Communications on Applied Mathematics and Computation
https://doi.org/10.1007/s42967-023-00302-9

ORIGINAL PAPER

Convergence of Hyperbolic Neural Networks Under Riemannian Stochastic Gradient Descent

Wes Whiting[1] · Bao Wang[2] · Jack Xin[1]

Received: 31 October 2022 / Revised: 28 July 2023 / Accepted: 31 July 2023

Abstract

We prove, under mild conditions, the convergence of a Riemannian gradient descent method for a hyperbolic neural network regression model, both in batch gradient descent and stochastic gradient descent. We also discuss a Riemannian version of the Adam algorithm. We show numerical simulations of these algorithms on various benchmarks.

Keywords Hyperbolic neural network · Riemannian gradient descent · Riemannian Adam (RAdam) · Training convergence

Mathematics Subject Classification 53Z50 · 68T07

1 Introduction

In machine learning, we routinely encode data as vectors in the Euclidean space. However, some data have a latent geometric structure which is not suited to Euclidean embedding, and instead can be represented more faithfully in other geometries. In particular, hyperbolic embeddings have recently gained momentum, especially for data with latent hierarchical or tree-like structure, like synonym or type hierarchies [3]. Just as ordinary neural networks can be used to interpolate functions of data in the Euclidean space, a hyperbolic neural network can be used to interpolate functions on data embedded in the hyperbolic space.

Hyperbolic feed-forward neural networks, or HFFNNs, were introduced in [4], and generalizations to Riemannian manifolds for common training methods like the gradient descent were introduced in [1, 2]. It is known that HFFNNs can outperform conventional FFNNs on some problems such as textual entailment and noisy-prefix recognition [4], and a universal approximation theorem for HFFNNs was proved in [6]. However, little is known about their convergence under the gradient descent.

✉ Wes Whiting
wwhiting@uci.edu

[1] Department of Mathematics, University of California, Irvine, CA, USA

[2] Department of Mathematics, Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA

Published online: 05 October 2023

In this paper, we show that under natural conditions, the training of an HFFNN under Riemannian gradient descent will indeed converge. We also present numerical results of training a network under Riemannian gradient descent versus Riemannian Adam (RAdam), showing that it provides the accelerated convergence in practice. Previous papers have examined only networks of limited depth or which are fully linear, while our result applies to nonlinear HFFNNs of arbitrary depth, in exchange for requiring an adaptive step size to be computed at each iteration of training.

2 Background

2.1 Hyperbolic Neural Networks

Here we summarize some facts about gyrovector spaces and hyperbolic neural networks which we will need for our main result. A more complete introduction of hyperbolic neural networks can be found in [4], and of gyrovector spaces in [12].

The hyperbolic space, denoted as $\mathbb{D}_c^n$, can be modeled in several ways. We use the Poincaré ball model, which is the Riemannian manifold $(\mathbb{D}_c^n, g^{\mathbb{D}})$ where

$$\begin{aligned} \mathbb{D}_c^n &= \{x \in \mathbb{R}^n \colon c||x||^2 < 1\}, \\ g_x^{\mathbb{D}} &= (\lambda_x^c)^2 \mathbf{I}_n, \text{ where } \lambda_x^c = 2/(1 - c||x||^2). \end{aligned}$$

This model has the benefit of being conformal to the Euclidean space, as well as allowing us to use the closed-form expressions from [4] for certain operations. Other models have advantages of their own. In particular, the Lorentz half-sheet model can avoid the numerical instability that is encountered in the Poincaré model [11].

In $\mathbb{R}^n$ considered as a Riemannian manifold, vector addition can be written as $x + y = \exp_x^c(P_{0\to x} \log_0^c(y))$, where $P_{0\to x}$ is the parallel transport from 0 to x—this is simply the "tip-to-tail" method of vector addition often taught to students, restated in the language of manifolds. Analogously, we can define the operation of *Möbius addition* in $\mathbb{D}_c^n$ by $x \oplus_c y = \exp_x^c(P_{0\to} \log_0^c(y))$, where now the parallel transport is taken along the Levi-Civita connection of $\mathbb{D}_c^n$. This can be computed explicitly by

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

Note in particular that $x \oplus_c y$ is a rational function in the coordinates of x and y, and by Cauchy-Schwarz the denominator is nonvanishing in $\mathbb{D}_c^n$, so gyrovector addition is smooth, a fact which we will use later.

Analogously we can define Möbius scalar multiplication by

$$r \otimes_c x = \exp_0^c(r \log_0^c(x)),$$

and $\mathbb{D}_c^n$ equipped with the operations $\oplus_c, \otimes_c$ is called a gyrovector space. More generally, given any function $f\colon \mathbb{R}^n \to \mathbb{R}^m$, we can define its Möbius version $f^{\otimes_c}\colon \mathbb{D}_c^n \to \mathbb{D}_c^m$ by

$$f^{\otimes_c}(x) = \exp_0^c(f(\log_0^c(x))).$$

In particular, we will discuss the Möbius version of an activation function σ and of a matrix W acting as a linear map, which we denote by $\sigma^{\otimes_c}$ and $W^{\otimes_c}$, respectively. Since we are working in the Poincaré ball model, we may use the coordinates of the model to unambiguously identify linear maps with matrices, so henceforth we will not mention the distinction.

With these definitions in hand, a hyperbolic neural network layer with the weight matrix W, the bias b, and the activation function σ is given by $x \mapsto \sigma^{\otimes_c}(W^{\otimes_c} x \oplus_c b)$, analogous to a Euclidean network layer $x \mapsto \sigma(Wx + b)$, and a network f of k hidden layers of n_i neurons is given by

$$\begin{aligned} f(x) &= f_k \circ f_{k-1} \circ \cdots \circ f_2 \circ f_1(x), \\ f_i(x) &= \sigma_i^{\otimes_c}(W_i^{\otimes_c} x \oplus_c b_i), \quad 1 \leqslant i \leqslant k, \end{aligned} \tag{1}$$

such a network has parameters $\mathbf{W} = (W_1, b_1, \cdots, W_k, b_k)$ with $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$, $b_i \in \mathbb{D}_c^{n_i}$. Euclidean and hyperbolic layers can be used in sequence by applying $\log_0^c$ or $\exp_0^c$ to move from $\mathbb{D}_c^n$ to $\mathbb{R}^n$ or vice versa, respectively. Note that unlike the Euclidean case, an HFFNN is always nonlinear, and thus a non-identity activation function is not strictly necessary to fit nonlinear data and may not even improve the performance.

2.2 Riemannian Training

Stochastic gradient descent, or SGD, is perhaps the most straightforward way to train a model. Although the use of the Poincaré ball model allows us to embed $\mathbb{D}_c^n$ into $\mathbb{R}^n$ and thus we could compute a gradient in the Euclidean sense, this is not geometrically meaningful, and can behave badly near the boundary. In fact, an ordinary gradient update could even send the bias outside $\mathbb{D}_c^n$, which is clearly undesirable. Bonnabel [2] introduced the algorithm of Riemannian stochastic gradient descent, or RSGD, to address this problem on a Riemannian manifold. Analogous to gradient descent in $\mathbb{R}^n$ which proceeds by $w_{t+1} = w_t - \gamma_t \nabla \mathcal{L}(\mathbf{X}_t, w_t) = \exp_{w_t}^c(-\gamma_t \nabla \mathcal{L}(\mathbf{X}_t, w_t))$ where $\mathbf{X}_t$ is the stochastically chosen sample from the dataset $\mathbf{X}$ and $\nabla \mathcal{L}$ is the gradient in the calculus sense, the RSGD proceeds by

$$w_{t+1} = \exp_{w_t}^c(-\gamma_t \operatorname{grad} \mathcal{L}(\mathbf{X}_t, w_t)), \tag{2}$$

where $\operatorname{grad} \mathcal{L}$ is the gradient as a vector field on the manifold. Naturally, we require the samples to be chosen in such a way that $\mathcal{L}(\mathbf{X}, w) = \mathbb{E}[\mathcal{L}(\mathbf{X}_t, w)]$.

A common modification of gradient descent is to implement momentum, which can accelerate the convergence and help escape saddle points. The *Adam* algorithm [5] uses momentum as a second-moment estimate. Bécigneul and Ganea [1] generalized these algorithms to Riemannian manifolds as well, interpreting Adam's adaptivity in each coordinate as the adaptivity in the components of a product of manifolds. This is well-suited to our approach, since training a hyperbolic neural network with m layers of sizes $n_1, \cdots, n_m$ is precisely an optimization problem on a product manifold of the form $\mathbb{R}^{n_1 \times n_0} \times \mathbb{D}_c^{n_1} \times \cdots \times \mathbb{R}^{n_m \times n_{m-1}} \times \mathbb{D}_c^{n_m}$. We restate these algorithms below, alongside the Euclidean versions for comparison.

Algorithm 1 SGD with momentum

```
Require: $\theta_0, \beta \in [0, 1), \{\eta_t\}_{t=1}^{\infty}$
  Set $m_0 = 0, t = 0$
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow \beta m_{t-1} + (1 - \beta) g_t$
    $\theta_t \leftarrow \theta_{t-1} - \eta_t m_t$
  end while
  return $\theta_t$
```

Algorithm 2 RSGD with momentum

```
Require: $\theta_0, \beta \in [0, 1), \{\eta_t\}_{t=1}^{T}$
  Set $m_0 = 0, t = 0$
  while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \mathrm{grad}_\theta f_t(\theta_{t-1})$
    $m_t \leftarrow P_{\theta_{t-2}}^{\theta_{t-1}}(\beta m_{t-1}) + (1 - \beta) g_t$
    $\theta_t \leftarrow \exp_{\theta_{t-1}}^{c}(-\eta_t m_t)$
  end while
  return $\theta_t$
```

As in Euclidean training, it is often helpful to examine whether a Riemannian gradient is Lipschitz. We use a generalization of the notion of a Lipschitz vector field, introduced in [9].

Definition 1 A function $f\colon (M, g) \to \mathbb{R}$ is said to have C-Lipschitz gradient if for any two points $a, b \in M$,

$$|\,\mathrm{grad} f(b) - P_{a \to b}\, \mathrm{grad} f(a)|_g \leqslant C\, \mathrm{dist}_M(a, b).$$

As one would hope, a Lipschitz Riemannian gradient guarantees that a sufficiently small step size will cause the loss to decrease, as shown in the following proposition.

Algorithm 3 Adam

```
Require: $\theta_0, \{\beta_1, \beta_2\}, \{\eta_t\}_{t=1}^{\infty}$
  $m_0, v_0, t \leftarrow 0$
  while $\theta_t$ not converged do
      $t \leftarrow t + 1$
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
      $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
      $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
      $\theta_t \leftarrow \theta_{t-1} - \eta \frac{m_t}{\sqrt{v_t} + \epsilon}$
  end while
  return $\theta_t$
```

Algorithm 4 Riemannian Adam

```
Require: $\theta_0, \{\beta_1, \beta_2\}, \{\eta_t\}_{t=1}^{\infty}$
  $m_0, v_0, t \leftarrow 0$
  while $\theta_t$ not converged do
      $t \leftarrow t + 1$
      for $1 \leqslant i \leqslant n$ do
          $g_t^i \leftarrow \mathrm{grad}_{\theta^i} f_t^i(\theta_{t-1})$
          $m_t^i \leftarrow P_{\theta_{t-2}}^{\theta_{t-1}} \beta_1 m_{t-1}^i + (1 - \beta_1) g_t^i$
          $v_t^i \leftarrow \beta_2 v_{t-1}^i + (1 - \beta_2) ||g_t^i||_{\rho^i}^2$
          $\theta_t^i \leftarrow \exp_{\theta_{t-1}^i}^c (-\alpha \frac{m_t^i}{\sqrt{v_t^i} + \epsilon})$
      end for
  end while
  return $\theta_t$
```

Proposition 1 *Let $f \colon M \to \mathbb{R}$ be a differentiable function on a manifold M, and denote the Levi-Civita connection by $P_{a \to b}$. If f has C-Lipschitz gradient in the above sense, then $f(b) - f(a) \leqslant \langle \mathrm{grad} f(a), \log_a(b) \rangle_g + \frac{C}{2} \mathrm{dist}(a, b)^2$.*

Proof Let $\gamma \colon [0, 1] \to M$ be the geodesic curve from a to b. (Note that in general, such paths are not unique, but on the manifolds under consideration in this paper, they are.)

$$\begin{aligned}
f(b)-f(a) &= \int_\gamma \operatorname{grad} f \text{ by the fundamental theorem of line integrals}\\
&= \int_0^1 \langle \operatorname{grad} f(\gamma(t)), \gamma'(t)\rangle_g \,\mathrm{d}t\\
&= \int_0^1 \langle P_{a\to\gamma(t)} \operatorname{grad} f(a), \gamma'(t)\rangle_g \,\mathrm{d}t\\
&\quad + \int_0^1 \langle \operatorname{grad} f(\gamma(t)) - P_{a\to\gamma(t)} \operatorname{grad} f(a), \gamma'(t)\rangle_g \,\mathrm{d}t\\
&= \int_0^1 \langle \operatorname{grad} f(a), \gamma'(0)\rangle_g \,\mathrm{d}t + \int_0^1 \langle \operatorname{grad} f(\gamma(t)) - P_{a\to\gamma(t)} \operatorname{grad} f(a), \gamma'(t)\rangle_g \,\mathrm{d}t
\end{aligned}$$

since γ a geodesic and $P_{a\to\gamma(t)}$ metric-preserving

$$\begin{aligned}
&= \langle \operatorname{grad} f(a), \gamma'(0)\rangle_g \cdot \operatorname{length}(\gamma)\\
&\quad + \int_0^1 \langle \operatorname{grad} f(\gamma(t)) - P_{a\to\gamma(t)} \operatorname{grad} f(a), \gamma'(t)\rangle_g \,\mathrm{d}t\\
&= \langle \operatorname{grad} f(a), \log_a(b)\rangle_g + \int_0^1 \langle \operatorname{grad} f(\gamma(t)) - P_{a\to\gamma(t)} \operatorname{grad} f(a), \gamma'(t)\rangle_g \,\mathrm{d}t\\
&\leqslant \langle \operatorname{grad} f(a), \log_a(b)\rangle_g + \int_0^1 |\operatorname{grad} f(\gamma(t)) - P_{a\to\gamma(t)} \operatorname{grad} f(a)|_g |\gamma'(t)|_g \,\mathrm{d}t\\
&\leqslant \langle \operatorname{grad} f(a), \log_a(b)\rangle_g + \int_0^1 C \operatorname{dist}(a, \gamma(t)) |\gamma'(t)|_g \,\mathrm{d}t\\
&= \langle \operatorname{grad} f(a), \log_a(b)\rangle_g + \frac{C}{2} \operatorname{dist}(a,b)^2.
\end{aligned}$$

In particular, in RGD since $w_{t+1} = \exp_{w_t}(-\gamma_t \operatorname{grad} f(w_t))$, the distance is $\operatorname{dist}(w_{t+1}, w_t) = \gamma_t |\operatorname{grad} f(w_t)|_g$, so we obtain

$$\begin{aligned}
f(w_{t+1}) - f(w_t) &\leqslant \langle \operatorname{grad} f(w_t), -\gamma_t \operatorname{grad} f(w_t)\rangle_g + \frac{C}{2} |\gamma_t \operatorname{grad} f(w_t)|_g^2\\
&= \left(-\gamma_t + \frac{C}{2}\gamma_t^2\right) |\operatorname{grad} f(w_t)|_g^2.
\end{aligned} \tag{3}$$

From this, we recover the familiar condition that if $\gamma_t < \frac{2}{C}$, then $f(w_{t+1}) < f(w_t)$.

3 Convergence of HFFNNs Under RSGD

An HFFNN layer has two kinds of parameters: the weight matrix W, which is an ordinary $m \times n$ matrix, and the bias b which is a gyrovector in the Poincaré ball. As such, the gradient descent must be performed in a hybrid fashion. The weights are Euclidean and thus can be updated in the usual fashion, but since the biases are in the hyperbolic space rather than $\mathbb{R}^n$, we use the RSGD. This hybrid gradient descent scheme is equivalent to the RSGD on the product manifold $\prod_{i=1}^n \mathbb{R}^{n_{i+1} \times n_i} \times \mathbb{D}_c^{n_{i+1}}$, and so we will consider this unified interpretation.

We will show that an HFFNN converges under the RGD or the RSGD using any C^2 activation functions.[1] We rely on the following theorem from [2].

Theorem (Bonnabel 2013) *Assume that*

$$\sum \gamma_t^2 < \infty, \sum \gamma_t = \infty.$$

Suppose there exists a compact set K such that $w_t \in K$ *for all* $t \geqslant 0$, *and that the gradient of the loss* $\operatorname{grad} \mathcal{L}$ *is bounded on K. Then under the RSGD, the loss* $\mathcal{L}(w_t)$ *converges almost surely, and* $\operatorname{grad} \mathcal{L}(w_t)$ *converges almost surely to zero.*

It is worth noting that, because this theorem is so general, it gives only a weak notion of the convergence, and in particular does not guarantee that the weights themselves converge to any sort of critical point.

Theorem 1 *Consider an HFFNN of arbitrarily many layers as described above, with loss given by*

$$\mathcal{L}(\mathbf{X}, \mathbf{W}) = \frac{1}{k} \sum_{x \in \mathbf{X}} (f(x; \mathbf{W}) - y_i)^2 + \lambda |\mathbf{W}|_2^2 \tag{4}$$

for the dataset $\mathbf{X}$ *and parameters* $\mathbf{W}$, *with any* C^2 *activation functions. Let* C_p *denote the local Lipschitz constant of* $\mathcal{L}$ *on the ball of radius* 1 *around p established in Lemma* 1. *If we select the step size by*

$$\gamma_t \leqslant \min\left(\frac{2}{C_{\mathbf{W}_{t-1}}}, \frac{1}{|\operatorname{grad}_{\mathbf{W}} \mathcal{L}(\mathbf{X}, \mathbf{W}_{t-1})|_g}, \frac{1}{t} \right),$$

then under non-stochastic Riemannian gradient descent, the loss $\mathcal{L}(\mathbf{W}_t)$ *converges, and its gradient converges to* 0.

Proof The loss has locally Lipschitz gradient by Lemma 1, so by (dynamically) selecting a sufficiently small step size—small enough to not leave a compact neighborhood of the current parameters (say, a ball of radius 1, which $\gamma_t \leqslant \frac{1}{|\operatorname{grad} \mathcal{L}(\mathbf{X}_t, \mathbf{W}_{t-1})|_g}$ is sufficient to guarantee), and smaller than $2/C_t$ where C_t is the local Lipschitz constant on this ball—we can guarantee that the loss decreases at each step by (3). Furthermore, we restrict these step sizes γ_t, so that $\gamma_t \leqslant 1/t$, which enforces the condition that $\sum \gamma_t^2 < \infty$. If $\sum \gamma_t < \infty$ as well, then the weights converge absolutely, which gives us an even better result than expected.

Otherwise $\sum \gamma_t = \infty$ and we are in the case examined by [2]. In this case, the sequence of loss values is decreasing and in particular is bounded. Thanks to its L^2 regularization term, the loss $\mathcal{L}$ is coercive, so the sequence of weights is also bounded, and in particular they lie in a compact set, and since $\mathcal{L}$ is smooth, it is certainly bounded on this set. By the theorem of Bonnabel from [2], the loss converges and the gradient of the loss converges to zero.

[1] In particular, the identity function is C^2 and can work perfectly well as an activation function, as we noted earlier (1). Many activations such as logistic or tanh are also C^2, but unfortunately ReLU is not.

Theorem 2 *Consider an HFFNN as in Theorem 1, with the stochastic objective function $\mathcal{L}_\theta$ satisfying $\mathbb{E}_\theta[\mathcal{L}_\theta] = \mathcal{L}$. Let $C^\theta_{\mathbf{W}_{t-1}}$ be the local Lipschitz constant of $\mathcal{L}_\theta$ in the unit ball around $\mathbf{W}_{t-1}$. If the step size γ_t is selected to satisfy*

$$\gamma_t \leqslant \min\left(\frac{2}{C^\theta_{\mathbf{W}_{t-1}}}, \frac{1}{|\operatorname{grad}_{\mathbf{W}}\mathcal{L}_\theta(\mathbf{X}, \mathbf{W}_{t-1})|_g}, \frac{1}{t}\right),$$

then under RSGD, the loss $\mathcal{L}(\mathbf{W}_t)$ converges almost surely, and the gradient of the loss converges almost surely to zero.

Proof Again by (3) above, we see that for any fixed θ, we must have $\mathcal{L}_\theta(\mathbf{X}, \mathbf{W}_{t+1}) \leqslant \mathcal{L}_\theta(\mathbf{X}, \mathbf{W}_t)$, and thus

$$\mathcal{L}(\mathbf{X}, \mathbf{W}_{t+1}) = \mathbb{E}_\theta[\mathcal{L}_\theta(\mathbf{X}, \mathbf{W}_{t+1})] \leqslant \mathbb{E}_\theta[\mathcal{L}_\theta(\mathbf{X}, \mathbf{W}_t)] = \mathcal{L}(\mathbf{X}, \mathbf{W}_t).$$

This shows that $\mathcal{L}(\mathbf{X}, \mathbf{W}_t)$ is decreasing, and thus the same argument as in Theorem 1 above goes through.

Lemma 1 *If the activation functions σ_i are C^2, then the loss function $\mathcal{L}$ (4) has a locally Lipschitz gradient.*

Proof Denote $M = \prod_{i=1}^n \mathbb{R}^{n_{i+1}\times n_i} \times \mathbb{D}_c^{n_{i+1}}$, the Riemannian manifold containing the parameters of the network. Let $K \subset \prod_{i=1}^n \mathbb{R}^{n_{i+1}\times n_i} \times \mathbb{D}_c^{n_{i+1}}$ be a compact set, and without loss of generality, a rectangle (i.e., $K = \prod_{i=1}^n R_i \times H_i$, for some compact sets $R_i \subset \mathbb{R}^{n_{i+1}\times n_i}$ and $H_i \subset \mathbb{D}_c^{n_{i+1}}$). We will work in the standard coordinates for the Poincaré ball. As before, we use ∇ to denote the gradient in the calculus sense, while grad denotes the gradient in the manifold sense, and similarly we will use $||\cdot||$ to denote the Euclidean norm when identifying a tangent space of $\mathbb{D}_c^n$ with $\mathbb{R}^n$.

Since we are performing gradient descent on the product manifold M, it will be helpful to consider the gradients separately with respect to the weight matrices $W^i \in \mathbb{R}^{n_{i+1}\times n_i}$ and the biases $b^i \in \mathbb{D}_c^{n_i}$. Specifically, by Proposition 1, we have

$$\operatorname{grad}\mathcal{L} = (\operatorname{grad}_{W^1}\mathcal{L}, \operatorname{grad}_{b^1}\mathcal{L}, \cdots, \operatorname{grad}_{W^n}\mathcal{L}, \operatorname{grad}_{b^n}\mathcal{L}).$$

Furthermore, recall that on a product manifold $N = \prod N_i$, $|v|_{g_N} = \sqrt{\sum |v|^2_{g_i}}$ where g_i is the metric in the component N_i, and likewise that $\operatorname{dist}_N(a, b) = \sqrt{\sum \operatorname{dist}_{N_i}(a_i, b_i)^2}$. Note in particular that this implies that for any i,

$$\operatorname{dist}_{N_i}(a_i, b_i) \leqslant \operatorname{dist}_N(a, b). \tag{5}$$

It is elementary that if each of a family of functions f_i are C_i-Lipschitz, respectively, then $\sqrt{\sum f_i^2}$ is $\sqrt{\sum C_i^2}$-Lipschitz, so we will show that the components of the gradient are each locally Lipschitz on their respective components of the product to show that the entire gradient is Lipschitz as well.

For a bias component $b_i^t \in \mathbb{D}_c^{n_i}$ at the iteration t, we can write down the following:

$$\begin{aligned}
&|\operatorname{grad}_{b_i}\mathcal{L}(b_i^{t+1}) - P_{b_i^t\to b_i^{t+1}}\operatorname{grad}_{b_i}\mathcal{L}(b_i^t)|_{g(b_i^{t+1})}\\
&= (\lambda^c_{b_i^{t+1}})^2||(\lambda^c_{b_i^{t+1}})^{-2}\nabla_{b_i}\mathcal{L}(b_i^{t+1}) - P_{b_i^t\to b_i^{t+1}}(\lambda^c_{b_i^t})^{-2}\nabla_{b_i}\mathcal{L}(b_i^t)||\\
&\leqslant (\lambda^c_{b_i^{t+1}})^2\Big[||(\lambda^c_{b_i^{t+1}})^{-2}\nabla_{b_i}\mathcal{L}(b_i^{t+1}) - (\lambda^c_{b_i^{t+1}})^{-2}\nabla_{b_i}\mathcal{L}(b_i^t)||\\
&\quad + ||(\lambda^c_{b_i^{t+1}})^{-2}\nabla_{b_i}\mathcal{L}(b_i^t) - (\lambda^c_{b_i^t})^{-2}\nabla_{b_i}\mathcal{L}(b_i^t)||\\
&\quad + ||(\lambda^c_{b_i^t})^{-2}\nabla_{b_i}\mathcal{L}(b_i^t) - P_{b_i^t\to b_i^{t+1}}(\lambda^c_{b_i^t})^{-2}\nabla_{b_i}\mathcal{L}(b_i^t)||\Big]\\
&= ||\nabla_{b_i}\mathcal{L}(b_i^{t+1}) - \nabla_{b_i}\mathcal{L}(b_i^t)|| + \left|1-\left(\frac{\lambda^c_{b_i^{t+1}}}{\lambda^c_{b_i^t}}\right)^2\right|||\nabla_{b_i}\mathcal{L}(b_i^t)||\\
&\quad + \left(\frac{\lambda^c_{b_i^{t+1}}}{\lambda^c_{b_i^t}}\right)^2|I - P_{b_i^t\to b_i^{t+1}}|\,||\nabla_{b_i}\mathcal{L}(b_i^t)||.
\end{aligned}$$

We will control these three terms. By compactness, on H_i, the Euclidean and hyperbolic distances differ by at most a constant factor C_1. Together with (5), it thus suffices to bound these quantities by a multiple of $||b_i^{t+1} - b_i^t||$, since

$$||b_i^{t+1} - b_i^t|| \leqslant C_1\operatorname{dist}_{\mathbb{D}_c^n}(b_i^{t+1}, b_i^t) \leqslant \operatorname{dist}_M(\mathbf{W}_{t+1}, \mathbf{W}_t).$$

First recall that the Möbius addition is given by

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

which is a rational function in the components of x and y, and so it is smooth except at its poles. Moreover, by Cauchy-Schwarz, the denominator is nonvanishing when $||x|| < 1/\sqrt{c}$ and $||y|| < 1/\sqrt{c}$. Thus, the Möbius addition is smooth on the Poincaré ball. Also, by Lemma 2 from [4], we have

$$\begin{aligned}
\exp_x^c(v) =& x\oplus_c\left(\tanh\left(\sqrt{c}\frac{\lambda_x^c||v||}{2}\right)\frac{v}{\sqrt{c}||v||}\right),\\
\log_x^c(y) =& \frac{2}{\sqrt{c}\lambda_x^c}\tanh^{-1}(\sqrt{c}||-x\oplus_c y||)\frac{-x\oplus_c y}{||-x\oplus_c y||},
\end{aligned}$$

which are smooth as well. Inspecting (1) and (4), and since $W^{\otimes_c}x = \exp_0^c(W\log_0^c(x))$, we see that $\mathcal{L}$ is written as compositions of these functions and the C^2 activation functions σ_i, plus a smooth regularization term, and so $\mathcal{L}$ is C^2 on $\mathbb{D}_c^{n_i}$. Thus, its Hessian (in the calculus sense) is continuous and thus bounded on compact sets, so $\nabla\mathcal{L}$ is locally C_2-Lipschitz in the Euclidean sense for some constant C_2. Thus, we obtain a control on the first term by

$$||\nabla_{b_i}\mathcal{L}(b_i^{t+1}) - \nabla_{b_i}\mathcal{L}(b_i^t)|| \leqslant C_2||b_i^{t+1} - b_i^t||.$$

Now since $\lambda_x^c = \frac{2}{1-c||x||^2}$, the coefficient of the second term can be rewritten as

$$\begin{aligned} 1-\left(\frac{\lambda^c_{b_i^{t+1}}}{\lambda^c_{b_i^t}}\right)^2 &= 1-\frac{1-c||b_i^t||^2}{1-c||b_i^{t+1}||^2} \\ &= \frac{c||b_i^t||^2-c||b_i^{t+1}||^2}{1-c||b_i^{t+1}||^2} \\ &= \frac{c||b_i^t||+c||b_i^{t+1}||}{1-c||b_i^{t+1}||^2}(||b_i^t||-||b_i^{t+1}||) \\ &\leqslant C_3||b_i^{t+1}-b_i^t||, \end{aligned}$$

where C_3 is a bound on the coefficient term in H_i since it is continuous.

For the third term, by the continuity we immediately can bound

$$\left(\frac{\lambda^c_{b_i^{t+1}}}{\lambda^c_{b_i^t}}\right)^2 ||\nabla_{b_i}\mathcal{L}(b_i^t)|| \leqslant C_4$$

on H_i, so we turn our attention to the remaining factor $|I-P_{b_i^t\to b_i^{t+1}}|$. Since parallel transport along the Levi-Civita connection is metric preserving, we can write $P_{b_i^t\to b_i^{t+1}} = \frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}U$ for U a rotation. Then we decompose

$$\begin{aligned} |I-P_{b_i^t\to b_i^{t+1}}| &= \left|I-\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}U\right| \\ &\leqslant \left|I-\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}I\right| + \left|\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}I-\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}U\right| \\ &= \left(1-\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}\right)+\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}}|I-U|. \end{aligned}$$

Analogous to the above, on H_i we have

$$\begin{aligned} &1-\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}} = \frac{c||b_i^{t+1}||^2-c||b_i^t||^2}{1-c||b_i^t||^2} \leqslant C_5||b_i^{t+1}-b_i^t||, \\ &\frac{\lambda^c_{b_i^t}}{\lambda^c_{b_i^{t+1}}} \leqslant C_6. \end{aligned}$$

As we said, U is a rotation. The geodesic through b_i^t and b_i^{t+1} is an arc of a circle, which crosses the boundary perpendicularly, and so the center of this circle is outside the unit ball. In particular, if r is the radius of this circle, then $r > d = \operatorname{dist}_{\mathbb{R}^{n_i}}(H_i, \partial\mathbb{D}_c^{n_i})$. The angle θ by which U rotates is precisely the central angle of the arc $\widehat{b_i^t b_i^{t+1}}$ (see Fig. 1). For any unit vector x, we have

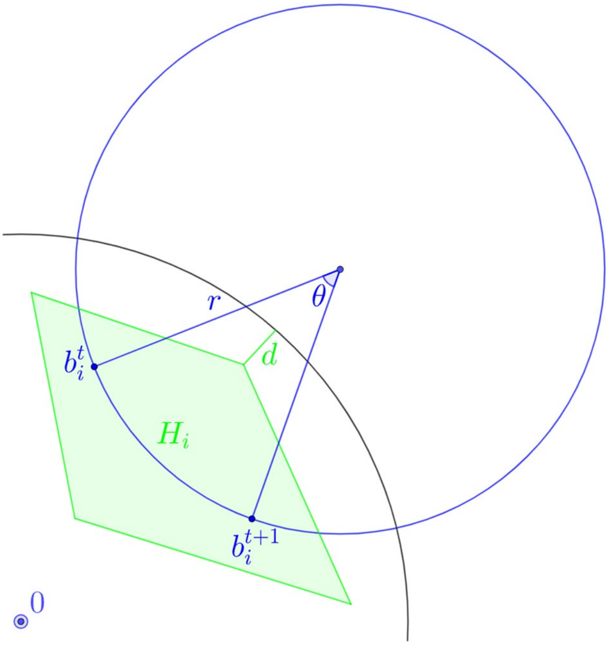


Fig. 1 The transported vector is rotated by angle θ

$$|(I - U)x| = \frac{||b_i^{t+1} - b_i^t||}{r} \leqslant \frac{||b_i^{t+1} - b_i^t||}{d},$$

which shows that this term is Lipschitz as well, and specifically that

$$\left(\frac{\lambda^c_{b_i^{t+1}}}{\lambda^c_{b_i^t}}\right)^2 |I - P_{b_i^t \to b_i^{t+1}}| \, ||\nabla_{b_i}\mathcal{L}(b_i^t)|| \leqslant C_4(C_5 + C_6/d)||b_i^{t+1} - b_i^t||.$$

Therefore, $\operatorname{grad}_{b_i}\mathcal{L}$ is Lipschitz on K in the Riemannian sense, with the coefficient $C_1(C_2+ C_3 + C_4(C_5 + C_6/d))$.

For the weights, $\operatorname{grad}_{W_i}\mathcal{L} = \nabla_{W_i}\mathcal{L}$ since the weights are Euclidean. By the same considerations as above, the loss function is C^2 with respect to W as well, so we immediately obtain that $\operatorname{grad}_W\mathcal{L}$ is locally Lipschitz since its Hessian is continuous, and thus bounded on compact sets.

4 Experiments

We evaluate these methods on two benchmarks: the realizable case where the ground-truth output is given by another network, the MNIST dataset, and the WordNet dataset [7]. Note that, although we required a dynamic choice of step size above, our results here all use a constant step size, which suggests that a dynamic step size is not fundamentally required.

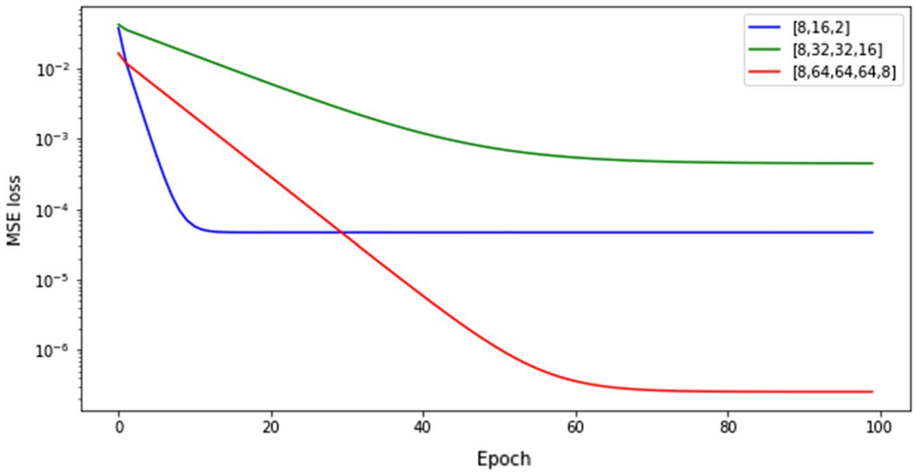
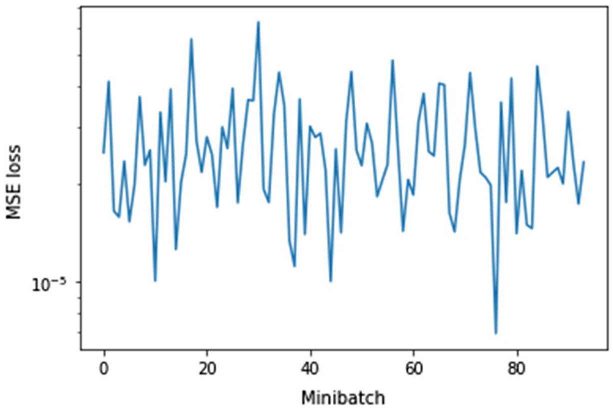


Fig. 2 Training by RSGD in the realizable case

Fig. 3 Loss after each sample in one epoch of training



4.1 Realizable Case

We train by RSGD a model f with hidden hyperbolic layers and a Euclidean output layer, with labels y_i generated by a ground-truth network $y_i = f(\mathbf{W}^*, x_i)$ of the same architecture. The dataset is $x_i \sim \exp_0^c(\mathcal{N}(0, I))$. (This is the pseudohyperbolic Gaussian [8].)

Our results for networks with various depth and width are shown in Fig. 2. We can see that training does appear to converge regardless of depth.

Although losses appear to decrease smoothly over epochs, by inspecting samples within an epoch, the stochasticity can be observed (Fig. 3).

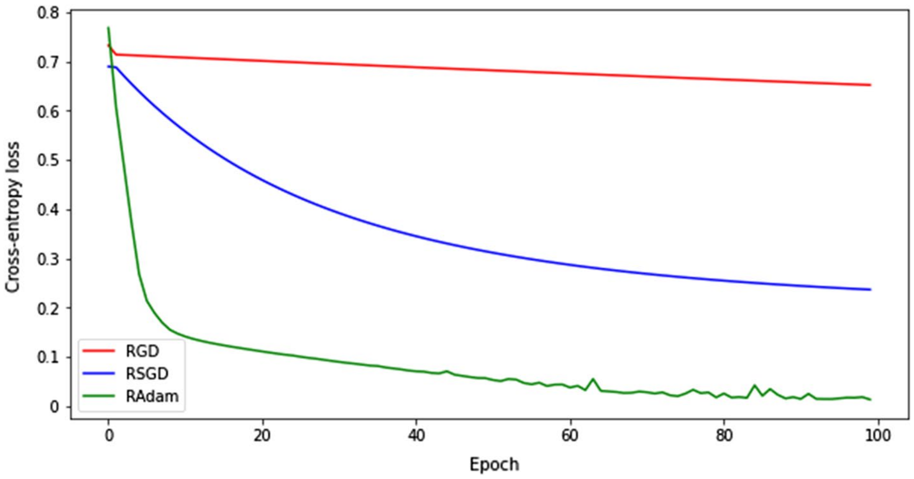


Fig. 4 Training by RGD, RSGD, and RAdam on WordNet `mammals`

4.2 WordNet

WordNet is a large database of English noun synsets (collections of synonyms), together with hyponymy relations ("is-a" relationships). This means that the WordNet dataset has the hierarchical structure of a directed tree, which is the type of problem on which we expect hyperbolic networks to perform well in practice. We first generate a hyperbolic embedding of the transitive closure of the WordNet `mammal` subtree into $\mathbb{D}^2$, as in [10].

We then train a classifier with 2 hyperbolic hidden layers of width 32 to determine whether a given point in the embedding represents an aquatic mammal (i.e., is a child node of the `aquatic_mammal` synset), using each of the training algorithms discussed in this paper. Results are shown in Fig. 4. As expected, we see that each algorithm does converge, with RAdam converging much faster and achieving much better results before plateauing.

5 Conclusion

We showed that HFFNNs are guaranteed to converge under training by RSGD, given some natural conditions, regardless of the choice of curvature or the number or size of layers. Experimentally, we use our own implementation to show that the convergence rate is reasonable. Furthermore, we implement RAdam and show that it provides substantial speedup in practice.

Acknowledgements The work was partially supported by NSF Grants DMS-1854434, DMS-1952644, and DMS-2151235 at UC Irvine, and Bao Wang is supported by NSF Grants DMS-1924935, DMS-1952339, DMS-2110145, DMS-2152762, and DMS-2208361, and DOE Grants DE-SC0021142 and DE-SC0002722.

Data Availability The data that support the findings of this study are available from the corresponding author, Wes Whiting, upon reasonable request.

Compliance with Ethical Standards

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Bécigneul, G., Ganea, O.-E.: Riemannian adaptive optimization methods. arXiv:1810.00760 (2019)
2. Bonnabel, S.: Stochastic gradient descent on Riemannian manifolds. IEEE Trans. Autom. Control **58**(9), 2217–2229 (2013)
3. De Sa, C., Gu, A., Ré, C., Sala, F.: Representation tradeoffs for hyperbolic embeddings. CoRR, arXiv:1804.03329 (2018)
4. Ganea, O.-E., Bécigneul, G., Hofmann, T.: Hyperbolic neural networks. CoRR, arXiv:1805.09112 (2018)
5. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980v9 (2014)
6. Kratsios, A., Bilokopytov, I.: Non-Euclidean universal approximation. CoRR, arXiv:2006.02341 (2020)
7. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: Introduction to WordNet: an on-line lexical database. Int. J. Lexicogr. **3**(4), 235–244 (1990)
8. Nagano, Y., Yamaguchi, S., Fujita, Y., Koyama, M.: A wrapped normal distribution on hyperbolic space for gradient-based learning. arXiv.1902.02992 (2019)
9. Neto, J.C., De Lima, L., Oliveira, P.: Geodesic algorithms in Riemannian geometry. Balkan J. Geom. Appl. (BJGA) **3**, 01 (1998)
10. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. arXiv:1705.08039v2 (2017)
11. Peng, W., Varanka, T., Mostafa, A., Shi, H., Zhao, G.: Hyperbolic deep neural networks: a survey. arXiv:2101.04562 (2021)
12. Ungar, A.A.: A Gyrovector Space Approach to Hyperbolic Geometry. Morgan & Claypool Publishers, San Rafael (2009)