of the curve (or, physically, its velocity). The distance between two given points x and y is the length of the shortest curve having x and y as its endpoints: $d(x,y) = \inf_\gamma L(x,y)$, where $\gamma : [0,1] \mapsto \mathcal{M}$ with $\gamma(0) = x, \gamma(1) = y$. In this case γ is called a *geodesic*.

An *exponential map* (Fig. 2) $\exp_x : U \subset T_x\mathcal{M} \mapsto \mathcal{M}$ takes a point v in the tangent space and maps it to a point in the manifold $\exp_x(v)$ by moving a unit length along the geodesic γ starting at x and having direction $\dot{\gamma}(0) = v$. The *logarithmic map* $\log_x : \mathcal{M} \mapsto T_x\mathcal{M}$ is the inverse of the exponential map. *Parallel transport* (Fig. 2)

$\Gamma_{g\mapsto h} : T_g\mathcal{M} \mapsto T_h\mathcal{M}$ connects the geometries of nearby points by moving vectors $u \in T_g\mathcal{M}$ between tangent spaces $T_g\mathcal{M}$ and $T_h\mathcal{M}$ along geodesics. Finally, the *Riemannian gradient* of a function $f : \mathcal{M} \mapsto \mathbb{R}$ is the unique tangent vector $\nabla_R f(x)$ such that the directional derivative in direction η can be written as: $Df(x)[\eta] = g_x(\nabla_R f(x), \eta)$. Expanding this, we can deduce that: (1) $\nabla_R f(x) = (g_x^{-1})^T Df(x)$.

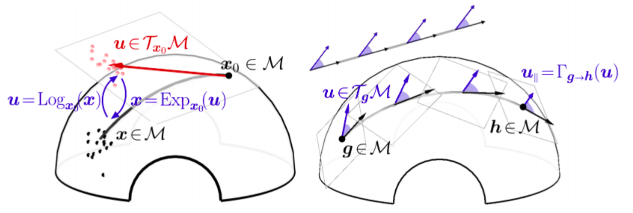
$\boldsymbol{u}\in\mathcal{T}_{\boldsymbol{x}_0}\mathcal{M}$
$\boldsymbol{x}_0 \in \mathcal{M}$
$\boldsymbol{u}=\mathrm{Log}_{\boldsymbol{x}_0}(\boldsymbol{x})$
$\boldsymbol{x}=\mathrm{Exp}_{\boldsymbol{x}_0}(\boldsymbol{u})$
$\boldsymbol{x}\in\mathcal{M}$
$\boldsymbol{u}\in\mathcal{T}_{\boldsymbol{g}}\mathcal{M}$
$\boldsymbol{u}_{\parallel}=\Gamma_{\boldsymbol{g}\to\boldsymbol{h}}(\boldsymbol{u})$
$\boldsymbol{h}\in\mathcal{M}$
$\boldsymbol{g}\in\mathcal{M}$

Figure 2. **Left**: exponential and logarithmic maps, mapping points from the tangent space to the manifold and viceversa. **Right**: parallel transport, moving vectors between tangent spaces by preserving inner products in the tangent spaces. Source: [8]

2.2. The Poincaré ball model

The Poincaré ball model fits the whole geometry in a unit-norm ball: $\mathbb{D}^n = \{x \in \mathbb{R}^n | \|x\|_2 < 1\}$. Because of this, it is ideal for visualizations in small dimensions (Fig. 3 left, middle). Its Riemannian metric tensor is given by $g_x^{\mathbb{D}} = \lambda_x^2 g^E$, where λ_x the conformal factor, (2) $\lambda_x = \frac{2}{1-\|x\|^2}$ and $g^E = I$ is the Euclidean metric tensor. The induced distance function is $d_{\mathbb{D}}(x, y) = \cosh^{-1}(1 + \frac{1}{2}\lambda_x\lambda_y\|x - y\|_2^2), \forall x, y \in \mathbb{D}^n$. Towards the boundary of the disk, the distances grow exponentially, which makes geodesics be curved in that region and straight near the origin (Fig. 3, middle). Because of the bounded norm, we cannot inherit the vectorial structure of $\mathbb{R}^n$. We introduce the framework of gyrovector spaces and the Möbius addition $\oplus$ and scalar multiplication $\otimes$ in Appendix A. Then, we can write the exponential and logarithmic maps in closed form as follows: $\exp_x(v) = x \oplus \left(\frac{v}{\|v\|}\tanh(\frac{\lambda_x\|v\|}{2})\right)$ and $\log_x(v) = \frac{2}{\lambda_x}\tanh^{-1}(\| - x \oplus y\|)\frac{-x\oplus y}{\|-x\oplus y\|}$. Parallel transport of a vector $v \in T_0\mathbb{D}^n$ to the tangent space $T_x\mathbb{D}^n$ is given by: $P_{0\mapsto x}(v) = \log_x(x \oplus \exp_0(v)) = \frac{\lambda_0}{\lambda_x}v$ [12].

2.3. The Lorentz hyperboloid model

Alternatively, the hyperbolic space of negative curvature K can be defined as the Riemannian manifold $\mathbb{H}_K^n = \{x \in \mathbb{R}^{n+1} | \langle x, x\rangle_{\mathcal{L}} = \frac{1}{K}, x_0 > 0, K < 0\}$, where the Lorentzian inner product is $\langle x, y\rangle_{\mathcal{L}} = -x_0y_0 + \sum_{i=1}^n x_iy_i$. The diffeomorphism $p : \mathbb{H}_K^n \mapsto \mathbb{D}^n, p(x_0, x_1, \cdots x_n) = \frac{(x1,\cdots x_n)}{x_0+1}$ maps the points in the Lorentz model into the Poincaré ball, thus making the two models equivalent (Fig. 3, right).

The Riemannian metric tensor $g_x^{\mathbb{H}}$ is the diagonal matrix with (3) $(g_x^{\mathbb{H}})_{00} = -1$ and $(g_x^{\mathbb{H}})_{ii} = 1, \forall i \neq 0$, which is its own inverse, thus making it an appealing choice for optimization (cf. (1)). The induced distance function is: $d(x, y)_{\mathcal{K}} = \frac{1}{\sqrt{-K}}\cosh^{-1}(-K\langle x, y\rangle_{\mathcal{L}})$. Analogously to the sphere, the tangent space at a point x consists of all the vectors v that are perpendicular to x, i.e. $T_x\mathbb{H}_K^n = \{v \in \mathbb{R}^{n+1} | \langle v, x\rangle_{\mathbb{L}} = 0\}$. The exponential map is (4) $\exp_x^K(v) = \cosh\left(\frac{\|v\|_{\mathcal{L}}}{R}\right)x + \sinh\left(\frac{\|v\|_{\mathcal{L}}}{R}\right)\frac{Rv}{\|v\|_{\mathcal{L}}}$, where $R = \frac{1}{\sqrt{-K}}$ and the logarithmic map is $\log_x^K(y) = \frac{\cosh^{-1}\alpha}{\sqrt{\alpha^2-1}}(y - \alpha x)$, where $\alpha = K\langle x, y\rangle_{\mathcal{L}}$. Finally, parallel transport of $v \in T_x\mathbb{H}_K^n$ to $v' \in T_y\mathbb{H}_K^n$ is given by $P_{x\mapsto y}^K(v) = v + \frac{\langle y,v\rangle_{\mathcal{L}}}{R^2-\langle x,y\rangle_{\mathcal{L}}}(x + y)$.

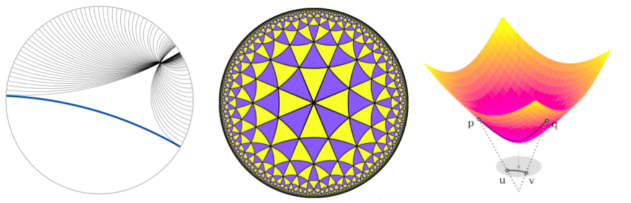

Figure 3. **Left**: multiple hyperbolic parallel lines in the Poincaré disk parallel to the blue line and passing through a point (source). **Middle**: hyperbolic tiling with identical triangles in the Poincaré disk model - note how the perceived triangle area gets smaller close to the boundary (source). **Right**: diffeomorphic mapping of points p and q (and the line between them) in the hyperboloid model (in pink) to their Poincaré disk projections u and v (in gray).

3. Methods

3.1. Riemannian Optimization

Most machine learning problems can be formulated as a minimization problem $w^* = \min_w C(w) = \min_w \mathbb{E}_x Q(x, w)$, where w are real parameters, C represents a cost function and Q a loss function. Stochastic Gradient Descent (SGD) is a popular technique for iteratively solving for w^* by using the following update rule: $w^{(t+1)} = w^{(t)} - \alpha_t g^{(t)}$, where α_t is the learning rate at time t and $g^{(t)}$ is the gradient, which is approximated using a batch of data. For parameters belonging to a smooth manifold $w \in \mathcal{M}$, a generalized algorithm was proposed in [5]:

$$w^{(t+1)} = \exp_{w^{(t)}}(-\alpha g^{(t)})) \approx R_{w^{(t)}}(-\alpha g^{(t)}),$$

where $g^{(t)}$ is now the Riemannian gradient (1). $R_{w^{(t)}}$, named *retraction*, is a first order approximation of the exponential map at $w^{(t)}$. Intuitively, the exp operator is required, as it projects the parameters back onto the manifold $\mathcal{M}$. Note that the procedure is equivalent to SGD when $\mathcal{M}$ is a Euclidean space, as the exponential map in this case is $\exp_x(v) = x + v$. ADAM and ADAGRAD have also been adapted to the Riemannian setting [4]. The main challenge here is that the update rule must be defined intrinsically. The proposed solution treats each dimension of $w_i^{(t)}$ as a separate coordinate, assuming that $\mathcal{M}$ can be written as a *product manifold*, i.e. $\mathcal{M} = \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$

and $w_i^{(t)} \in \mathcal{M}_i, \forall i$. For example, the Euclidean ADAGRAD rule $w_i^{(t+1)} = w_i^{(t)} - \alpha \frac{g_i^{(t)}}{\sqrt{\sum_{k=1}^{t}(g_i^{(k)})^2}}$ is adapted into RADAGRAD as follows:

$$w_i^{(t+1)} = \left(\exp_{w_i^{(t)}}\left(-\alpha \frac{g_i^{(t)}}{\sqrt{\sum_{k=1}^{t} \|g_i^{(k)}\|^2_{x_i^{(k)}}}}\right)\right)_i$$

The generalization of ADAM and convergence guarantees are presented in detail in [4].

The interest in non-Euclidean machine learning has also motivated the development of specialized packages to support Riemannian optimization and manifold operations. Tools such as PyManOpt [33], Geoopt [16] and GeomStats [22] can be used for general-purpose Riemannian optimization in Python. Geoopt, however, is integrated in PyTorch and it provides several models of the hyperbolic space, as well as the basic operations on these manifolds.

3.2. Hyperbolic Embeddings

As a first application, we review the use of Poincaré embeddings for datasets exhibiting hierarchical structure [25], such as WordNet [21]. Given a set of symbols $\mathcal{S} = \{x_i\}_{i=1}^n$, the goal is to obtain Poincaré embeddings $\Theta = \{\theta_i\}_{i=1}^n$ in an unsupervised manner, such that (1) their similarity is reflected in the embedding space and (2) the latent hierarchical structure is implicitly captured in the embedding space without specifying it. Assuming a given problem-dependent loss function, the optimization problem is $\Theta^* = \arg\min_\Theta \mathcal{L}(\Theta)$ such that $\|\theta_i\| < 1, \forall \theta_i \in \Theta$. This is solved using RSGD with $R_{\theta^{(t)}}(v) = \theta^{(t)} + v$ as a retraction, yielding a rule similar to natural gradient descent [1]:

$$\theta^{(t+1)} = \pi\big(\theta^{(t)} - \alpha_t (g^{\mathbb{D}}_{\theta^{(t)}})^{-1} \nabla_E \mathcal{L}(\theta^{(t)})\big)$$

where $(g^{\mathbb{D}}_{\theta^{(t)}})^{-1}$ can be computed using Eq. (2), ∇_E is the Euclidean gradient and π represents a projection that guarantees that $\theta \in \mathbb{D}^n$, i.e. $\pi(\theta) = \frac{\theta}{\|\theta\|} - \epsilon$ with $\epsilon = 10^{-5}$ when $\|\theta\| >= 1$ and otherwise $\pi(\theta) = \theta$. Building upon this, [26] proposes a method for learning embeddings using the Lorentz model, improving the quality of embeddings, especially in low dimensions. Similar to before, the Riemannian gradient is computed using Eq. (1) and together with the involutory metric tensor Eq. (3) and the closed form of the exponential map Eq. (4), the RSGD algorithm can be efficiently applied.

3.3. Hyperbolic Neural Networks

A step further is made by introducing hyperbolic neural network building blocks, making it possible to adapt common architectures [12]. The key element here is the *Möbius version* of a function. Given $f : \mathbb{R}^n \mapsto \mathbb{R}^m$, its Möbius version, $f^{\otimes} : \mathbb{D}^n \mapsto \mathbb{D}^m$, is computed as follows:

$$f^{\otimes}(x) = \exp_0(f(\log_0(x))),$$

where $\log_0 : \mathbb{D}^n \mapsto T_{0_n}\mathbb{D}^n$ and $\exp_0 : T_{0_m}\mathbb{D}^m \mapsto \mathbb{D}^m$. Pointwise non-linearities can be adapted by applying their Möbius version element-wise. Also, it is straightforward to generalize the matrix-vector multiplication operation. If $M : \mathbb{R}^n \mapsto \mathbb{R}^m, x \in \mathbb{D}^n$ and $Mx \neq 0$, then the Möbius matrix-vector multiplication is:

$$M^{\otimes}(x) \overset{\text{def}}{=} M \otimes x = \tanh\left(\frac{\|Mx\|}{\|x\|} \tanh^{-1}(\|x\|)\right) \frac{Mx}{\|Mx\|}$$

Finally, the Möbius translation of a point $x \in \mathbb{D}^n$ with the bias $b \in \mathbb{D}^n$ is given by: (5) $\exp_x(P_{0 \mapsto x}(\log_0(b))) = x \oplus b$.

Using these primitives, a feedforward layer with inputs $x \in \mathbb{D}^n$ and having weights $W \in \mathbb{D}^{m \times n}$, bias $b \in \mathbb{D}^m$ and non-linearity φ will simply be $\varphi(W \otimes x \oplus b)$. Furthermore, architectures such as RNN, which compute a hidden state $h^{(t+1)} = \varphi(Wh^{(t)} + Ux^{(t)} + b)$ from inputs $x^{(t)}$ and previous state $h^{(t)}$ will have the following hyperbolic version:

$$h^{(t+1)} = \varphi^{\otimes}(W \otimes h^{(t)} \oplus U \otimes x^{(t)} \oplus b),$$

where $\varphi^{\otimes}$ is the Möbius version of the non-linearity.

3.4. Hyperbolic Graph Neural Networks

The ideas in the previous sections have also been extended to graph representation learning. Given a graph $G = (V, E)$, and input node features $X = \{x_v\}_{v \in V} \in \mathbb{R}^{|V| \times d}$, the goal is to learn a function outputting node embedding vectors, i.e. $f : (G, X) \mapsto Z \in \mathbb{R}^{|V| \times d'}$ such that Z captures both the local graph structure and the node features and can be used for downstream task such as node prediction. Graph Convolutional Networks (GCN) [14] consist of multiple layer updating the embeddings $H^{(t)} = \{h_v^{(t)}\}_{v \in V}$ (with $H^{(0)} = X$) by aggregating neighbor information using the adjacency matrix A and the diagonal degree matrix D: (6) $H^{(t+1)} = \varphi\big(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}H^{(t)}W^{(t)}\big)$, where $W^{(t)}$ are the weights used in layer t and φ the non-linearity.

Leveraging the fact that many graph properties can be explained by assuming a hyperbolic geometry, [19] propose a GCN variant that produces hyperbolic node embeddings. By projecting the previous node embeddings $H^{(t)}$ onto the tangent space, the aggregation in Eq. (6) can be directly applied, as all the operands are then Euclidean. In order to prevent the collapse into a fully Euclidean model, the aggregation result is projected back onto the manifold before applying the elementwise non-linearity, yielding the following propagation rule:

$$H^{(t+1)} = \varphi\big(\exp_{x'}(D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}} \log_{x'}(H^{(t)})W^{(t)})\big),$$

where $x' \in \mathcal{M}$. Note that we require $\varphi : \mathcal{M} \mapsto \mathcal{M}$ in order to guarantee that the embeddings will be on the manifold. For example, if using the Poincaré ball model, it suffices to use a norm decreasing activation such as (leaky) ReLU.

Fully hyperbolic graph networks (Fig. 6) are derived by [9] using the Lorentz model. The hyperboloid version of a map f, denoted by $f^{\otimes_K}$, is built by applying the operation in the tangent space, similar to the Möbius version in [12], i.e. $W \otimes^K x = \exp_o^K(W \log_o^K(x))$ with o being the origin point. Moreover, the hyperboloid bias addition is also defined using the notion of parallel transport, but using an Euclidean bias vector: $x \oplus^K b = \exp_x^K(P_{o\mapsto x}(b))$. The resulting embeddings are hyperbolic and the feature transform happens in the hyperbolic space:

$$m_i^{(t+1)} = (W^{(t+1)} \otimes^K h_i^{(t)}) \oplus^K b^{(t+1)}$$

Furthermore, instead of using a mean aggregation in the tangent space, an attention-based mechanism is proposed. Given 2 hyperbolic embeddings, attention weights are computed using a multi-layer perceptron (MLP) applied on the concatenation of their projections in the tangent space at o:

$$\text{AGG}^K(m)_i = \exp_{m_i}^K\Big(\sum_{j\in\mathcal{N}(i)} w_{ij} \log_{m_i}^K(m_j)\Big), \text{ where}$$

$$w_{ij} = \text{SOFTMAX}\Big(\text{MLP}(\log_o^K(h_i) || \log_o^K(h_j))\Big), \forall j \in \mathcal{N}(i)$$

with $\mathcal{N}(i)$ being the set of neighbors of node i. The message passing update of HGCN then reads:

$$h_i^{(t+1)} = \varphi^{\otimes^K}(\text{AGG}^K(m^{(t+1)})_i)$$

3.5. Hyperbolic probability distributions

Probability distributions are ubiquitous in machine learning. However, it is not straightforward to define densities on hyperbolic spaces. For example, of all real distributions with a fixed mean μ and variance σ^2, the Gaussian distribution $\mathcal{N}(x|\mu, \sigma^2)$ is the one with maximum entropy [11]. The Riemannian normal [29] on a manifold $\mathcal{M}$ with mean μ and dispersion σ is defined similarly:

$$\mathcal{N}_{\mathcal{M}}(x|\mu,\sigma^2) = \frac{1}{\xi(\sigma)} \exp\Big(-\frac{d_{\mathcal{M}}^2(x,\mu)}{2\sigma^2}\Big), \sigma > 0,$$

where $\xi(\sigma)$ is a normalization constant and $d_{\mathcal{M}}$ is the induced distance on the manifold $\mathcal{M}$. Note that μ must lie on the manifold, so it is a generalized mean called the Fréchet mean, which for n points $\{x_i\}_{i=1}^n$ is defined as the solution of the optimization problem $\mu^* = \arg\min_{\mu\in\mathcal{M}} \frac{1}{n}\sum_{i=1}^n d(x_i,\mu)^2$ [20]. Other probability distributions are discussed in [6] and [32].

3.6. Hyperbolic normalizing flows

Normalizing flows [28] are generative models that learn an invertible and differentiable function that transforms a simple base distribution into an arbitrary complex distribution. As standard Euclidean normalizing flows are not suitable for modelling data with hierarchical structure, a hyperbolic variant was proposed in [6]. The key element is again juggling between the manifold and the tangent bundle: points from the base distribution defined on $\mathbb{H}_K^n$ are mapped onto $T_o\mathbb{H}_K^n$, where an Euclidean flow is applied before projecting back onto the manifold using the exponential map [6].

4. Discussion

The hyperbolic space can be seen as a continous version of trees and it induces a structural bias: it successfully learns hierarchical relationships and also captures similarities well [25]. In graphs, heterogeneous degree distributions and strong clustering are a result of the negative curvature [17]. The Lorentz model is suited for efficient Riemannian optimization and it avoids the numerical instabilities caused by the fraction in the distance on the Poincaré ball. The strengths of the Poincaré model are its interpretation and visualization properties. However, as the models are diffeomorphic, both strengths can be exploited by transforming points when necessary.

The presented tools have been successfully applied to tasks such as embedding of taxonomies [25], [26], link prediction in networks [25] (Fig. 5), [9], textual entailment [12], as well as graph node prediction [9] (Fig. 7), molecular property prediction [19] and analysis of RNA data [15], yielding results at least as good as their Euclidean counterparts. [9] observe a correlation between the hyperbolicity of the dataset (i.e. how tree-like a graph is) and the performance of hyperbolic methods, especially in low dimensions.

5. Conclusion

In this paper we reviewed several generalizations of machine learning algorithms to Riemannian manifolds of constant negative curvature. The properties of the hyperbolic space have been leveraged in order to learn better representations for hierarchical and graph-structured data and experiments have shown that this structural bias allows for better learning of the underlying structure of the data. Clearly, nature also produces other types of non-Euclidean data [24], so other fields might benefit from advances in learning methods on more complicated manifolds. Recent work is already focusing on architectures that leverage mixed curvature manifolds [32], [13] or general manifolds of constant curvature [3]. As future work, it would also be interesting to see Riemannian geometry tools applied for better understanding of the low-dimensional data manifold [2].

References

[1] Amari, S.-I. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[2] Arvanitidis, G., Hauberg, S., and Schölkopf, B. Geometrically enriched latent spaces. In *arXiv preprint*, 2020.

[3] Bachmann, G., Bécigneul, G., and Ganea, O. Constant curvature graph convolutional networks. In *International Conference on Machine Learning*, pp. 486–496. PMLR, 2020.

[4] Becigneul, G. and Ganea, O.-E. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2018.

[5] Bonnabel, S. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[6] Bose, A. J., Smofsky, A., Liao, R., Panangaden, P., and Hamilton, W. L. Latent variable modelling with hyperbolic normalizing flows. *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[7] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[8] Calinon, S. Gaussians on riemannian manifolds: Applications for robot learning and adaptive control. *IEEE Robotics & Automation Magazine*, 27(2):33–45, 2020.

[9] Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems*, pp. 4868–4879, 2019.

[10] De Sa, C., Gu, A., Ré, C., and Sala, F. Representation tradeoffs for hyperbolic embeddings. *Proceedings of machine learning research*, 80:4460, 2018.

[11] Edwards, S. Elements of information theory, thomas m. cover, joy a. thomas, john wiley & sons, inc.(2006), 2008.

[12] Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in neural information processing systems*, pp. 5345–5355, 2018.

[13] Gu, A., Sala, F., Gunel, B., and Ré, C. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2018.

[14] Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[15] Klimovskaia, A., Lopez-Paz, D., Bottou, L., and Nickel, M. Poincaré maps for analyzing complex hierarchies in single-cell data. *Nature Communications*, 11(1):1–9, 2020.

[16] Kochurov, M., Karimov, R., and Kozlukov, S. Geoopt: Riemannian optimization in pytorch. *arXiv preprint arXiv:2005.02819*, 2020.

[17] Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., and Boguná, M. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

[18] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[19] Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 8230–8241, 2019.

[20] Lou, A., Katsman, I., Jiang, Q., Belongie, S. J., Lim, S., and Sa, C. D. Differentiating through the fréchet mean. PMLR, 2020.

[21] Miller, G. A. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[22] Miolane, N., Mathe, J., Donnat, C., Jorda, M., and Pennec, X. Geomstats: A python package for riemannian geometry in machine learning. *arXiv preprint arXiv:1805.08308*, 2018.

[23] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. 2013. NIPS Deep Learning Workshop 2013.

[24] Nechaev, S. Non-euclidean geometry in nature. *arXiv preprint arXiv:1705.08013*, 2017.

[25] Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pp. 6338–6347, 2017.

[26] Nickel, M. and Kiela, D. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2018.

[27] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[28] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

[29] Pennec, X. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127, 2006.

[30] Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[31] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[32] Skopek, O., Ganea, O.-E., and Bécigneul, G. Mixed-curvature variational autoencoders. In *International Conference on Learning Representations*, 2019.

[33] Townsend, J., Koep, N., and Weichwald, S. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759, 2016.

[34] Ungar, A. A. Hyperbolic trigonometry and its application in the poincaré ball model of hyperbolic geometry. *Computers & Mathematics with Applications*, 41(1-2): 135–147, 2001.

[35] Ungar, A. A. Gyrogroups, the grouplike loops in the service of hyperbolic geometry and einstein's special theory of relativity. *arXiv preprint arXiv:1303.0218*, 2013.

[36] Vermeer, J. A geometric interpretation of ungar's addition and of gyration in the hyperbolic plane. *Topology and its Applications*, 152(3):226–242, 2005.

Appendices

A. Gyrovector spaces [34], [35]

Our goal is to equip the Poincaré ball with a vectorial structure, i.e. with a vector addition operation and a scalar multiplication operation such that their results remain in the Poincaré ball.

For any real vector space V with an inner product $\langle \cdot, \cdot \rangle$, let V_c be the open ball of V of radius c centered in the origin. Then, we define **Möbius addition** (Fig. 4), $\oplus : V_c \times V_c \mapsto \mathbb{R}$, as follows:

$$u \oplus v = \frac{(1 + \frac{2}{c^2}\langle u, v\rangle + \frac{1}{c^2}\|v\|^2)u + (1 - \frac{1}{c^2}\|u\|^2)v}{1 + \frac{2}{c^2}\langle u, v\rangle + \frac{1}{c^4}\|u\|^2\|v\|^2}$$

The Möbius addition satisfies the following properties:
(1) **Identity**: $0 \oplus a = a \oplus 0 = a, \forall a \in V_c$
(2) **Inverse**: $\forall a \in V_c, \exists!$ inverse element $\ominus a \in V_c$ such that $\ominus a \oplus a = a \oplus (\ominus a) = 0$
(3) **Left-cancellation law**: $\forall x, y \in V_c, (\ominus x) \oplus (x \oplus y) = y$

Note that, in general, $\oplus$ is neither associative, nor commutative. For infinitely large c we recover the Euclidean addition:

$$\lim_{c \to \infty} u \oplus v = u + v \text{ and, respectively } \lim_{c \to \infty} V_c = V$$

The **Möbius scalar multiplication** $\otimes : \mathbb{R} \times V_c \mapsto V_c$ is computed using the next equation:

$$r \otimes v = c \tanh\left(r \tanh^{-1} \frac{\|v\|}{c^2}\right) \frac{v}{\|v\|} \text{ if } v \neq 0 \text{ and otherwise } r \otimes 0 = 0$$

It satisfies the following properties:
(1) **Additive Scalar Law**: $\forall r_1, r_2 \in \mathbb{R}, \forall v \in V_c, (r_1 + r_2) \otimes v = r_1 \otimes v \oplus r_2 \otimes v$
(2) **Multiplicative Scalar Law**: $\forall r_1, r_2 \in \mathbb{R}, \forall v \in V_c, (r_1 r_2) \otimes v = r_1 \otimes (r_2 \otimes v)$
(3) **Scalar-Norm Product Law**: $\forall r \in \mathbb{R}, \forall v \in V_c, \|r \otimes v\| = |r| \otimes \|v\|$
(1) n **additions**: $\forall n \in \mathbb{N}^*, \forall v \in V_c,$ then $n \otimes v = \underbrace{v \oplus v \oplus \ldots \oplus v}_{n \text{ times}}$

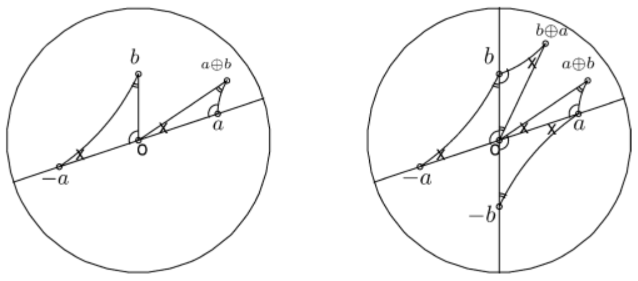

Figure 4. **Left**: Visualization of Möbius addition. The resulting point $a \oplus b$ is obtained by translating the triangle $\triangle(-a, O, b)$ to $\triangle(O, a, a \oplus b)$. **Right**: The Möbius addition is not a commutative operation. Source [36]

B. Hyperbolic architectures and results

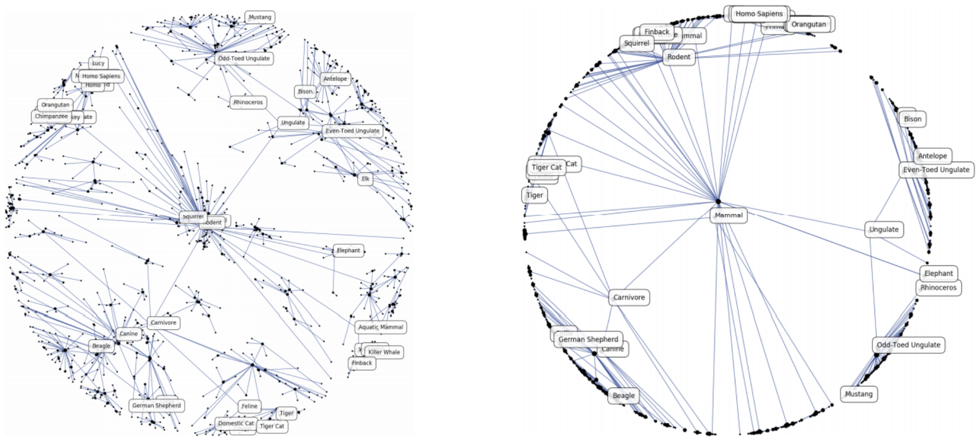

Figure 5. **Left**: Poincaré embeddings of the WORDNET mammal subtree after 20 epochs. **Right**: Final Poincaré embeddings. Note the preserved hierarchical tree structure and the clustering of similar concepts. Source [25]

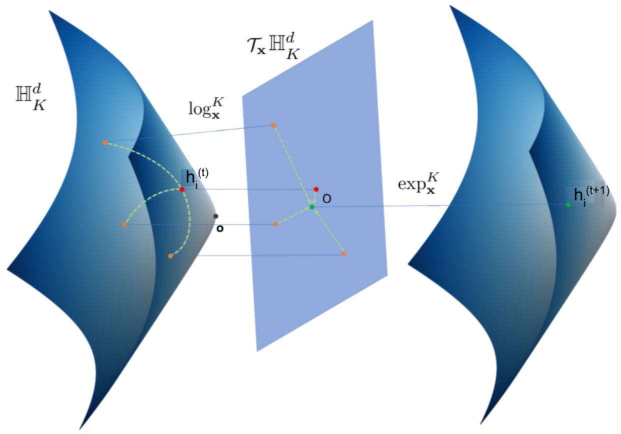

Figure 6. Visualizing HGCN. First, the hyperbolic feature transform is applied on the current embeddings for node i, $h_i^{(t)}$. Using the logarithmic map, these transformed embeddings are then mapped onto the tangent space, where attention-based aggregation is performed (middle) and the result is mapped back onto the hyperboloid via the exponential map (right). Adapted from [9].

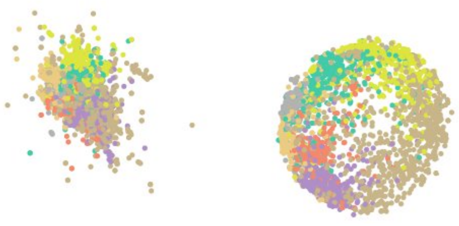

Figure 7. GCN (left) and HGCN (right) embeddings on the CORA dataset [31]. HGCN clearly exhibits better class separation. Source [9]