REPRESENTATION LEARNING IN RIEMANNIAN MANIFOLDS

Henry Steinitz
NYU Courant
New York, NY 10012

Shan-Conrad Wolf
NYU Courant
New York, NY 10012

May 13, 2019

ABSTRACT

Euclidean representation learning describes a class of techniques for learning mappings from a raw input space to Euclidean space that map similar objects to nearby points. The learned Euclidean representations are often useful for other untrained tasks such as analogical reasoning [11]. However, standard Euclidean representation learning techniques break down when one deals with data that is inherently non-Euclidean. The result is high-dimensional, low-quality embeddings. The aim of this report is to illustrate to the reader how the tools of non-Euclidean geometry can be applied to representation learning. To this end, we focus on hyperbolic geometry, which is well-suited to data with inherent hierarchical structure. We first review standard representation learning techniques such as Word2Vec models and variational autoencoders (VAEs), and discuss some of their emergent properties. We then explain the theoretical limitations of embedding into Euclidean space. In particular, Euclidean embeddings require a prohibitively high dimension in order to model distributions with hierarchical structure [13]. This difficulty is inherent to Euclidean space and cannot be circumvented through the use of non-linear embeddings [1]. To overcome these limitations, Nickel *et al.* introduce embeddings into hyperbolic space [6]. We show that hyperbolic space shares structural similarities with trees, making them suitable for representing hierarchical relations. After a brief review of Riemannian optimization [2] and the various models of hyperbolic geometry, we examine at the embedding methodology proposed by Nickel *et al.* [6] [7]. Finally, we look at how VAEs can be adapted to models with hyperbolic latent spaces [5].

1 Introduction

The process of learning good representations for raw input data is fundamental to machine learning. Nearly all supervised learning and reinforcement learning architectures explicitly or implicitly optimise embeddings into a latent feature space, and the unsupervised learning setting is entirely based on discovering such embeddings.

The notion of an *embedding* or a *representation* is somewhat informal. Given an input space $\mathcal{X}$ and a latent space $\mathcal{Z}$, an embedding is typically a function $f : \mathcal{X} \to \mathcal{Z}$. In other cases, we think of embeddings as mappings from X to probability distributions over Z. We often also overload notation by referring to the image of some dataset $\mathcal{D}$ under f as an embedding. Whilst the notion of what constitutes a good embedding of data is not strictly defined, it is generally taken to be one that captures the factors of variation contained in $\mathcal{D}$ that are relevant to the task at hand in such a way that there exist simple functions (such as a shallow neural network) that can map an embedded datapoint to some attribute of that datapoint we would like to know. Throughout this survey, we use the terms embedding and representation interchangably.

Most contemporary machine learning architectures, such as autoencoders, word embeddings, convolutional networks, *et c.*, learn representations in *flat Euclidean space*. While this has been successful for many tasks, there are fundamental

geometric limitations of Euclidean space that make it suboptimal for learning representations with hierarchical or complex network structure [13].

More general spaces called *Riemannian manifolds* have recently been studied in representation learning algorithms. For example, it's natural to picture hyperbolic space (negatively curved Riemannian manifolds) as the continuous analog of trees, which are natural data structures for representing hierarchies. We thus develop the basic theory of Riemannian manifolds and of Riemannian gradient descent, highlighting the special case of hyperbolic geometry. We also collect recent successful applications of non-Euclidean representations including hyperbolic word embeddings [6, 7] and the hyperbolic variational autoencoder [5].

2 Euclidean embeddings

In this section, we give two important methods for obtaining good embeddings into Euclidean space. We then discuss the limitations these methods have when applied to hierarchical data.

2.1 Word embeddings

When dealing with non-numerical data, it is generally necessary to first create a numerical representation of the data before running a machine learning algorithm. Words, being non-numerical, discrete data, are often represented by a one-hot embedding, whereby each word w in a vocabulary of size V is associated with an integer n_w between 1 and V and is encoded by a 'one-hot' vector $(v_w)_i = \mathbb{I}\{i = n_w\}$. Because vocabularies may contain hundreds of thousands or even millions of words, and because one-hot vectors do not contain any linguistic information about the word it encodes, this encoding of words is of limited use. A widely used method of obtaining useful word embeddings is the skip-gram model [11]. A skip-gram model consists of a 2-layer neural network (1 hidden layer) with no non-linearity at the hidden layer, and a softmax non-linearity at the final layer. The dimension of the hidden layer is generally chosen to be somewhere between 50 and 300 [16]. For a given word $w^{(i)}$ in a document, the model is trained to maximise the log-likelihood of the neighbouring words $w^{(i-c)}, \dots, w^{(i-1)}, w^{(i+1)}, \dots, w^{(i+c)}$ under the skip-gram model. These surrounding words make up the *context* of word $w^{(i)}$. The idea behind this is that words with similar meanings occur in similar contexts (known as the distributional hypothesis [18]), and so words with similar meanings should be mapped to similar embeddings. After training a skip-gram model, each word in the vocabulary is assigned the embedding vector obtained by passing its one-hot vector through the first layer of the network.

This general method of creating embeddings has many uses, such as in natural language processing (*e.g.* name-entity recognition [12] and analogical reasoning [11]) and in bioinformatics (*e.g.* protein prediction [17]).

2.2 Variational autoencoders

A variational autoencoder (VAE) is a generative model corresponding to a directed graphical model $Z \to X$. Points are drawn from the latent space according to some (possibly learnt) probability distribution $p_\theta(z)$ and passed through a neural network f_θ to generate a sample x. Since the integral for the marginal likelihood $p(x) = \int_{\mathcal{X}} p_\theta(z)p_\theta(x \mid z)dz$ and the true posterior distribution $p_\theta(z \mid x)$ are intractable for neural networks with non-linear activation, we approximate the true posterior with a distribution $q_\phi(z \mid x)$ defined by a neural network with parameters ϕ. The optimisation procedure then consists of maximising the evidence lower bound (ELBO) of the average log likelihood over θ and ϕ [14]:

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= \log(p_\theta(x^{(i)} \mid z)) + \log\left(\frac{p_\theta(z)}{q_\phi(z \mid x^{(i)})}\right) + \log\left(\frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right) \\
&= \mathbb{E}_q \log(p_\theta(x^{(i)} \mid z)) - \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z)) + \mathrm{KL}(q_\phi(z \mid x^{(i)}) \| p_\theta(z \mid x^{(i)})) \\
&\geq \mathbb{E}_q \log p_\theta(x^{(i)} \mid z) - \mathrm{KL}(q(z \mid x^{(i)}) \| p_\theta(z)) =: \mathrm{ELBO}(x_i; \theta, \phi).
\end{aligned} \tag{1}$$

where $\{x_i\}_{i=1}^N$ is the observed data. The first term in the ELBO then acts as a reconstruction loss whilst the second term acts as a regularisation term. Note that the inequality follows from the non-negativity of the KL-divergence, and that the ELBO equals the log-likelihood if and only if $q_\phi(z \mid x^{(i)})$ is equal to the true posterior $p_\theta(z \mid x^{(i)})$.

The latent spaces of VAEs have often been found to 'factorise' the observed data in the sense of encoding independent factors of variation. For example, a VAE trained on a dataset of images of chairs may have each dimension of its latent space encode a different factor of variation, such as azimuth, chair size, or chair type. [15] provides a good (though perhaps not fully convincing) explanation of this phenomenon in terms of the cost of encoding information.

2.3 Beyond Euclidean embeddings

Despite their successes, Euclidean embeddings are unsuitable for embedding certain types of data. One such type is data with hierarchical structure. A regular tree with branching factor b has $O(b^l)$ nodes at level l, whereas the surface area of a k-sphere of radius R grows as $O(R^k)$. Consequently, an embedding into a $(k+1)$-ball of radius R requires the dimension of the embedding to grow proprtionally to the hierarchy depth of a dataset [13]. This results in computationally impractical embedding sizes, which runs contrary to our general notion of what constitutes a good embedding. In addition to this, it is not clear whether the Euclidean distance between two points is always a reliable indicator of similarity. Indeed, when the data has a tree structure, a better measure of similarity between two entities might be the distance to their most recent common ancestor. In this report, we examine how hyperbolic geometry provides a solution to these problems.

3 Riemannian Geometry

In this section we quickly review concepts from differential geometry needed to understand curvature and perform gradient-based optimization in Riemannian manifolds. For a more complete treatment, we refer the reader to [4].

3.1 Smooth manifolds

A topological manifold M of dimension n is a connected topological space that looks locally like $\mathbb{R}^n$. That is, for each point $p \in M$, there exists a neighborhood U of p that is homeomorphic to an open subset V of $\mathbb{R}^n$. A smooth manifold is a topological manifold in which each point has a neighborhood that can be assigned "local coordinates". These coordinates are given by injective maps $\{\phi_i : U_i \subset \mathbb{R}^n \to M\}$ called charts. The composition of these charts must be smooth. More precisely, a *smooth structure* on a topological manifold M is a collection of homeomorphisms $\{\phi_i : U_i \subset \mathbb{R}^n \to M\}$ such that

1. $\bigcup_i \phi_i(U_i) = M$
2. $\phi_i^{-1} \circ \phi_j$ is smooth for all ϕ_i, ϕ_j with $U_i \cap U_j \neq \emptyset$.

A *smooth manifold* is a topological manifold with a maximal smooth structure. By maximal, we mean that there are no additional maps ϕ_i that can be added to the smooth structure so that it still satisfies conditions 1 and 2. From now on, we'll denote smooth manifolds of dimension n as M^n when we wish to highlight its dimension. Finally, we say that $f : M \to \mathbb{R}^k$ is smooth if $f \circ \phi_i$ is smooth for every chart ϕ_i.

3.2 Tangent spaces

To each point $x \in M$, we can associate a real vector space of dimension n which we intuitively picture as lying tangent to M at x. We call this vector space the *tangent space* of M at x and denote it T_xM. One convenient definition of the tangent space is the following: Let $\alpha : (-1, 1) \to M$ be a curve that passes through $x = \alpha(0) \in M$. The α defines a differential operator on smooth functions $f : M \to \mathbb{R}$ defined as

$$\alpha'(0)(f) = \left.\frac{d(f \circ \alpha)}{dt}\right|_{t=0} \tag{2}$$

Note that these operators are linear. The tangent space T_xM is then the set of differential operators formed by curves α passing x at $t = 0$, and is a vector space of dimension n. A vector field is a map $Y : M \to T_xM, x \mapsto Y_x$. The vector field is smooth if $Y(f)$ (defined pointwise) is smooth for all smooth f. The primary use of the tangent space is to hold differential information about structures on M such as curves and scalar functions.

3.3 Riemannian metrics

A *Riemannian metric* on M is a smoothly varying inner product structure $\langle \cdot, \cdot \rangle_x$ on each tangent space T_xM. By smoothly varying, we mean that, given any two smooth vector fields X, Y, the map

$$\langle X, Y \rangle : M \to \mathbb{R} \tag{3}$$

$$x \mapsto \langle X_x, Y_x \rangle_x \tag{4}$$

is a smooth scalar function on M. The term Riemannian metric is a misnomer, since it isn't a metric in the ordinary sense. However, it can be used to define an ordinary metric on M:

$$d(x, y) = \inf_\gamma \int_0^1 \sqrt{\langle \gamma'(t), \gamma'(t) \rangle_{\gamma(t)}} dt \tag{5}$$

where γ ranges over all smooth curves from $[0, 1]$ to M such that $\gamma(0) = x$ and $\gamma(1) = y$. The corresponding minimising curve is called a *geodesic*.

3.4 Affine connections

Curvature and geodesics depend on structures called *affine connections* and the unique torsion-free affine connection on a Riemannian manifold is called the *Levi-Citiva connection*. We don't give the full definitions here, but chapter 2 of [**?**] is a good resource. Let X, Y be vector fields. Intuitively, a connection ∇_X provides an operation that allows one to differentiate Y with respect to X and produce a third vector field $\nabla_X Y$.

3.5 Curvature

The *Riemannian curvature* R can be viewed as a mapping $V(M) \times V(M) \times V(M) \to V(M)$ given by

$$R(X, Y, Z) = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z.$$

where $V(M)$ is the set of smooth vector fields on M and $[X, Y]$ is the Lie bracket of smooth vector fields X and Y (see [9]). The Riemmanian curvature at x measures the change in the direction of a vector parallely transported in a loop starting at x.

Let $x \in M$ and let σ be a 2-dimensional subspace of $T_x M$. The *sectional curvature* of σ at x is then

$$K(\sigma) = \frac{\langle R(v, w, v), w \rangle}{\sqrt{|v|^2 |w|^2 - \langle v, w \rangle^2}}$$

for any $v, w \in \sigma$. It's straight-forward to show that this notion is well-defined. There are three Riemannian manifolds with constant sectional curvature up to metric scale: spherical space (positive curvature), flat space (0 curvature), and hyperbolic space (negative curvature). The sectional curvature completely determines the Riemannian curvature completely, and consequently determines coarser notions like scalar curvature. Intuitively, spherical spaces are good at representing cyclical structure (e.g. the nodes in a cyclic graph). The work presented in this survey employs negative curvature spaces, which are good at representing hierarchical structure.

3.6 Geodesics and the exponential map

A *geodesic* is a curve that locally minimises distance. That is, a curve $\alpha : (a, b) \to \mathbb{R}$ such that

$$\nabla_{\gamma'(t)}(\gamma'(t)) = 0$$

Since $\gamma'(t)$ only defines a vector field along a curve, the above equation is computed by first smoothly extending $\gamma'(t)$ to vector fields on M, then restricting the resulting vector field back down to $\gamma(a, b)$. On the sphere, for example, geodesics are segments of great circles.

As we'll see in the next section, for any $x \in M$ and $v \in T_x M$, it's useful to be able to compute the geodesic starting at x and heading in the direction of v. The map $\exp_x : T_x M \to M$ that performs this computation is called the *exponential map*. Its inverse $\log_x : M \to T_x M$ is called the *logarithmic map*.

3.7 Riemannian gradient descent

Let $f : M \to \mathbb{R}$ be a smooth scalar function on M. By thinking of tangent vectors as differential operators, we can define the gradient $\nabla_x f$ at $x \in M$ as the vector $v \in T_x M$ such that

$$\langle v, w \rangle = w(f)$$

for all $w \in T_x M$. Such a vector v always exists because the binding of f on an element of $T_x M$ is a linear operation.

Riemannian gradient descent on f is then similar to Euclidean gradient descent, except we must take steps along geodesics instead of straight lines. We can solve this problem by passing through the exponential map. The iterations of Riemannian gradient descent are thus given by:

$$x_{t+1} = \exp_{x_t}(-\gamma \nabla_{x_t} f). \tag{6}$$

The value $\gamma > 0$ is the learning rate, and it may depend on t. Note that $\exp(d/dt)|_{t=0} f(t)$ is just the Taylor expansion of f about $t = 0$. The chain rule still applies on maps between Riemannian manifolds [4], and so we may still utilize backpropagation to compute gradients.

Alternatively, if the exponential map is too difficult to compute, one can use a linear approximation to the exponential map called a *retraction*. When M is immersed in an ambient Euclidean space $\mathbb{R}^n$ from which the Riemannian metric is inherited, the retraction becomes normal vector addition in $\mathbb{R}^n$ followed by a projection back onto M.

In some cases, the manifold of parameters M is immersed in an ambient Euclidean space $\mathbb{R}^n$, and the function f that one wants to optimize is defined on all of $\mathbb{R}^n$. There's no guarantee that the gradient of the function lies in the embedded tangent space (for example, consider a 2-sphere embedded in $\mathbb{R}^3$), and so in this case one must project onto the tangent space before applying the exponential map.

4 Hyperbolic Geometry

Most recent work that use Riemannian manifolds as spaces for representation learning use *hyperbolic* spaces, which are spaces with constant negative curvature. Often the latent features producing a data distribution have a hierarchal structure that is naturally represented as trees or complex networks [19].

4.1 Models of hyperbolic geometry

4.1.1 The Poincaré Ball

This is the model of hyperbolic geometry we'll use throughout this section. The Poincaré ball $\mathbb{D}_c^k$ with curvature $c \geq 0$ is the set $\{x \in \mathbb{R}^k : c|x|^2 < 1\}$ equipped with a Riemannian metric given by

$$\langle v, w \rangle_x^{\mathbb{D}_c^k} = \frac{2}{1 - c|x|^2} \langle v, w \rangle_x^{\mathbb{R}^k}, \tag{7}$$

where $v, w \in T_x\mathbb{D}_c^k$.[1] Note that Poincarè metric is the Euclidean metric multiplied by a scalar function. The scalar appears frequently in derived formulas, so we give it a name:

$$\lambda_x^c = \frac{2}{1 - c|x|^2} \tag{8}$$

Also, note that taking $c = 0$ recovers normal Euclidean space with the standard inner product.

4.1.2 Hyperboloid

The Minkowski bilinear form $\langle x, y \rangle_*$ for two vectors $x, y \in \mathbb{R}^{k+1}$ is defined as

$$\langle x, y \rangle_* = -x_1 y_1 + x_2 y_2 + x_3 y_3 \cdots + x_{k+1} y_{k+1}$$

This defines a (possibly negative) squared-norm $|x|_*^2 = \langle x, x \rangle_*$. The *hyperboloid model* (also known as the *Lorentz model*) is then defined as the set

$$\mathbb{H}^k = \{x \in \mathbb{R}^{k+1} : |x|_* = -K, x_1 > 0\}$$

with each tangent space $T_x\mathbb{H}^k$ induced as $\{y : \langle x, y \rangle_* = 0\}$. The inner product on each tangent space is also $\langle \cdot, \cdot \rangle_*$.

4.2 Mobius addition

To generalize many neural network architectures, we'd like to be able meaningfully combine points in hyperbolic space under closed operations. We can turn the Poincaré ball into a pseudo-vector space by introducing *Mobius operations*. The Mobius addition of two vectors in the Poincaré ball is given by

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

A natural interpretation of Mobius operations and their connection to parallel transport is given in [3]. Again, note that when $c = 0$, this equation reduces to standard vector addition on all of $\mathbb{R}^k$. Mobius addition is in general not commutative and not associative, but it does satisfy

$$0 \oplus x = x \oplus 0 = x,$$

and

$$-x \oplus x = x \oplus -x = 0$$
$$-x \oplus (x + y) = y.$$

[1] More rigourously, we really mean $\langle f_* v, f_* w \rangle_{f(x)}^{\mathbb{R}^k}$ instead of $\langle v, w \rangle_x^{\mathbb{R}^k}$, where f is the identity mapping from $\mathbb{D}_c^k$ to $\mathbb{R}^k$ and $f_* v$ is the push-forward of v from $T_x\mathbb{D}_c^k$ to $T_x\mathbb{R}^k$.

4.3 The exponential map

For the Poincarè ball $\mathbb{D}_c^k$, the exponential map can be computed explicitly using Mobius addition [3]:

$$\exp_x(v) = x \oplus_c (\tanh(\sqrt{c}\frac{\lambda_x^c|v|}{2})\frac{v}{\sqrt{c}|v|}).$$

The logarithmic map can also be computed explicitly:

$$\log_x(y) = \frac{2}{\sqrt{c}\lambda_x^c}\tanh^{-1}(\sqrt{c}|-x \oplus_c y|)\frac{-x \oplus_c y}{|-x \oplus_c y|}$$

Ganea et al. [3] give a derivation of these two equations that involves writing the geodesic equation in terms of Mobius addition and Mobius scalar multiplication (which we have not defined).

4.4 Affine transformation

A scalar-valued affine transformation $f_{a,b}(x) = \langle a, x - b\rangle$ in $\mathbb{R}^k$ can be written in the form

$$f_{a,b}(x) = \text{sign}(\langle a, x - b\rangle)|a|d(x, H_{a,b})$$

where $H_{a,b} = \{x \in \mathbb{R}^k : \langle a, x - b\rangle = 0\}$ is a hyperplane and $d(x, H_{a,b})$ is the distance from x to $H_{a,b}$. Ganea et al. [3] and Mathieu et al. [5] generalize this to scalar valued affine transormations $f_{a,b}^c : \mathbb{D}_c^k \to \mathbb{R}$ on the Poincaré ball

$$f_{a,b}^c = \text{sign}(\langle a, \log_x(b)\rangle_b)|a|d_p^c(x, H_{a,b}^c)$$

with the analogous hyperplane distance derived to be

$$\frac{1}{\sqrt{c}}\sinh^{-1}\left(\frac{2\sqrt{c}|\langle -1 \oplus_c x, a\rangle|}{(1 - c|-b \oplus a|^2)|a|}\right)$$

(Note that [5] claims this is a map $f_{a,b}^c : \mathbb{D}_c^k \to \mathbb{R}^{k'}$, but we don't believe that makes sense as written.)

5 Hyperbolic embeddings

5.1 Why hyperbolic space?

We now turn to a concrete application of using hyperbolic geometry to produce high quality, low-dimensional embeddings of hierarchical data. [6] considered embeddings into the Poincaré ball $\mathbb{D}_1^k$ with curvature -1. The metric in equation 8 gives rise to the calculus of variations problem

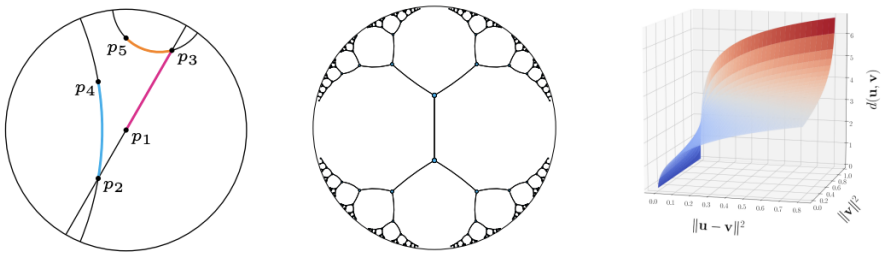
$$\int_u^v \frac{|\dot{z}(t)|}{1 - |\dot{z}|^2}dt. \tag{9}$$

Solving this problem in 2 dimensions with polar coordinates is straightforward and generalises to higher dimensions by noting that the Poincaré ball is rotationally symmetric and applying standard fixed-point theorems (see, *e.g.*, [9]). This leads to the following closed form expression for the distance between two points in the Poincaré ball:

$$d(u, v) = \cosh^{-1}\left(1 + 2\frac{\|u - v\|^2}{(1 - \|u\|^2)(1 - \|v\|^2)}\right) \tag{10}$$

In figure 1, we display three diagrams taken from [6], which provides a useful overview of the properties of the Poincaré ball model. We now discuss how these make the Poincaré ball an appropriate choice of embedding space.

The geodesics on the Poincaré ball correspond to straight lines or arcs that approach the boundary perpendicularly. The reason for this is the fast growth rate of the Poincaré distance, where the denominator terms in 10 means that, for $\|u - v\| > \epsilon > 0$ bounded below, $d(u, v)$ tends to infinity as u and v approach the boundary $\partial\mathbb{D}_1^k$ of the Poincaré ball. Shortest paths therefore consist of first moving from the starting point closer to the origin, where the distance metric is smaller, and then moving away from the origin, in order to reach the required end point. In particular, this means that points near the centre of the Poincaré ball are close to most points in the Poincaré ball and that two points near the edge of the Poincaré ball are about as far from each other as any other pair of points near the edge of the Poincaré ball (since a greater angular displacement requires the connecting geodesic to move through the centre of the Poincaré ball, and



(a) Geodesics of the Poincaré disk (b) Embedding of a tree in $\mathcal{B}^2$ (c) Growth of Poincaré distance

Figure 1: a) geodesics in $\mathbb{D}_1^2$; embedding of a tree with branching factor 2 into $\mathbb{D}_1^2$; and a plot of the Poincaré distance for constant $\|u\|^2$. Taken from [6].

this segment does not contribute much to the distance), unless the angular displacement of one of the pairs is close to zero [2].

The Poincaré ball is therefore a very suitable space for embedding hierarchical data, in that it acts as a structural prior that leads to data representing more general concepts being embedded at the centre of the Poincaré ball, and more specific concepts being embedded at the edges. In addition to this, the infinite distance to the boundary of the Poincaré ball means that an infinite number of points could be embedded into even a 2-dimensional space. This would be impossible in Euclidean space [13]. As a caveat, however, we caution against naively applying to Poincaré embeddings standard techniques used on Euclidean embeddings like linear interpolation and similarity inner products (since the Poincaré ball is not a vector space) [11] and clustering (since the metric used is not homogeneous) [12].

5.2 Optimisation

Given a set of symbols $\mathcal{S} = \{x_i\}_{i=1}^n$, [6] directly optimised over embeddings $\Theta = \{\theta_i\}_{i=1}^n$, where $\theta_i \in \mathbb{D}_1^k$. That is, they sought to find:

$$\Theta' = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{L}(\Theta) \quad \text{s.t.} \quad \forall \theta_i \in \Theta \,:\, \|\theta_i\| < 1. \tag{11}$$

To do this, [6] used a retraction mapping as described in section 3.7. Embedding $\mathbb{D}_1^k$ into $\mathbb{R}^k$, the Riemannian stochastic gradient descent (RSGD) updates involve differentiating $\mathcal{L}$ with respect to θ and pre-multiplying by the matrix of the inverse metric (and the learning rate) in order to obtain a contravariant vector, before subtracting this from the current value of θ and projecting back into the unit sphere:

$$\theta_{t+1} = \theta_t - \text{proj}\left(\theta_t - \eta_t \frac{\left(1 - \|\theta_t\|^2\right)^2}{4} \left.\frac{\partial \mathcal{L}}{\partial \theta}\right|_{\theta_t}\right), \tag{12}$$

where

$$\text{proj}(\theta) = \begin{cases} \theta / \|(\theta\| - \epsilon) & \text{if } \|\theta\| \geq 1, \\ \theta & \text{otherwise} \end{cases}.$$

5.3 Experiments

[6] carried out experiments on the transitive closure of WordNet (a standard dataset of word hypernyms) and on a collection of datasets of researcher names (graphs with researchers as nodes and with links between researchers who have co-authored a paper).

The first of these two experiments used a soft-ranking loss

$$\mathcal{L}(\Theta) = \sum_{(u,v) \in \mathcal{N}(u)} \log \frac{e^{-d(u,v)}}{\sum_{v' \in \mathcal{N}(u)} e^{-d(u,v')}}. \tag{13}$$

where

$$\mathcal{N}(u) = \{v | (u,v) \notin \mathcal{D}\} \cup \{u\}. \tag{14}$$

[2] Recall that $\text{arccosh} = \log\left(x + \sqrt{x^2 - 1}\right)$, and so $d(u,v) \approx \log\left(\|u - v\|\right) - \log\left(1 - \|u\|^2\right) - \log\left(1 - \|v\|^2\right)$ for u and v close to the ∂D_c^k.

Here $\mathcal{D}$ is the set of observed hypernymy relations.

The second of the two experiments used a logistic loss, with link prediction probability given by a Fermi-Dirac distribution

$$P((u,v)=1 \mid \Theta) = \frac{1}{e^{(d(u,v)-r)/t}+1}, \tag{15}$$

where $r, t > 0$ are hyperparameters.

5.4 Results

The resulting embeddings were tested on reconstruction (generating the embedded dataset from the embeddings) and link prediction (a number of links between entities were omitted at training time, and the embeddings were tested to see if they predicted these omitted links). Embeddings into the Poincaré ball were found to significantly outperform Euclidean embeddings in both cases. In particular, the former considerably outperformed the latter when the embedding dimension was small. We refer the reader to the actual paper [6] for a full description of the results.

5.5 Embeddings into the Lorentz model

A follow-up paper [7] considered instead the hyperboloid model described in section 4.1.2. The Poincaré disc model corresponds to a projection of the hyperboloid model onto a unit disc, and the two can be seen to be equivalent via the following transformations (y corresponding to points in the Poincaré disc, and (t, x) corresponding to points on the hyperboloid):

$$y_i = \frac{x_i}{1+t}, \tag{16}$$

$$(t, x_i) = \frac{(1+\sum_{i=1}^k y_i^2, 2y_i)}{1-\sum_{i=1}^k y_i^2}. \tag{17}$$

Whilst the solutions to 11 are the same (up to one of the above two transformations) in both models, optimisation via RSGD turns out to be easier in the Lorentz model. This is because of a non-singular metric and an easily computable, closed form for the exponential map, which means that we move directly along geodesics, rather than obtaining a first order approximation to a new point on the manifold and then reprojecting:

$$\exp_x(v) = \cosh(\|v\|_L)x + \sinh(\|v\|_L)\frac{v}{\|v\|_L}, \tag{18}$$

where inner products are taken with respect to the Lorentzian metric.

6 Hyperbolic Variational Autoencoders

Mathieu *et al.* [5] reformulate variational autoencoders so that the latent distributions are generalizations of the normal distribution on the Poincaré ball $\mathbb{D}_c^k$.

6.1 Latent space

The authors present two methods for generalizing the Euclidean normal distribution to Riemannian manifolds. The first uses the fact that the Euclidean normal distribution maximizes entropy for a given mean and variance. This distribution is called the *Riemannian normal*:

$$\mathcal{N}^R_{\mathbb{D}_c^k}(x \mid \mu, \sigma^2) = \frac{1}{Z^R}\exp\big(-\frac{d(\mu,x)}{2\sigma^2}\big)$$

Note the the exponential here is the normal exponential function from $\mathbb{R}$ to $\mathbb{R}$. The formula for the normalizing constant Z^R is also derived. A second way to generalize the Euclidean normal distribution pushes a normal distribution on a tangent space $T_\mu M$ down to the manifold:

$$x \sim \exp_\mu\big(\frac{v}{\lambda_\mu^c}\big)$$

with v drawn from a normal with mean 0 and variance σ^2. The also derive the closed form density:

$$\mathcal{N}^W_{\mathbb{D}_c^k} = \frac{1}{(\sqrt{2\pi}\sigma)^d} e^{d(\mu,x)^2/2\sigma^2}\left(\frac{\sqrt{c}d(\mu,x)}{\sinh(\sqrt{c}d(u,x))}\right)^{d-1}$$

This density is called the *wrapped normal*.

As in Euclidean VAE's, we can reparameterize the distribution and compute gradients via Monte-Carlo sampling. The authors reparameterize through polar coordinates on the tangent space:

$$x = \exp_\mu \left(\frac{r}{\lambda_\mu^c} \alpha \right),$$

where r and μ are random variables on $T_\mu M$. The direction α is drawn according to a uniform distribution on the hypersphere. In the wrapped normal, r is the Chi-squared distribution:

$$\rho^W(r) \propto 1_{\mathbb{R}_+}(r) e^{-\frac{r^2}{2\alpha^2}} r^{d-1}$$

The gradients $\nabla_\mu x, \nabla_\sigma x$ of the wrapped normal latent distribution are then easy to compute. Computing $\nabla_\sigma x$ of the Riemannian normal latent distribution is more complicated, and we refer the reader to the original paper.

6.2 Encoder Architecture

Like Euclidean variational autoencoders, the hyperbolic VAE's encoder produces a mean (Frèchet expectation) and log-variance from its input. Mathieu *et al.* use two fully connected layers connected layers with ReLU nonlinearity in between. No nonlinearity is applied to the log-variance $\log(\sigma^2) \in \mathbb{R}$ after the second layer. The mean $\mu \in P^c$ is pushed from the affine transformation output of the second layer into the Poincarè ball through the exponential map.

6.3 Decoder architecture

The decoder architecture uses the formula for affine transformations in the Poincaré ball presented in section 4. The samples from the Poincaré ball are fed to a series of affine transformations which are then concatenated and fed into a standard neural network. Again, the neural network has 2 layers with a ReLU nonlinearity in between.

6.4 Objective and training

The authors show that the variational lower bound (ELBO) can be extended to general Riemannian latent spaces:

$$\begin{aligned} \log(p(x)) &\geq \text{ELBO}_M(x; \theta, \phi) \\ &= \int_M \left(\frac{p_\theta(x \mid z) p(x)}{q_\phi(z \mid x)} \right) q_\phi(z \mid x) dM \end{aligned}$$

(See [4] for definition of Riemannian measures and integration on Riemannian manifolds.) With the densities for the distributions with respect to parameters θ and ϕ computed, the objective can be estimated by Monte-Carlo sampling.

The model is trained using the Adam optimiser [8], and the parameters optimised are the affine transformation offset parameters in $\mathbb{D}_c^k$ and the ordinary Euclidean neural network parameters. However, the offset parameters are treated as the projection of a real vector b' in the $T_0 \mathbb{D}_c^k$. That is,

$$b = \exp_0(b').$$

Since we have an explicit formula for the exponential map, no special treatment of the offset parameters is needed during optimisation.

6.5 Experiments

Mathieu et al. tested their autoencoder on two datasets. All tests were run on a Euclidean VAE and a selection of Poincaré VAEs with increasing curvatures.

The first was a synthetic dataset generated via a branching diffusion process: A finite sequence of points $y_i \in \mathbb{R}^k$ are normally distributed such that the mean of y_i is y_{i-1}. All variances are 1, and the mean of y_0 is 0. At each y_i, 5 observations $(x_{i,1}, \ldots, x_{i,5})$ are sampled from a normal with mean y_i and variance $\frac{1}{5}$. The models have access only to $\{x_{ij}\}$ and therefore do not have direct access to the hierarchical structure. The high curvature VAEs performed best in terms of test marginal likelihood, but all models (including the Euclidean VAE) captured the hierarchical structure. The authors suggest that Poincaré VAE latent representations are "qualitatively more pleasing".

The second dataset the group looked at was the MNIST dataset which contains 60,000 training and 10,000 test images of handwritten digits, each 28x28 pixels. Mathieu *et al.* empirically evaluated the latent representation learned by the

hyperbolic model and concluded that it "produced a clearer partitioning of the digits". The group also quantitatively measured each digit's latent representation by training classifiers that mapped the latent space to digit labels. The hyperbolic latent space performed better on 6 digits and had a higher average accuracy across all digits. The group never directly addressed whether the latent representations had exploited hierarchical structure. They note that the Riemannian normal tends to perform better than the wrapped normal on this task, despite their implementation being unstable.

References

[1] Guillaume Bouchard and Sameer Singh and Théo Trouillon. On Approximate Reasoning Capabilities of Low-Rank Vector Spaces. AAAI Spring Symposia. 2015.

[2] S. Bonnabel. Stochastic gradient descent on Riemannian Manifolds. *IEEE Trans. Automatic Control*, 58(9):2217-2221, 2013.

[3] Octavian-Eugen Ganea, Gary Bécigneul, and Thomos Hofmann. Hyperbolic Neural Networks arxiv preprint arxiv:1805.09112v2, 2018.

[4] Manfredo Perdigão do Carmo. Riemannian Geometry Birkhãuser, 1992.

[5] Hierarchical Representations with Poincaré Variational Auto-Encoders arxiv preprint arxiv:1901.06033v2, 2019.

[6] Maximilian Nickel and Douwe Kiela. Poincaré Embeddings for Learning Hierarchical Representations. Advances in Neural Information Processing Systems 30: 6338-6347, 2017.

[7] Maximilian Nickel and Douwe Kiela. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In International Conference on Machine Learning, 2018.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of 3rd International Conference on Learning Representations, 2015.

[9] Jeffrey Lee. Manifolds and differential geometry. Graduate Studies in Mathematics Volume 107. American Mathematical Society.

[10] Boris Muzellec and Marco Cuturi. Generalizing point embeddings using the wasserstein space of elliptical distributions. arXiv preprint arXiv:1805.07594, 2018.

[11] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013a.

[12] Jiang Guo, Wanxiang Che, and Haifeng Wang, and Ting Liu. Revisiting Embedding Features for Simple Semi-supervised Learning. Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 110–120, 2014.

[13] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Reducing the Rank of Relational Factorization Models by Including Observable Patterns. Advances in Neural Information Processing Systems 27: 1179-1187. 2014

[14] Diederik Kingma and Max Welling. Auto-Encoding Variational Bayes. Proceedings of the 2nd International Conference on Learning Representations (ICLR). 2014.

[15] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. 2017 NIPS Workshop on Learning Disentangled Representations.

[16] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. Empirical Methods in Natural Language Processing, pages 1532-1543. 2014.

[17] Ehsaneddin Asgari and Mohammad R. K. Mofrad. Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics. PLoS One. 2015.

[18] Zellig S. Harris. Distributional Structure. Word, 10(2-3): 146-162, 1954.

[19] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. Physical Review E, 67(2):026112, 2003.