A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

Yoshihiro Nagano ¹  Shoichiro Yamaguchi ²  Yasuhiro Fujita ²  Masanori Koyama ²

Abstract

Hyperbolic space is a geometry that is known to be well-suited for representation learning of data with an underlying hierarchical structure. In this paper, we present a novel hyperbolic distribution called pseudo-hyperbolic Gaussian, a Gaussian-like distribution on hyperbolic space whose density can be evaluated analytically and differentiated with respect to the parameters. Our distribution enables the gradient-based learning of the probabilistic models on hyperbolic space that could never have been considered before. Also, we can sample from this hyperbolic probability distribution without resorting to auxiliary means like rejection sampling. As applications of our distribution, we develop a hyperbolic-analog of variational autoencoder and a method of probabilistic word embedding on hyperbolic space. We demonstrate the efficacy of our distribution on various datasets including MNIST, Atari 2600 Breakout, and WordNet.

1. Introduction

Recently, hyperbolic geometry is drawing attention as a powerful geometry to assist deep networks in capturing fundamental structural properties of data such as a hierarchy. Hyperbolic attention network (Gülçehre et al., 2019) improved the generalization performance of neural networks on various tasks including machine translation by imposing the hyperbolic geometry on several parts of neural networks. Poincaré embeddings (Nickel & Kiela, 2017) succeeded in learning a parsimonious representation of symbolic data by embedding the dataset into Poincaré balls. In the task of data embedding, the choice of the target space determines the properties of the dataset that can be learned from the embedding. For the dataset with a hierarchical structure, in particular, the number of relevant features can grow exponentially with the depth of the hierarchy. Euclidean space is often inadequate for capturing the structural information (Figure 1). If the choice of the target space of the embedding is limited to Euclidean space, one might have to prepare extremely high dimensional space as the target space to guarantee small distortion. However, the same embedding can be done remarkably well if the destination is the hyperbolic space (Sarkar, 2012; Sala et al., 2018).

Figure 1: The visual results of Hyperbolic VAE applied to an artificial dataset generated by applying random perturbations to a binary tree. The visualization is being done on the Poincaré ball. The red points are the embeddings of the original tree, and the blue points are the embeddings of noisy observations generated from the tree. The pink × represents the origin of the hyperbolic space. The VAE was trained without the prior knowledge of the tree structure. Please see 6.1 for experimental details.

1Department of Complexity Science and Engineering, The University of Tokyo, Japan. ²Preferred Networks, Inc., Japan. Correspondence to: Yoshihiro Nagano <nagano@mns.k.u-tokyo.ac.jp>.

Copyright 2019 by the authors.
Now, the next natural question is; “how can we extend these works to probabilistic inference problems on hyperbolic space?” When we know in advance that there is a hierarchical structure in the dataset, a prior distribution on hyperbolic space might serve as a good informative prior. We might also want to make Bayesian inference on a dataset with hierarchical structure by training a variational autoencoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) with latent variables defined on hyperbolic space. We might also want to conduct probabilistic word embedding into hyperbolic space while taking into account the uncertainty that arises from the underlying hierarchical relationship among words. Finally, it would be best if we can compare different probabilistic word embedding methods to hyperbolic space using our distribution, which is also the length of the geodesic that connects two points.

The endeavors we mentioned in the previous paragraph all require probability distributions on hyperbolic space that admit a parametrization of the density function that can be computed analytically and differentiated with respect to the parameter. Also, we want to be able to sample from the distribution efficiently; that is, we do not want to resort to auxiliary methods like rejection sampling.

In this study, we present a novel hyperbolic distribution called pseudo-hyperbolic Gaussian, a Gaussian-like distribution on hyperbolic space that resolves all these problems. We construct this distribution by defining Gaussian distribution on the tangent space at the origin of the hyperbolic space and projecting the distribution onto hyperbolic space after transporting the tangent space to a desired location in the space. This operation can be formalized by a combination of the parallel transport and the exponential map for the Lorentz model of hyperbolic space.

We can use our pseudo-hyperbolic Gaussian distribution to construct a probabilistic model on hyperbolic space that can be trained with gradient-based learning. For example, our distribution can be used as a prior of a VAE (Figure 1, Figure 6). It is also possible to extend the existing probabilistic embedding method to hyperbolic space using our distribution, such as probabilistic word embedding. We will demonstrate the utility of our method through the experiments of probabilistic hyperbolic models on benchmark datasets including MNIST, Atari 2600 Breakout, and WordNet.

2. Background

2.1. Hyperbolic Geometry

Hyperbolic geometry is a non-Euclidean geometry with a constant negative Gaussian curvature, and it can be visualized as the forward sheet of the two-sheeted hyperboloid. There are four common equivalent models used for the hyperbolic geometry: the Klein model, Poincaré disk model, and Lorentz (hyperboloid/Minkowski) model, and Poincaré half-plane model. Many applications of hyperbolic space to machine learning to date have adopted the Poincaré disk model as the subject of study (Nickel & Kiela, 2017; Ganea et al., 2018a;b; Sala et al., 2018). In this study, however, we will use the Lorentz model that, as claimed in Nickel & Kiela (2018), comes with a simpler closed form of the geodesics and does not suffer from the numerical instabilities in approximating the distance. We will also exploit the fact that both exponential map and parallel transport have a clean closed form in the Lorentz model.

Lorentz model $\mathbb{H}^n$ (Figure 2(a)) can be represented as a set of points $z \in \mathbb{R}^{n+1}$ with $z_0 > 0$ such that its Lorentzian product (negative Minkowski bilinear form) $\langle z, z' \rangle_L = -z_0 z'_0 + \sum_{i=1}^{n} z_i z'_i$, with itself is $-1$. That is, $\mathbb{H}^n = \{z \in \mathbb{R}^{n+1}: \langle z, z \rangle_L = -1, \ z_0 > 0 \}$. (1)

Lorentzian inner product also functions as the metric tensor on hyperbolic space. We will refer to the one-hot vector $\mu_0 = [1, 0, 0, \ldots 0] \in \mathbb{H}^n \subset \mathbb{R}^{n+1}$ as the origin of the hyperbolic space. Also, the distance between two points $z, z'$ on $\mathbb{H}^n$ is given by $d_L(z, z') = \arccosh(-\langle z, z' \rangle_L)$, which is also the length of the geodesic that connects $z$ and $z'$.

2.2. Parallel Transport and Exponential Map

The rough explanation of our strategy for the construction of pseudo-hyperbolic Gaussian $\mathcal{G}(\mu, \Sigma)$ with $\mu \in \mathbb{H}^n$ and a positive definite matrix $\Sigma$ is as follows. We (1) sample a vector from $\mathcal{N}(0, \Sigma)$, (2) transport the vector from $\mu_0$ to $\mu$ along the geodesic, and (3) project the vector onto the surface. To formalize this sequence of operations, we need to define the tangent space on hyperbolic space as well as the way to transport the tangent space and the way to project a vector in the tangent space to the surface. The transportation of the tangent vector requires parallel transport, and the projection of the tangent vector to the surface requires the definition of exponential map.

Tangent space of hyperbolic space

Let us use $T_{\mu} \mathbb{H}^n$ to denote the tangent space of $\mathbb{H}^n$ at $\mu$ (Figure 2(a)). Representing $T_{\mu} \mathbb{H}^n$ as a set of vectors in the same ambient space $\mathbb{R}^{n+1}$ into which $\mathbb{H}^n$ is embedded, $T_{\mu} \mathbb{H}^n$ can be characterized as the set of points satisfying the orthogonality relation with respect to the Lorentzian product:

$$T_{\mu} \mathbb{H}^n := \{u: \langle u, \mu \rangle_L = 0 \}. \quad (2)$$
According to the basic theory of differential geometry, every \( \mu \in T_\mu \mathbb{H}^n \) determines a unique maximal geodesic \( \gamma_\mu : [0, 1] \to \mathbb{H}^n \) with \( \gamma_\mu(0) = \mu \) and \( \gamma_\mu(1) = u \). Exponential map \( \exp \mu : T_\mu \mathbb{H}^n \to \mathbb{H}^n \) is a map defined by \( \exp \mu(u) = \gamma_\mu(1) \), and we can use this map to project a vector \( v \in T_{\mu'} \mathbb{H}^n \) onto \( \mathbb{H}^n \) in a way that the distance from \( \mu \) to destination of the map coincides with \( \|v\|_\mathcal{L} \), the metric norm of \( v \). For hyperbolic space, this map (Figure 2(c)) is given by

\[
\exp \mu(u) = \cosh (\|u\|_\mathcal{L}) \mu + \sinh (\|u\|_\mathcal{L}) \frac{u}{\|u\|_\mathcal{L}}.
\]

As we can confirm with straightforward computation, this exponential map is norm preserving in the sense that

\[
d_\mathcal{L}(\mu, \exp \mu(u)) = \arccosh \left( -\langle \mu, \exp \mu(u) \rangle_\mathcal{L} \right) = \|u\|_\mathcal{L}.
\]

Now, in order to evaluate the density of a point on hyperbolic space, we need to be able to map the point back to the tangent space, on which the distribution is initially defined. We, therefore, need to be able to compute the inverse of the exponential map, which is also called logarithm map, as well.

Solving eq. (5) for \( u \), we can obtain the inverse exponential map as

\[
u = \exp^{-1}_\mu(z) = \frac{\arccosh(\alpha)}{\sqrt{\alpha^2 - 1}} (z - \alpha \mu), \quad (6)
\]

where \( \alpha = -\langle \mu, z \rangle_\mathcal{L} \). See Appendix A.1 for further details.

3. Pseudo-Hyperbolic Gaussian

3.1. Sampling

Finally, we are ready to formally explain our method of generating our pseudo-hyperbolic Gaussian \( \mathcal{G}(\mu, \Sigma) \) with \( \mu \in \mathbb{H}^n \) and a positive definite \( \Sigma \).

In the language of the differential geometry, our strategy can be re-described as follows:

1. Sample a vector \( \tilde{v} \) from the Gaussian distribution \( \mathcal{N}(0, \Sigma) \) defined over \( \mathbb{R}^n \).
Algorithm 1 Sampling on hyperbolic space

| Input: parameter $\mu \in \mathbb{H}^n$, $\Sigma$ |
| Output: $z \in \mathbb{H}^n$ |
| Require: $\mu_0 = (1, 0, \ldots, 0)^T \in \mathbb{H}^n$ |

Sample $\tilde{v} \sim \mathcal{N}(0, \Sigma) \in \mathbb{R}^n$

Let $v = [0, \tilde{v}] \in T_{\mu_0} \mathbb{H}^n$

Move $v$ to $u = PT_{\mu_0} \mathbb{H}^n(v) \in T_{\mu} \mathbb{H}^n$ by eq. (3)

Map $u$ to $z = \exp_\mu(u) \in \mathbb{H}^n$ by eq. (5)

2. Interpret $\tilde{v}$ as an element of $T_{\mu_0} \mathbb{H}^n \subset \mathbb{R}^{n+1}$ by rewriting $\tilde{v}$ as $v = [0, \tilde{v}]$.

3. Parallel transport the vector $v$ to $u \in T_{\mu} \mathbb{H}^n \subset \mathbb{R}^{n+1}$ along the geodesic from $\mu_0$ to $\mu$.

4. Map $u$ to $\mathbb{H}^n$ by $\exp_\mu$.

Algorithm 1 is an algorithmic description of the sampling procedure.

The most prominent advantage of this construction is that we can compute the density of the probability distribution.

3.2. Probability Density Function

Note that both $PT_{\mu_0} \mu$ and $\exp_\mu$ are differentiable functions that can be evaluated analytically. Thus, by the construction of $\mathcal{G}(\mu, \Sigma)$, we can compute the probability density of $\mathcal{G}(\mu, \Sigma)$ at $z \in \mathbb{H}^n$ using a composition of differentiable functions, $PT_{\mu_0} \mu$ and $\exp_\mu$. Let $\text{proj}_\mu := \exp_\mu \circ PT_{\mu_0} \mu$ (Figure 3).

In general, if $X$ is a random variable endowed with the probability density function $p(x)$, the log likelihood of $Y = f(X)$ at $y$ can be expressed as

$$\log p(y) = \log p(x) - \log \det \left( \frac{\partial f}{\partial x} \right)$$

where $f$ is a invertible and continuous map. Thus, all we need in order to evaluate the probability density of $\mathcal{G}(\mu, \Sigma)$ at $z = \text{proj}_\mu(v)$ is the way to evaluate

$$\log p(z) = \log p(v) - \log \det \left( \frac{\partial \text{proj}_\mu(v)}{\partial v} \right).$$

Algorithm 2 is an algorithmic description for the computation of the pdf.

For the implementation of algorithm 1 and algorithm 2, we would need to be able to evaluate not only $\exp_\mu(u)$, $PT_{\mu_0} \mu(v)$ and their inverses, but also need to evaluate the determinant. We provide an analytic solution to each of them below.

Algorithm 2 Calculate log-pdf

| Input: sample $z \in \mathbb{H}^n$, parameter $\mu \in \mathbb{H}^n$, $\Sigma$ |
| Output: $\log p(z)$ |
| Require: $\mu_0 = (1, 0, \ldots, 0)^T \in \mathbb{H}^n$ |

Map $z$ to $u = \exp_\mu^{-1}(z) \in T_{\mu_0} \mathbb{H}^n$ by eq. (6)

Move $u$ to $v = PT_{\mu_0} \mathbb{H}^n(u) \in T_{\mu_0} \mathbb{H}^n$ by eq. (4)

Calculate $\log p(z)$ by eq. (7)

Log-determinant

We compute the log-determinant of the Jacobian of $\text{proj}_\mu := \exp_\mu \circ PT_{\mu_0} \mu$. This is required in the evaluation of (7).

Appealing to the chain-rule and the rule of the determinant, we can decompose the expression into two components:

$$\det \left( \frac{\partial \text{proj}_\mu(v)}{\partial v} \right) = \det \left( \frac{\partial \exp_\mu(u)}{\partial u} \right) \cdot \det \left( \frac{\partial PT_{\mu_0} \mu(v)}{\partial v} \right).$$

For the first term, we get

$$\frac{\partial \exp_\mu(u)}{\partial u} = \frac{\sinh(r)}{r} \begin{bmatrix} r \mu + \left( \frac{r}{\tanh(r)} - 1 \right) d^\top J \end{bmatrix},$$

where $r = \|u\|_\mathbb{L}$ and $d = u/\|u\|_\mathbb{L}$. Now, using the identity $\det(c(I + uv^\top)) = c^n(1 + u^\top v)$, we obtain

$$\det \left( \frac{\partial \exp_\mu(u)}{\partial u} \right) = \left( \frac{\sinh(r)}{r} \right)^n \cosh(r).$$

See Appendix A.3 for further details.

Next, the second term can be computed as

$$\frac{\partial PT_{\mu_0} \mu(v)}{\partial v} = I + \frac{1}{\alpha + 1}(\mu_0 + \mu)(\mu - \alpha \mu_0)^\top J,$$

$$\det \left( \frac{\partial PT_{\mu_0} \mu(v)}{\partial v} \right) = 1 + \frac{(\mu_0 + \mu - \alpha \mu_0) \mathbb{L}}{\alpha + 1} = \alpha,$$

where $\alpha = -\langle \mu_0, \mu \rangle \mathbb{L}$. See Appendix A.4 for further details.

Putting these computations together, we can obtain the desired determinant in a simple and clean form:

$$\det \left( \frac{\partial \text{proj}_\mu(v)}{\partial v} \right) = \left( \frac{\sinh(r)}{r} \right)^n \cosh(r) \alpha.$$
which the latent variables are defined on hyperbolic space. As an application of pseudo-hyperbolic Gaussian VAE (Kingma & Welling, 2014; Rezende et al., 2014) in (a) 4.1. Hyperbolic Variational Autoencoder that is defined for each model is trained together with the encoder model \( q \). The decoder aims to train a decoder model \( p \) from \( p \).

Hyperbolic VAE is a simple modification of the classic VAE in which \( p_\theta = \mathcal{G}(\mu, \Sigma) \) and \( q_\phi = \mathcal{G}(\mu, \Sigma) \). The model of \( \mu \) and \( \Sigma \) is often referred to as encoder. This parametric formulation of \( q_\phi \) is called reparametrization trick, and it enables the evaluation of the gradient of the objective function with respect to the network parameters. To compare our method against, we used \( \beta \)-VAE (Higgins et al., 2017), a variant of VAE that applies a scalar weight \( \beta \) to the KL term in the objective function.

In Hyperbolic VAE, we assure that output \( \mu \) of the encoder is in \( \mathbb{H}^n \) by applying \( \exp_{\mu_0} \) to the final layer of the encoder. That is, if \( h \) is the output, we can simply use

\[
\mu = \exp_{\mu_0}(h) = \left( \cosh(\|h\|_2), \ \sinh(\|h\|_2) \frac{h}{\|h\|_2} \right)^T.
\]

As stated in the previous sections, our distribution \( \mathcal{G}(\mu, \Sigma) \) allows us to evaluate the ELBO exactly and to take the gradient of the objective function. In a way, our distribution of the variational posterior is an hyperbolic-analog of the reparametrization trick.

4.2. Word Embedding

We can use our psudo-hyperbolic Gaussian \( \mathcal{G} \) for probabilistic word embedding. The work of Vilnis & McCallum (2015) attempted to extract the linguistic and contextual properties of words in a dictionary by embedding every word and every context to a Gaussian distribution defined on Euclidean space. We may extend their work by changing the destination of the map to the family of \( \mathcal{G} \). Let us write \( a \sim b \) to convey that there is a link between words \( a \) and \( b \), and let us use \( q_s \) to designate the distribution to be assigned to the word \( s \). The objective function used in Vilnis & McCallum (2015) is given by

\[
L(\theta) = E_{s \sim t} \left[ \max \left( 0, m + E(s, t) - E(s, t') \right) \right],
\]

where \( E(s, t) \) represents the measure of similarity between \( s \) and \( t \) evaluated with \( D_{KL}(q_s || q_t) \). In the original work, \( q_s \) and \( q_t \) were chosen to be a Gaussian distribution. We can incorporate hyperbolic geometry into this idea by choosing \( q_s = \mathcal{G}(\mu(s), \Sigma(s)) \).

5. Related Work

As mentioned in the introduction, most studies to date that use hyperbolic space consider only deterministic mappings (Nickel & Kiela, 2017; 2018; Ganea et al., 2018a;b; Gülçehre et al., 2019).

Very recently, Ovinnikov (2019) proposed an application of Gaussian distribution on hyperbolic space. However, the
A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

formulation of their distribution cannot be directly differenti-  
ated nor evaluated because of the presence of error function  
in their expression of pdf. For this reason, they resort to  
Wasserstein Maximum Mean Discrepancy (Gretton et al.,  
2012) to train their encoder network. Our distribution $G$  
has broader application than the distribution of Ovinnikov  
(2019) because it allows the user to compute its likelihood  
and its gradient without approximation. One advantage of  
our distribution $G(\mu, \Sigma)$ is its representation power. Our  
distribution $G(\mu, \Sigma)$ can be defined for any $\mu$ in $\mathbb{R}^n$  
and any positive definite matrix $\Sigma \in \mathbb{R}^{n \times n}$. Meanwhile,  
the hyperbolic Gaussian studied in Ovinnikov (2019) can only  
express Gaussian with variance matrix of the form $\sigma I$.

For word embedding, several deterministic methods have  
been proposed to date, including the celebrated Word2Vec  
(Mikolov et al., 2013). The aforementioned Nickel & Kiela  
(2017) uses deterministic hyperbolic embedding to exploit  
the hierarchical relationships among words. The probabilis-  
tic word embedding was first proposed by Vilnis & McCall-  
um (2015). As stated in the method section, their method  
maps each word to a Gaussian distribution on Euclidean  
space. Their work suggests the importance of investigating  
the uncertainty of word embedding. In the field of represent-  
ation learning of word vectors, our work is the first in using  
hyperbolic probability distribution for word embedding.

On the other hand, the idea to use a noninformative, non-  
Gaussian prior in VAE is not new. For example, Davidson  
et al. (2018) proposes the use of von Mises-Fisher prior,  
and Rolfe (2017); Jang et al. (2017) use discrete distributions  
as their prior. With the method of Normalizing flow  
(Rezende & Mohamed, 2015), one can construct even more  
complex priors as well (Kingma et al., 2016). The appropri-  
ate choice of the prior shall depend on the type of dataset.

As we will show in the experiment section, our distribution  
is well suited to the dataset with underlying tree structures.  
Another choice of the VAE prior that specializes in such  
such dataset has been proposed by Vikram et al. (2018). For the  
sampling, they use time-marginalized coalescent, a model  
that samples a random tree structure by a stochastic process.  
Theoretically, their method can be used in combination with  
our approach by replacing their Gaussian random walk with  
a hyperbolic random walk.

### 6. Experiments

#### 6.1. Synthetic Binary Tree

We trained Hyperbolic VAE for an artificial dataset con-  
structed from a binary tree of depth $d = 8$. To construct  
the artificial dataset, we first obtained a binary representa-  
tion for each node in the tree so that the Hamming distance  
between any pair of nodes is the same as the distance on  
the graph representation of the tree (Figure 1(a)). Let us  
call the set of binaries obtained this way by $A_0$. We then  
generated a set of binaries, $A$, by randomly flipping each  
coordinate value of $A_0$ with probability $\epsilon = 0.1$. The binary  
set $A$ was then embedded into $\mathbb{R}^d$ by mapping $a_1a_2...a_d$ to  
$[a_1, a_2, ..., a_d]$. We used an Multi Layer Perceptron (MLP)  
of depth 3 and 100 hidden variables at each layer for both  
encoder and decoder of the VAE. For activation function we  
used $\tanh$.

Table 1 summarizes the quantitative comparison of Normal  
VAE against our Hyperbolic VAE. For each pair of points  
in the tree, we computed their Hamming distance as well  
as their distance in the latent space of VAE. That is, we  
used Hyperbolic distance for Hyperbolic VAE, and used  
Euclidean distance for Normal VAE. We used the strength  
of correlation between the Hamming distances and the dis-  
tances in the latent space as a measure of performance.  
Hyberbolic VAE was performing better both on the original  
tree and on the artificial dataset generated from the tree.  
Normal VAE performed the best with $\beta = 2.0$, and  
collapsed with $\beta = 3.0$. The difference between Normal  
VAE and Hyperbolic VAE can be observed with much more  
clearly using the 2-dimensional visualization of the gener-  
atized dataset on Poincaré Ball (See Figure 1 and Appendix  
C.1). The red points are the embeddings of $A_0$, and the  
blue points are the embeddings of all other points in $A$. The pink × mark designates the origin of hyperbolic space. For  
the visualization, we used the canonical diffeomorphism  
between the Lorenz model and the Poincaré ball model.

#### 6.2. MNIST

We applied Hyperbolic VAE to a binarized version of  
MNIST. We used an MLP of depth 3 and 500 hidden units  
at each layer for both the encoder and the decoder of the  
VAE. Table 2 shows the quantitative results of the experi-  
ments. Log-likelihood was approximated with an empirical  
integration of the Bayesian predictor with respect to the  
latent variables (Burda et al., 2016). Our method outper-  
formed Normal VAE with small latent dimension. Figure  
4(a) are the samples of the Hyperbolic VAE that was  
trained with 5-dimensional latent variables, and Figure 4(b)  
are the Poincaré Ball representations of the interpolations

| Model    | Correlation | Correlation w/ noise |
|----------|-------------|----------------------|
| Normal   | $\beta = 0.1$ | 0.48 | 0.45 |
|          | $\beta = 1.0$ | 0.66 | 0.51 |
|          | $\beta = 2.0$ | 0.71 | 0.55 |
|          | $\beta = 3.0$ | 0.47 | 0.01 |
|          | $\beta = 4.0$ | 0.22 | 0.01 |
| Hyperbolic |               | 0.77 | 0.60 |

Table 1: Results of tree embedding experiments for the  
Hyperbolic VAE and Normal VAEs trained with different  
weight constants for the KL term.
A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

|       | Normal VAE | Hyperbolic VAE |
|-------|------------|----------------|
| $n$   | ELBO  | LL  | ELBO  | LL  |
| 2     | -148.58 | -143.38 | -145.34 | **-139.94** |
| 5     | -110.99 | -106.76 | -110.36 | **-105.32** |
| 10    | -90.40  | -85.66  | -92.00  | -86.19 |
| 20    | -82.98  | -76.90  | -85.24  | -77.47 |

Table 2: Quantitative comparison of Hyperbolic VAE against Normal VAE on the MNIST dataset in terms of ELBO and log-likelihood (LL) for several values of latent space dimension $n$. LL was computed using 500 samples of latent variables.

The Figure 6 is a visualization of our results. The top three rows are the samples from Normal VAE, and the bottom three rows are the samples from Hyperbolic VAE. Each row consists of samples generated from latent variables of the form $a\tilde{v}/\|\tilde{v}\|_2$ with positive scalar $a$ in range $[1, 10]$. Samples in each row are listed in increasing order of $a$. For Normal VAE, we used $\mathcal{N}(0, I)$ as the prior. For Hyperbolic VAE, we used $\mathcal{G}(\mu_0, I)$ as the prior. We can see that the number of blocks decreases gradually and consistently in each row for Hyperbolic VAE. Please see Appendix C.2 for more details and more visualizations.

Figure 7 shows the estimated proportions of remaining blocks for Normal and Hyperbolic VAEs with different norm of $\tilde{v}$. For Normal VAE, samples generated from $\tilde{v}$ with its norm as large as $\|\tilde{v}\|_2 = 200$ contained considerable amount of blocks. On the other hand, the number of blocks contained in a sample generated by Hyperbolic VAE decreased more consistently with the norm of $\|\tilde{v}\|_2$. This fact suggests that the cumulative reward up to a given state can be approximated well by the norm of Hyperbolic VAE’s latent representation. To validate this, we computed latent representation for each state in the test set and measured its correlation with the cumulative reward. The correlation was 0.8540 for the Hyperbolic VAE. For the Normal VAE, the correlation was 0.712. We emphasize that no information regarding the reward was used during the training of both Normal and Hyperbolic VAEs.

6.4. Word Embeddings

Lastly, we applied pseudo-hyperbolic Gaussian to word embedding problem. We trained probabilistic word embedding models with WordNet nouns dataset (Miller, 1998) and eval-
A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

Figure 6: Samples from Normal and Hyperbolic VAEs trained on Atari 2600 Breakout screens. Each row was generated by sweeping the norm of $\tilde{v}$ from 1.0 to 10.0 in a log-scale.

Figure 7: Estimated proportions of remaining blocks for Normal and Hyperbolic VAEs trained on Atari 2600 Breakout screens as they vary with the norm of latent variables sampled from a prior.

Table 3: Experimental results of the reconstruction performance on the transitive closure of the WordNet noun hierarchy for several latent space dimension $n$.

| $n$ | Euclid MAP | Hyperbolic MAP | Nickel & Kiela (2017) MAP |
|-----|------------|----------------|--------------------------|
|     | Rank       | Rank           |                          |
| 5   | 0.359      | 0.544          | 0.823                    |
| 10  | 0.773      | 0.817          | 0.851                    |
| 20  | 0.897      | 0.905          | 0.855                    |
| 50  | 0.953      | 0.969          | 0.86                     |
| 100 | 0.955      | 0.977          | 0.857                    |

7. Conclusion

In this paper, we proposed a novel parametrization for the density of Gaussian on hyperbolic space that can both be differentiated and evaluated analytically. Our experimental results on hyperbolic word embedding and hyperbolic VAE suggest that there is much more room left for the application of hyperbolic space. Our parametrization enables gradient-based training of probabilistic models defined on hyperbolic space and opens the door to the investigation of complex models on hyperbolic space that could not have been explored before.

Acknowledgements

We would like to thank Tomohiro Hayase, Kenta Oono, and Masaki Watanabe for helpful discussions. We also thank Takeru Miyato and Sosuke Kobayashi for insightful reviews.

We followed the procedure of Poincaré embedding (Nickel & Kiela, 2017) and initialized all embeddings in the neighborhood of the origin. In particular, we initialized each weight in the first linear part of the embedding by $\mathcal{N}(0, 0.01)$. We treated the first 50 epochs as a burn-in phase and reduced the learning rate by a factor of 4 after the burn-in phase.

In Table 3, ‘Euclid’ refers to the word embedding with Gaussian distribution on Euclidean space (Vilnis & McCallum, 2015), and ‘Hyperbolic’ refers to our proposed method based on pseudo-hyperbolic Gaussian. Our hyperbolic model performed better than Vilnis’ Euclidean counterpart when the latent space is low dimensional. We used diagonal variance for both models above. Appendix C.3 shows the results with unit variance. The performance difference with small latent dimension was much more remarkable when we use unit variance.
on the paper. This paper is based on results obtained from Nagano’s internship at Preferred Networks, Inc.

References

Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In Proceedings of the 4th International Conference on Learning Representations, 2016.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. 34th Conference on Uncertainty in Artificial Intelligence, 2018.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic entailment cones for learning hierarchical embeddings. In Proceedings of the 35th International Conference on Machine Learning, pp. 1632–1641, 2018a.

Ganea, O., Bécigneul, G., and Hofmann, T. In Advances in Neural Information Processing Systems 31, pp. 5350–5360, 2018b.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. The Journal of Machine Learning Research, 13:723–773, March 2012.

Gülçehre, Ç., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., and de Freitas, N. Hyperbolic attention networks. In Proceedings of the 7th International Conference on Learning Representations, 2019.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. β-vae: Learning basic visual concepts with a constrained variational framework. In Proceedings of the 5th International Conference on Learning Representations, 2017.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning, volume 37, pp. 448–456, 2015.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In Proceedings of the 5th International Conference on Learning Representations, 2017.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In Proceedings of the 2nd International Conference on Learning Representations, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In Advances in Neural Information Processing Systems 29, pp. 4743–4751, 2016.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems 26, pp. 3111–3119, 2013.

Miller, G. WordNet: An electronic lexical database. MIT press, 1998.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. Nature, 518(7540): 529, 2015.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In Advances in Neural Information Processing Systems 30, pp. 6338–6347, 2017.

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, pp. 3776–3785, 2018.

Ovinnikov, I. Poincaré wasserstein autoencoder. CoRR, abs/1901.01427, 2019.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Proceedings of the 4th International Conference on Learning Representations, 2016.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In Proceedings of the 32nd International Conference on Machine Learning, pp. 1530–1538, 2015.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the 31st International Conference on Machine Learning, volume 32, pp. 1278–1286, 2014.

Rolfe, J. T. Discrete variational autoencoders. In Proceedings of the 5th International Conference on Learning Representations, 2017.

Sala, F., De Sa, C., Gu, A., and Re, C. Representation tradeoffs for hyperbolic embeddings. In Proceedings of the 35th International Conference on Machine Learning, volume 80, pp. 4460–4469, 2018.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In Graph Drawing, pp. 355–366, 2012.
A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

Vikram, S., Hoffman, M. D., and Johnson, M. J. The loracs prior for vaes: Letting the trees speak for the data. *CoRR*, abs/1810.06891, 2018.

Vilnis, L. and McCallum, A. Word representations via gaussian embedding. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.
Appendix: A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

A. Derivations

A.1. Inverse Exponential Map

As we mentioned in the main text, the exponential map from $T_{\mu} \mathbb{H}^n$ to $\mathbb{H}^n$ is given by

$$z = exp_\mu(u) = \cosh(\|u\|_\mathcal{L})\mu + \sinh(\|u\|_\mathcal{L})\frac{u}{\|u\|_\mathcal{L}}.$$  

Solving this equation for $u$, we obtain

$$u = \frac{\|u\|_\mathcal{L}}{\sinh(\|u\|_\mathcal{L})}(z - \cosh(\|u\|_\mathcal{L})\mu).$$

We still need to obtain the evaluable expression for $\|u\|_\mathcal{L}$. Using the characterization of the tangent space (main text, (2)), we see that

$$\langle \mu, u \rangle_\mathcal{L} = \frac{\|u\|_\mathcal{L}}{\sinh(\|u\|_\mathcal{L})}(\langle \mu, z \rangle_\mathcal{L} - \cosh(\|u\|_\mathcal{L})\langle \mu, \mu \rangle_\mathcal{L}) = 0,$$

$$\cosh(\|u\|_\mathcal{L}) = -\langle \mu, z \rangle_\mathcal{L},$$

$$\|u\|_\mathcal{L} = \text{arccosh}(-\langle \mu, z \rangle_\mathcal{L}).$$

Now, defining $\alpha = -\langle \mu, z \rangle_\mathcal{L}$, we can obtain the inverse exponential function as

$$u = \exp^{-1}_\mu(z) = \frac{\text{arccosh}(\alpha)}{\sqrt{\alpha^2 - 1}}(z - \alpha \mu).$$

A.2. Inverse Parallel Transport

The parallel transportation on the Lorentz model along the geodesic from $\nu$ to $\mu$ is given by

$$\text{PT}_{\nu\rightarrow\mu}(v) = v - \frac{\langle \mu, v \rangle_\mathcal{L}}{d_\mathcal{L}(\nu, \mu)^2}(\exp^{-1}_\nu(\mu) + \exp^{-1}_\mu(\nu))$$

$$= v + \frac{\langle \mu - \alpha \nu, v \rangle_\mathcal{L}}{\alpha + 1}(\nu + \mu),$$  \hspace{1cm} (10)

where $\alpha = -\langle \nu, \mu \rangle_\mathcal{L}$. Next, likewise, for the exponential map, we need to be able to compute the inverse of the parallel transform. Solving (10) for $v$, we get

$$v = u - \frac{\langle \mu - \alpha \nu, v \rangle_\mathcal{L}}{\alpha + 1}(\nu + \mu).$$

Now, observing that

$$\langle \nu - \alpha \mu, u \rangle_\mathcal{L} = \langle \nu, v \rangle_\mathcal{L} + \frac{\langle \mu - \alpha \nu, v \rangle_\mathcal{L}}{\alpha + 1}(\langle \nu, v \rangle_\mathcal{L} + \langle \mu, v \rangle_\mathcal{L})$$

$$= -\langle \mu, v \rangle_\mathcal{L} = -\langle \mu - \alpha \nu, v \rangle_\mathcal{L},$$
we can write the inverse parallel transport as

\[ v = \text{PT}_{\nu \to \mu}^{-1}(u) = u + \frac{(\nu - \alpha \mu, u)}{\alpha + 1} (\nu + \mu). \]

The inverse of parallel transport from \( \nu \) to \( \mu \) coincides with the parallel transport from \( \mu \) to \( \nu \).

### A.3. Determinant of exponential map

As for the first term of (8) in the main text, we can write

\[
\frac{\partial \exp_{\mu}(u)}{\partial u} = \frac{\partial \cosh(|u|_L)}{\partial u} \mu + \frac{\partial \sinh(|u|_L)}{\partial u} \frac{u}{|u|_L} + \sinh(|u|_L) \frac{\partial u}{|u|_L} = \frac{\sinh(|u|_L)}{|u|_L} uu^\top J + \frac{\cosh(|u|_L)}{|u|_L} uu^\top J + \frac{\sinh(|u|_L)}{|u|_L} \left( I - \frac{1}{|u|_L^2} uu^\top J \right),
\]

where we wrote

\[ J = \begin{pmatrix}
-1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 1
\end{pmatrix}. \]

Now, using the change of variables \( r = |u|_L, \ d = u/|u|_L \), we get

\[
\frac{\partial \exp_{\mu}(u)}{\partial u} = \sin(r) d^\top J + \cosh(r) dd^\top J + \frac{\sinh(r)}{r} (I - dd^\top J) = \frac{\sinh(r)}{r} \left[ I + \left\{ r \mu + \left( \frac{r}{\tanh(r)} - 1 \right) d \right\} d^\top J \right].
\]

Using the identity \( \det(I + uv^\top) = e^n(1 + u^\top v) \), we obtain

\[
\det \left( \frac{\partial \exp_{\mu}(u)}{\partial u} \right) = \left( \frac{\sinh(r)}{r} \right)^{n+1} \left[ 1 + \langle u, \mu \rangle_L + \left( \frac{r}{\tanh(r)} - 1 \right) \right] = \left( \frac{\sinh(r)}{r} \right)^n \cosh(r).
\]

### A.4. Determinant of parallel transport

Next, the second term (8) in the main text can be computed as

\[
\frac{\partial \text{PT}_{\mu_0 \to \mu}(v)}{\partial v} = \frac{\partial v}{\partial v} + \frac{\mu_0 + \mu}{\alpha + 1} \frac{\partial v}{\partial v} \langle \mu - \alpha \mu_0, v \rangle_L
\]

\[ = I + \frac{1}{\alpha + 1} (\mu_0 + \mu)(\mu - \alpha \mu_0)^\top J,
\]

where \( \alpha = -\langle \mu_0, \mu \rangle_L \). Using the identity \( \det(I + uv^\top) = 1 + u^\top v \), we get

\[
\det \left( \frac{\partial \text{PT}_{\mu_0 \to \mu}(v)}{\partial v} \right) = 1 + \frac{\mu_0 + \mu - \alpha \mu_0}{\alpha + 1} = \alpha.
\]
B. Visual Examples of Pseudo-Hyperbolic Gaussian $\mathcal{G}(\mu, \Sigma)$

Figure 8 shows examples of pseudo-hyperbolic Gaussian $\mathcal{G}(\mu, \Sigma)$ with various $\mu$ and $\Sigma$. We plotted the log-density of these distributions by heatmaps. We designate the $\mu$ by the $\times$ mark. The right side of these figures expresses their log-density on the Poincaré ball model, and the left side expresses the same one on the corresponding tangent space.

Figure 8: Visual examples of pseudo-hyperbolic Gaussian on $\mathbb{H}^2$. Log-density is illustrated on $\mathbb{B}^2$ by translating each point from $\mathbb{H}^2$ for clarity. We designate the origin of hyperbolic space by the $\times$ mark.
C. Additional Numerical Evaluations

C.1. Synthetic Binary Tree

We qualitatively compared the learned latent space of Normal and Hyperbolic VAEs. Figure 9 shows the embedding vectors of the synthetic binary tree dataset on the two-dimensional latent space. We evaluated the latent space of Normal VAE with $\beta = 0.1, 1.0, 2.0, \text{ and } 3.0$, and Hyperbolic VAE. Note that the hierarchical relations in the original tree were not used during the training phase. Red points are the embeddings of the noiseless observations. As we mentioned in the main text, we evaluated the correlation coefficient between the Hamming distance on the data space and the hyperbolic (Euclidean for Normal VAEs) distance on the latent space. Consistently with this metric, the latent space of the Hyperbolic VAE captured the hierarchical structure inherent in the dataset well. In the comparison between Normal VAEs, the latent space captured the hierarchical structure according to increase the $\beta$. However, the posterior distribution of the Normal VAE with $\beta = 3.0$ collapsed and lost the structure. Also, the blue points are the embeddings of noisy observation, and pink $\times$ represents the origin of the latent space. In latent space of Normal VAEs, there was bias in which embeddings of noisy observations were biased to the center side.

(a) A tree representation of the training dataset  
(b) Normal ($\beta = 0.1$)  
(c) Normal ($\beta = 1.0$)  
(d) Normal ($\beta = 2.0$)  
(e) Normal ($\beta = 3.0$)  
(f) Hyperbolic

Figure 9: The visual results of Normal and Hyperbolic VAEs applied to an artificial dataset generated by applying a random perturbation to a binary tree. The visualization is being done in the Poincaré ball. Red points are the embeddings of the original tree, and the blue points are the embeddings of all other points in the dataset. Pink $\times$ represents the origin of hyperbolic space. Note that the hierarchical relations in the original tree was not used during the training phase.
C.2. Atari 2600 Breakout

To evaluate the performance of Hyperbolic VAE for hierarchically organized dataset according to time development, we applied our Hyperbolic VAE to a set of trajectories that were explored by an agent with a trained policy during multiple episodes of Breakout in Atari 2600. We used a pretrained Deep Q-Network to collect trajectories, and Figure 10 shows examples of observed screens.

![Figure 10: Examples of observed screens in Atari 2600 Breakout.](image)

We showed three trajectories of samples from the prior distribution with the scaled norm for both models in the main text. We also visualize more samples in Figure 11 and 12. For both models, we generated samples with $\|\tilde{v}\|_2 = 0, 1, 2, 3, 5, \text{ and } 10$.

Normal VAE tended to generate oversaturated images when the norm $\|\tilde{v}\|$ was small. Although the model generated several images which include a small number of blocks as the norm increases, it also generated images with a constant amount of blocks even $\|\tilde{v}\| = 10$. On the other hand, the number of blocks contained in the generated image of Hyperbolic VAE gradually decreased according to the norm.
A Differentiable Gaussian-like Distribution on Hyperbolic Space for Gradient-Based Learning

Figure 11: Images generated by Normal VAE with constant norm $\|\tilde{v}\|^2 = a$.

(a) $\|\tilde{v}\|^2 = 0$  
(b) $\|\tilde{v}\|^2 = 1$  
(c) $\|\tilde{v}\|^2 = 2$

(d) $\|\tilde{v}\|^2 = 3$  
(e) $\|\tilde{v}\|^2 = 5$  
(f) $\|\tilde{v}\|^2 = 10$

Figure 12: Images generated by Hyperbolic VAE with constant norm $\|\tilde{v}\|^2 = a$.

(a) $\|\tilde{v}\|^2 = 0$  
(b) $\|\tilde{v}\|^2 = 1$  
(c) $\|\tilde{v}\|^2 = 2$

(d) $\|\tilde{v}\|^2 = 3$  
(e) $\|\tilde{v}\|^2 = 5$  
(f) $\|\tilde{v}\|^2 = 10$
C.3. Word Embeddings

We showed the experimental results of probabilistic word embedding models with diagonal variance in the main text. In this section, we show the results with unit variance (Table 4). When the dimensions of the latent variable are small, the performance of the model on hyperbolic space did not deteriorate much by changing the variance from diagonal to unit. However, the same change dramatically worsened the performance of the model on Euclidean space.

| n  | MAP Rank | MAP Rank |
|----|----------|----------|
| 5  | 0.163 272.8 | 0.535 \textbf{29.1} |
| 10 | 0.512 49.9  | 0.778 5.3  |
| 20 | 0.792 11.3  | 0.854 2.7  |
| 50 | 0.842 21.4  | 0.905 1.8  |
| 100| 0.854 19.3  | 0.874 2.5  |

Table 4: Experimental results of the word embedding models with unit variance on the WordNet noun dataset.

D. Network Architecture

Table 5 shows the network architecture that we used in Breakout experiments. We evaluated Normal and Hyperbolic VAEs with a DCGAN-based architecture (Radford et al., 2016) with the kernel size of the convolution and deconvolution layers as 3. We used leaky ReLU nonlinearities for the encoder and ReLU nonlinearities for the decoder. We set the latent space dimension as 20. We gradually increased $\beta$ from 0.1 to 4.0 linearly during the first 30 epochs. To ensure the initial embedding vector close to the origin, we initialized $\gamma$ for the batch normalization layer (Ioffe & Szegedy, 2015) of the encoder as 0.1. We modeled the probability distribution of the data space $p(x|z)$ as Gaussian, so the decoder output a vector twice as large as the original image.

| Encoder | Size |
|---------|------|
| Input   | $80 \times 80 \times 1$ |
| Convolution | $80 \times 80 \times 16$ |
| BatchNormalization | |
| Convolution | $40 \times 40 \times 32$ |
| BatchNormalization | |
| Convolution | $40 \times 40 \times 32$ |
| BatchNormalization | |
| Convolution | $20 \times 20 \times 64$ |
| BatchNormalization | |
| Convolution | $20 \times 20 \times 64$ |
| BatchNormalization | |
| Convolution | $10 \times 10 \times 64$ |
| Linear | $2n$ |

| Decoder | Size |
|---------|------|
| Linear | $10 \times 10 \times 64$ |
| BatchNormalization | |
| Deconvolution | $20 \times 20 \times 32$ |
| BatchNormalization | |
| Convolution | $20 \times 20 \times 32$ |
| BatchNormalization | |
| Deconvolution | $40 \times 40 \times 16$ |
| BatchNormalization | |
| Convolution | $40 \times 40 \times 16$ |
| Deconvolution | $80 \times 80 \times 2$ |
| Convolution | $80 \times 80 \times 2$ |

Table 5: Network architecture for Atari 2600 Breakout dataset.