Differentiating through the Fréchet Mean

Aaron Lou [*1] **Isay Katsman** [*1] **Qingxuan Jiang** [*1] **Serge Belongie** [1] **Ser-Nam Lim** [2] **Christopher De Sa** [1]

Abstract

Recent advances in deep representation learning on Riemannian manifolds extend classical deep learning operations to better capture the geometry of the manifold. One possible extension is the Fréchet mean, the generalization of the Euclidean mean; however, it has been difficult to apply because it lacks a closed form with an easily computable derivative. In this paper, we show how to differentiate through the Fréchet mean for arbitrary Riemannian manifolds. Then, focusing on hyperbolic space, we derive explicit gradient expressions and a fast, accurate, and hyperparameter-free Fréchet mean solver. This fully integrates the Fréchet mean into the hyperbolic neural network pipeline. To demonstrate this integration, we present two case studies. First, we apply our Fréchet mean to the existing Hyperbolic Graph Convolutional Network, replacing its projected aggregation to obtain state-of-the-art results on datasets with high hyperbolicity. Second, to demonstrate the Fréchet mean's capacity to generalize Euclidean neural network operations, we develop a hyperbolic batch normalization method that gives an improvement parallel to the one observed in the Euclidean setting.

1. Introduction

Recent advancements in geometric representation learning have utilized hyperbolic space for tree embedding tasks (Nickel & Kiela, 2017; 2018; Yu & De Sa, 2019). This is due to the natural non-Euclidean structure of hyperbolic space, in which distances grow exponentially as one moves away from the origin. Such a geometry is naturally equipped to embed trees, since if we embed the root of the tree near the origin and layers at successive radii, the geometry of hyperbolic space admits a natural hierarchical structure. More recent work has focused specifically on developing neural networks that exploit the structure of hyperbolic space (Ganea et al., 2018; Tifrea et al., 2019; Chami et al., 2019; Liu et al., 2019).

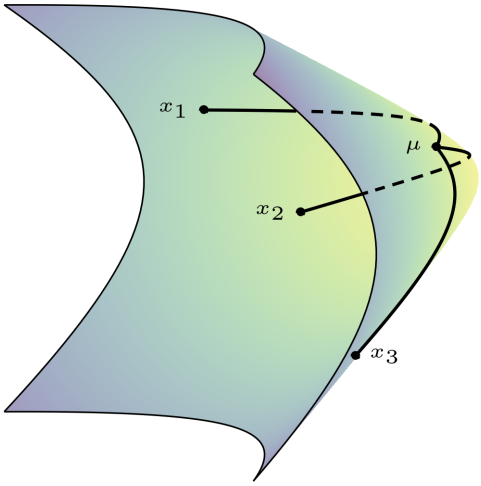

Figure 1. Depicted above is the Fréchet mean, μ, of three points, x_1, x_2, x_3 in hyperbolic space. As one can see, the Fréchet mean conforms with the geometry of the hyperboloid and is vastly different from the standard Euclidean mean.

One well-motivated approach to defining better architectures comes from generalizing fundamental operations in Euclidean space to hyperbolic space. One such fundamental operation is the Euclidean mean, which extends to the Fréchet mean in non-Euclidean geometries (Fréchet, 1948). Unlike the Euclidean mean, the Fréchet mean does not have a closed-form solution, and its computation involves an argmin operation that cannot be easily differentiated. This hampers the development of non-Euclidean counterparts of neural network structures that require mean computations. In this paper, we extend the methods in Gould et al. (2016) to differentiate through the Fréchet mean, and we apply our methods to downstream tasks. Concretely, our paper's contributions are that:

- We derive closed-form gradient expressions for the Fréchet mean on Riemannian manifolds.
- For the case of hyperbolic space, we present a novel

[*]Equal contribution [1]Department of Computer Science, Cornell University, NY, Ithaca, USA [2]Facebook AI, NY, New York, USA. Correspondence to: Aaron Lou <al968@cornell.edu>, Isay Katsman <isk22@cornell.edu>, Qingxuan Jiang <qj46@cornell.edu>.

Preprint. Under Review.

algorithm for quickly computing the Fréchet mean and a closed-form expression for its derivative.

- We use our Fréchet mean computation in place of the neighborhood aggregation step in Hyperbolic Graph Convolution Networks (Chami et al., 2019) and achieve state-of-the-art results on graph datasets with high hyperbolicity.
- We introduce a fully differentiable Riemannian batch normalization method which mimics the procedure and benefit of standard Euclidean batch normalization.

2. Related Work

Uses of Hyperbolic Space in Machine Learning. The usage of hyperbolic embeddings first appeared in Kleinberg (2007), in which the author uses them in a greedy embedding algorithm. Later analyses by Sarkar (2011) and Sala et al. (2018) demonstrate the empirical and theoretical improvement of this approach. However, only recently in Nickel & Kiela (2017; 2018) was this method extended to deep neural networks. Since then, models such as those in Ganea et al. (2018); Chami et al. (2019); Liu et al. (2019) have leveraged hyperbolic space operations to obtain better embeddings.

Fréchet Mean. The Fréchet mean (Fréchet, 1948), as the generalization of the classical Euclidean mean, offers a plethora of applications in downstream tasks. As a mathematical construct, the Fréchet mean has been thoroughly studied in texts such as Karcher (1977); Charlier (2013); Bacák (2014).

However, the Fréchet mean is an operation not without complications; the general formulation requires an argmin operation and offers no closed-form solution. As a result, both computation and differentiation are problematic, although previous works have attempted to resolve such difficulties.

To address computation, Gu et al. (2019) show that a Riemannian gradient descent algorithm recovers the Fréchet mean in linear time for products of Riemannian model spaces. However, without a tuned learning rate, it is too hard to ensure performance. Brooks et al. (2019) instead use the Karcher Flow Algorithm (Karcher, 1977); although this method is manifold-agnostic, it is slow in practice. We address such existing issues in the case of hyperbolic space by providing a fast, hyperparameter-free algorithm for computing the Fréchet mean.

Some works have addressed the differentiation issue by circumventing it, instead relying on pseudo-Fréchet means. In Law et al. (2019), the authors utilize a novel squared Lorentzian distance (as opposed to the canonical distance for hyperbolic space) to derive explicit formulas for the Fréchet mean in pseudo-hyperbolic space. In Chami et al. (2019), the authors use an aggregation method in the tangent space as a substitute. Our work, to the best of our knowledge, is the first to provide explicit derivative expressions for the Fréchet mean on Riemannian manifolds.

Differentiating through the argmin. Theoretical foundations of differentiating through the argmin operator have been provided in Gould et al. (2016). Similar methods have subsequently been used to develop differentiable optimization layers in neural networks (Amos & Kolter, 2017; Agrawal et al., 2019).

Given that the Fréchet mean is an argmin operation, one might consider utilizing the above differentiation techniques. However, a naïve application fails, as the Fréchet mean's argmin domain is a manifold, and Gould et al. (2016) deals specifically with Euclidean space. Our paper extends this theory to the case of general Riemannian manifolds, thereby allowing the computation of derivatives for more general argmin problems, and, in particular, for computing the derivative of the Fréchet mean in hyperbolic space.

3. Background

In this section, we establish relevant definitions and formulas of Riemannian manifolds and hyperbolic spaces. We also briefly introduce neural network layers in hyperbolic space.

3.1. Riemannian Geometry Background

Here we provide some of the useful definitions from Riemannian geometry. For a more in-depth introduction, we refer the interested reader to our Appendix C or texts such as Lee (2003) and Lee (1997).

Manifold and tangent space: An n-dimensional manifold $\mathcal{M}$ is a topological space that is locally homeomorphic to $\mathbb{R}^n$. The tangent space $T_x\mathcal{M}$ at x is defined as the vector space of all tangent vectors at x and is isomorphic to $\mathbb{R}^n$. We assume our manifolds are smooth, i.e. the maps are diffeomorphic. The manifold admits local coordinates $(x_1, \dots, x_n)$ which form a basis $(dx_1, \dots, dx_n)$ for the tangent space.

Riemannian metric and Riemannian manifold: For a manifold $\mathcal{M}$, a Riemannian metric $\rho = (\rho_x)_{x\in\mathcal{M}}$ is a smooth collection of inner products $\rho_x : T_x\mathcal{M} \times T_x\mathcal{M} \to \mathbb{R}$ on the tangent space of every $x \in \mathcal{M}$. The resulting pair $(\mathcal{M}, \rho)$ is called a Riemannian manifold. Note that ρ induces a norm in each tangent space $T_x\mathcal{M}$, given by $\|\vec{v}\|_\rho = \sqrt{\rho_x(\vec{v}, \vec{v})}$ for any $\vec{v} \in T_x\mathcal{M}$. We oftentimes associate ρ to its matrix form (ρ_{ij}) where $\rho_{ij} = \rho(dx_i, dx_j)$ when given local coordinates.

Geodesics and induced distance function: For a curve $\gamma : [a, b] \to \mathcal{M}$, we define the length of γ to be $L(\gamma) =$

Table 1. Summary of operations in the Poincaré ball model and the hyperboloid model ($K < 0$)

	Poincaré Ball	**Hyperboloid**
Manifold	$\mathbb{D}_K^n = \{x \in \mathbb{R}^n : \langle x, x\rangle_2 < -\frac{1}{K}\}$	$\mathbb{H}_K^n = \{x \in \mathbb{R}^{n+1} : \langle x, x\rangle_{\mathcal{L}} = \frac{1}{K}\}$
Metric	$g_x^{\mathbb{D}_K} = (\lambda_x^K)^2 g^{\mathbb{E}}$ where $\lambda_x^K = \frac{2}{1+K\|x\|_2^2}$ and $g^{\mathbb{E}} = I$	$g_x^{\mathbb{H}_K} = \eta$, where η is I except $\eta_{0,0} = -1$
Distance	$d_{\mathbb{D}}^K(x,y) = \frac{1}{\sqrt{\|K\|}} \cosh^{-1}\left(1 - \frac{2K\|x-y\|_2^2}{(1+K\|x\|_2^2)(1+K\|y\|_2^2)}\right)$	$d_{\mathbb{H}}^K(x,y) = \frac{1}{\sqrt{\|K\|}} \cosh^{-1}(\|K\|\langle x, y\rangle_{\mathcal{L}})$
Exp map	$\exp_x^K(v) = x \oplus_K \left(\tanh\left(\sqrt{\|K\|}\frac{\lambda_x^K\|v\|_2}{2}\right)\frac{v}{\sqrt{\|K\|}\|v\|_2}\right)$	$\exp_x^K(v) = \cosh(\sqrt{\|K\|}\|\|v\|\|_{\mathcal{L}})x + v\frac{\sinh(\sqrt{\|K\|}\|\|v\|\|_{\mathcal{L}})}{\sqrt{\|K\|}\|\|v\|\|_{\mathcal{L}}}$
Log map	$\log_x^K(y) = \frac{2}{\sqrt{\|K\|}\lambda_x^K}\tanh^{-1}(\sqrt{\|K\|}\|-x \oplus_K y\|_2)\frac{-x\oplus_K y}{\|-x\oplus_K y\|_2}$	$\log_x^K(y) = \frac{\cosh^{-1}(K\langle x,y\rangle_{\mathcal{L}})}{\sinh(\cosh^{-1}(K\langle x,y\rangle_{\mathcal{L}}))}(y - K\langle x,y\rangle_{\mathcal{L}}x)$
Transport	$PT_{x\to y}^K(v) = \frac{\lambda_x^K}{\lambda_y^K}\operatorname{gyr}[y,-x]v$	$PT_{x\to y}^K(v) = v - \frac{K\langle y,v\rangle_{\mathcal{L}}}{1+K\langle x,y\rangle_{\mathcal{L}}}(x+y)$

Table 2. Summary of hyperbolic counterparts of Euclidean operations in neural networks

Operation	**Formula**
Matrix-vector multiplication	$A \otimes^K x = \exp_0^K(A\log_0^K(x))$
Bias translation	$x \oplus^K b = \exp_x(PT_{0\to x}^K(b))$
Activation function	$\sigma^{K_1,K_2}(x) = \exp_0^{K_1}(\sigma(\log_0^{K_2}(x)))$

$\int_a^b \|\gamma'(t)\|_\rho \, dt$. For $x, y \in \mathcal{M}$, the distance $d(x,y) = \inf L(\gamma)$ where γ is any curve such that $\gamma(a) = x, \gamma(b) = y$. A geodesic γ_{xy} from x to y, in our context, should be thought of as a curve that minimizes this length[1].

Exponential and logarithmic map: For each point $x \in \mathcal{M}$ and vector $\vec{v} \in T_x\mathcal{M}$, there exists a unique geodesic $\gamma : [0,1] \to \mathcal{M}$ where $\gamma(0) = x, \gamma'(0) = \vec{v}$. The exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ is defined as $\exp_x(\vec{v}) = \gamma(1)$. Note that this is an isometry, i.e. $\|\vec{v}\|_\rho = d(x, \exp_x(\vec{v}))$. The logarithmic map $\log_x : \mathcal{M} \to T_x\mathcal{M}$ is defined as the inverse of $\exp_x$, although this can only be defined locally[2].

Parallel transport: For $x, y \in \mathcal{M}$, the parallel transport $PT_{x\to y} : T_x\mathcal{M} \to T_y\mathcal{M}$ defines a way of transporting the local geometry from x to y along the unique geodesic that preserves the metric tensors.

3.2. Hyperbolic Geometry Background

We now examine hyperbolic space, which has constant curvature $K < 0$, and provide concrete formulas for computation. The two equivalent models of hyperbolic space frequently used are the Poincaré ball model and the hyperboloid model. We denote $\mathbb{D}_K^n$ and $\mathbb{H}_K^n$ as the n-dimensional Poincaré ball and hyperboloid models with curvature $K < 0$, respectively.

3.2.1. Basic Operations

Inner products: We define $\langle x, y\rangle_2$ to be the standard Euclidean inner product and $\langle x, y\rangle_{\mathcal{L}}$ to be the Lorentzian inner product $-x_0y_0 + x_1y_1 + \cdots + x_ny_n$.

Gyrovector operations: For $x, y \in \mathbb{D}_K^n$, the Möbius addition (Ungar, 2008) is

$$x \oplus_K y = \frac{(1 - 2K\langle x,y\rangle_2 - K\|y\|_2^2)x + (1 + K\|x\|_2^2)y}{1 - 2K\langle x,y\rangle_2 + K^2\|x\|_2^2\|y\|_2^2} \tag{1}$$

This induces Möbius subtraction $\ominus_K$ which is defined as $x \ominus_K y = x \oplus_K -y$. In the theory of gyrogroups, the notion of the gyration operator (Ungar, 2008) is given by

$$\operatorname{gyr}[x,y]v = \ominus_K(x \oplus_K y) \oplus_K (x \oplus_K (y \oplus_K v)) \tag{2}$$

Riemannian operations on hyperbolic space: We summarize computations for the Poincaré ball model and the hyperboloid model in Table 1.

3.3. Hyperbolic Neural Networks

Introduced in Ganea et al. (2018), hyperbolic neural networks provide a natural generalization of standard neural networks.

Hyperbolic linear layer: Recall that a Euclidean linear layer is defined as $f : \mathbb{R}^m \to \mathbb{R}^n$, $f = \sigma(Ax + b)$ where $A \in \mathbb{R}^{n\times m}$, $x \in \mathbb{R}^m$, $b \in \mathbb{R}^n$ and σ is some activation function.

[1] Formally, geodesics are curves with 0 acceleration w.r.t. the Levi-Civita connection. There are geodesics which are not minimizing curves, such as the larger arc between two points on a great circle of a sphere; hence this clarification is important.

[2] Problems in definition arise in the case of conjugate points (Lee, 1997). However, exp is a local diffeomorphism by the inverse function theorem.

With analogy to Euclidean layers, a hyperbolic linear layer $g : \mathbb{H}^m \to \mathbb{H}^n$ is defined by $g = \sigma^{K,K}(A \otimes^K x \oplus^K b)$, where $A \in \mathbb{R}^{n\times m}$, $x \in \mathbb{H}^m$, $b \in \mathbb{H}^n$, and we replace the operations by hyperbolic counterparts outlined in Table 2.

Hyperbolic neural networks are defined as compositions of these layers, similar to how conventional neural networks are defined as compositions of Euclidean layers.

4. A Differentiable Fréchet Mean Operation for General Riemannian Manifolds

In this section, we provide a few theorems that summarize our method of differentiating through the Fréchet mean.

4.1. Background on the Fréchet Mean

Fréchet mean and variance: On a Riemannian manifold $(\mathcal{M}, \rho)$, the Fréchet mean $\mu_{fr} \in \mathcal{M}$ and Fréchet variance $\sigma^2_{fr} \in \mathbb{R}$ of a set of points $\mathcal{B} = \{x^{(1)}, \cdots, x^{(t)}\}$ with each $x^{(l)} \in \mathcal{M}$ are defined as the solution and optimal values of the following optimization problem (Bacák, 2014):

$$\mu_{fr} = \arg\min_{\mu\in\mathcal{M}} \frac{1}{t}\sum_{l=1}^{t} d(x^{(l)}, \mu)^2 \tag{3}$$

$$\sigma^2_{fr} = \min_{\mu\in\mathcal{M}} \frac{1}{t}\sum_{l=1}^{t} d(x^{(l)}, \mu)^2 \tag{4}$$

In Appendix A, we provide proofs to illustrate that this definition is a natural generalization of Euclidean mean and variance.

The Fréchet mean can be further generalized with an arbitrary re-weighting. In particular, for positive weights $\{w_l\}_{l\in[t]}$, we can define the weighted Fréchet mean as:

$$\mu_{fr} = \arg\min_{\mu\in\mathcal{M}} \sum_{l=1}^{t} w_l \cdot d(x^{(l)}, \mu)^2 \tag{5}$$

This generalizes the weighted Euclidean mean to Riemannian manifolds.

4.2. Differentiating Through the Fréchet Mean

All known methods for computing the Fréchet mean rely on some sort of iterative solver (Gu et al., 2019). While backpropagating through such a solver is possible, it is computationally inefficient and suffers from numerical instabilities akin to those found in RNNs (Pascanu et al., 2013). To circumvent these issues, recent works compute gradients at the solved value instead of differentiating directly, allowing for full neural network integration (Chen et al., 2018; Pogančić et al., 2020). However, to the best of our knowledge, no paper has investigated backpropagation on a manifold-based convex optimization solver. Hence, in this section, we construct the gradient, relying on the fact that the Fréchet mean is an argmin operation.

4.2.1. Differentiating through the argmin Operation

Motivated by previous works on differentiating argmin problems (Gould et al., 2016), we propose a generalization which allows us to differentiate the argmin operation on the manifold. The full theory is presented in Appendix D.

4.2.2. Construction of the Fréchet mean derivative

Since the Fréchet mean is an argmin operation, we can apply the theorems in Appendix D. Note that we define $\widetilde{\nabla}$ as the total derivative (or Jacobian) for notational convenience.

Theorem 4.1. *Let $(\mathcal{M}, \rho)$ be an n-dimensional Riemannian manifold, and let $\{x\} = (x^{(1)}, \ldots, x^{(t)}) \in (\mathcal{M})^t$ be a set of data points with weights $w_1, \ldots, w_t \in \mathbb{R}^+$. Let $f : (\mathcal{M})^t \times \mathcal{M} \to \mathcal{M}$ be given by $f(\{x\}, y) = \sum_{l=1}^{t} w_l \cdot d(x^{(l)}, y)^2$ and $\overline{x} = \mu_{fr}(\{x\}) = \arg\min_{y\in\mathcal{M}} f(\{x\}, y)$ be the Fréchet mean. Then with respect to local coordinates we have*

$$\widetilde{\nabla}_{x^{(i)}}\mu_{fr}(\{x\}) = -f_{YY}(\{x\}, \overline{x})^{-1} f_{X^{(i)}Y}(\{x\}, \overline{x}) \tag{6}$$

where the functions $f_{X^{(i)}Y}(\{x\}, y) = \widetilde{\nabla}_{x^{(i)}}\nabla_y f(\{x\}, y)$ and $f_{YY}(\{x\}, y) = \nabla^2_{yy} f(\{x\}, \overline{x})$ are defined in terms of local coordinates.

Proof. This is a special case of Theorem D.1 in the appendix. This is because the Fréchet objective function f is a twice differentiable real-valued function for specific $x^{(i)}$ and y (under our geodesic formulation); thus we obtain the desired formulation. The full explanation can be found in Remark D. □

While the above theorem gives a nice theoretical framework with minimal assumptions, it is in practice too unwieldy to apply. In particular, the requirement of local coordinates renders most computations difficult. We now present a version of the above theorem which assumes that the manifold is embedded in Euclidean space[3].

Theorem 4.2. *Assume the conditions and values in Theorem 4.1. Furthermore, assume $\mathcal{M}$ is embedded (not necessarily isometrically) in $\mathbb{R}^m$ with $m \geq n$, then we can write*

$$\widetilde{\nabla}_{x^{(i)}}\mu_{fr}(\{x\}) = -f^p_{YY}(\{x\}, \overline{x})^{-1} f^p_{X^{(i)}Y}(\{x\}, \overline{x}) \tag{7}$$

[3]We also present a more general way to take the derivative that drops this restriction via an exponential map-based parameterization in Appendix D.

where $f^p_{YY}(\{x\}, y) = \widetilde{\nabla}_y(\text{proj}_{T_{\overline{x}}\mathcal{M}} \circ \nabla_y f)(\{x\}, y)$, $f^p_{X^{(i)}Y}(\{x\}, y) = \widetilde{\nabla}_{x^{(i)}}(\text{proj}_{T_{\overline{x}}\mathcal{M}} \circ \nabla_y f)(\{x\}, y)$, *and* $\text{proj}_{T_{\overline{x}}\mathcal{M}} : \mathbb{R}^m \to T_{\overline{x}}\mathcal{M} \cong \mathbb{R}^n$ *is the linear subspace projection operator.*

Proof. Similar to the relationship between Theorem 4.1 and Theorem D.1, this is a special case of Theorem D.3 in the appendix. □

5. Hyperbolic Fréchet Mean

Although we have provided a formulation for differentiating through the Fréchet mean on general Riemannian manifolds, to properly integrate it in the hyperbolic setting we need to address two major difficulties:

1. The lack of a fast forward computation.
2. The lack of an explicit derivation of a backpropagation formula.

Resolving these difficulties will allow us to define a Fréchet mean neural network layer for geometric, and specifically hyperbolic, machine learning tasks.

5.1. Forward Computation of the Hyperbolic Fréchet Mean

Previous forward computations fall into one of two categories: (1) fast, inaccurate computations which aim to approximate the true mean with a pseudo-Fréchet mean, or (2) slow, exact computations. In this section we focus on outperforming methods in the latter category, since we strive to compute the exact Fréchet mean (pseudo-means warp geometry).

5.1.1. Former Attempts at Computing the Fréchet Mean

The two existing algorithms for Fréchet mean computation are (1) Riemannian gradient-based optimization (Gu et al., 2019) and (2) iterative averaging (Karcher, 1977). However, in practice both algorithms are slow to converge even for simple synthetic examples of points in hyperbolic space. To overcome this difficulty, which can cripple neural networks, we propose the following algorithm that is much faster in practice.

5.1.2. Algorithm for Fréchet Mean Computation via First-order Bound

The core idea of our algorithm relies on the fact that the square of distance metric is a concave function for both the Poincaré ball and hyperboloid model. Intuitively, we select an initial "guess" and use a first-order bound to minimize the Fréchet mean objective. The concrete algorithm for the Poincaré ball model is given as Algorithm 1 above. Note that the algorithm is entirely hyperparameter-free and does not require setting a step-size. Moreover, we can prove that the algorithm is guaranteed to converge.

Algorithm 1 Poincaré model Fréchet mean algorithm

Inputs: Data points $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}^n_{-1} \subseteq \mathbb{R}^{n+1}$, weights $w_1, \cdots, w_t \in \mathbb{R}^+$.
Algorithm:
$y_0 = x^{(1)}$
Define $g(y) = \frac{2\,\text{arccosh}(1+2y)}{\sqrt{y^2+y}}$
for $k = 1, \cdots, T$:
  for $l = 1, 2, \cdots, t$:
    $\alpha_l = w_l \cdot g\left(\frac{\|x^{(l)} - y_{k-1}\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_{k-1}\|^2)}\right) \cdot \frac{1}{1-\|x^{(l)}\|^2}$
  $a = \sum_{l=1}^{t} \alpha_l$, $b = \sum_{l=1}^{t} \alpha_l x^{(l)}$, $c = \sum_{l=1}^{t} \alpha_l \|x^{(l)}\|^2$
  $y_k = \left(\frac{a+c-\sqrt{(a+c)^2 - 4\|b\|^2}}{2\|b\|^2}\right) b$
return y_T

Theorem 5.1. *Let* $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}^n_{-1}$ *be* t *points*[4] *in the Poincaré ball,* $w_1, \ldots, w_t \in \mathbb{R}^+$ *be their weights, and let their weighted Fréchet mean be the solution to the following optimization problem.*

$$\mu_{fr} = \arg\min_{y \in \mathbb{D}^n} f(y) \tag{8}$$

$$\text{where } f(y) = \sum_{l=1}^{t} w_l \cdot d_{\mathbb{D}^n}(x^{(l)}, y)^2$$

$$= \sum_{l=1}^{t} w_l \,\text{arccosh}^2\left(1 + 2 \cdot \frac{\|x^{(l)} - y\|^2}{(1 - \|x^{(l)}\|^2)(1 - \|y\|^2)}\right) \tag{9}$$

Then Algorithm 1 gives a sequence of points $\{y_k\}$ *such that their limit* $\lim_{k\to\infty} y_k = \mu_{fr}$ *converges to the Fréchet mean solution.*

Proof. See Theorem E.2 in the appendix. □

The algorithm and proof of convergence for the hyperboloid model are given in Appendix E.1 and are omitted here for brevity.

5.1.3. Empirical Comparison to Previous Fréchet Mean Computation Algorithms

To demonstrate the efficacy of our algorithm, we compare it to previous approaches on randomly generated data.

[4] Here we present the version for $K = -1$ for cleaner presentation. The generalization to arbitrary $K < 0$ is easy to compute, but clutters presentation.

Namely, we compare against a naïve Riemannian Gradient Descent (RGD) approach (Udrişte, 1994) and against the Karcher Flow algorithm (Karcher, 1977). We test our Fréchet mean algorithm against these methods on synthetic datasets of ten on-manifold randomly generated 16-dimensional points. We run all algorithms until they are within $\epsilon = 10^{-12}$ of the true Fréchet mean in norm, and report the number of iterations this takes in Table 3 for both hyperboloid (H) and Poincaré (P) models of hyperbolic space. Note that we significantly outperform the other algorithms. We also observe that by allowing 200x more computation, a grid search on the learning hyperparameter[5] in RGD obtains nearly comparable or better results (last row of Table 3 for both models). However, we stress that this requires much more computation, and note that our algorithm produces nearly the same result while being **hyperparameter-free**.

Table 3. Empirical computation of the Fréchet mean; the average number of iterations required to become accurate within $\epsilon = 10^{-12}$ of the true Fréchet mean are reported. 5 trials are conducted, and standard deviation is reported. The primary baselines are the RGD (Udrişte, 1994) and Karcher Flow (Karcher, 1977) algorithms. (H) refers to hyperboloid and (P) refers to Poincaré.

		Iterations
H	RGD ($lr = 0.01$)	$808.6_{\pm 29.41}$
	Karcher Flow	$60_{\pm 5.66}$
	Ours	$\mathbf{13.4}_{\pm 0.89}$
	RGD + Grid Search on lr	$27.8_{\pm 0.84}$
P	RGD ($lr = 0.01$)	$751.8_{\pm 14.39}$
	Karcher Flow	$70.6_{\pm 12.12}$
	Ours	$\mathbf{13.2}_{\pm 0.45}$
	RGD + Grid Search on lr	$10.8_{\pm 0.45}$

5.2. Backward Computation of the Hyperbolic Fréchet Mean

For the backward computation, we re-apply the general Riemannian theory for differentiating through the Fréchet mean in Section 4 to hyperbolic space. Since most autodifferentiation packages do not support manifold-aware higher order differentiation, we derive the gradients explicitly. We begin with the Poincaré ball model by setting $\mathcal{M} = \mathbb{D}_K^n$ and applying Theorem 4.2.

Theorem 5.2. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}_K^n \subseteq \mathbb{R}^n$ be t points in the Poincaré ball and $w_1, \ldots, w_t \in \mathbb{R}^+$ be the weights. Let their weighted Fréchet mean μ_{fr} be solution to the following optimization problem*

$$\mu_{fr}(x^{(1)}, \cdots, x^{(t)}) = \arg\min_{y \in \mathbb{D}_K^n} f(\{x\}, y) \tag{10}$$

$$\text{where } f(\{x\}, y) = \sum_{l=1}^{t} w_l \cdot d_{\mathbb{D}}^K(\{x\}, y)^2 = \sum_{l=1}^{t} \frac{w_l}{K} \operatorname{arccosh}^2\left(1 - \frac{2K||x^{(l)} - y||_2^2}{(1 + K||x^{(l)}||_2^2)(1 + K||y||_2^2)}\right) \tag{11}$$

Then the derivative of μ_{fr} with respect to $x^{(i)}$ is given by

$$\widetilde{\nabla}_{x^{(i)}} \mu_{fr}(\{x\}) = -f_{YY} f(\{x\}, \overline{x})^{-1} f_{X^{(i)}Y}(\{x\}, \overline{x}) \tag{12}$$

where $\overline{x} = \mu_{fr}(\{x\})$ and f_{YY}, $f_{X^{(i)}Y}$ are defined in Theorem 4.2 [6].

The full concrete derivation of the above terms for the geometry induced by this manifold choice is given in Appendix Theorem F.3.

Proof. This is a concrete application of Theorem 4.2. In particular since our manifold is embedded in $\mathbb{R}^n$ ($\mathbb{D}_K^n \subseteq \mathbb{R}^n$). Note that this is the total derivative in the ambient Euclidean space[7]. For the full proof see Theorem F.3 in the Appendix. □

The derivation for the hyperboloid model is given in Appendix F.2.

6. Case Studies

To demonstrate the efficacy of our developed theory, we investigate the following test settings. In the first setting, we directly modify the hyperbolic aggregation strategy in Hyperbolic GCNs (Chami et al., 2019) to use our differentiable Fréchet mean layer. This was the original intent[8] but was not feasible without our formulation. In the second setting, we introduce Hyperbolic Batch Normalization (HBN) as an extension of the regular Euclidean Batch Normalization (EBN). When combined with hyperbolic neural networks (Ganea et al., 2018), HBN exhibits benefits similar to those of EBN with Euclidean networks.

[5] The grid search starts from $lr = 0.2$ and goes to $lr = 0.4$ in increments of 0.01.

[6] The projection operation is trivial since $\dim \mathbb{R}^n = \dim \mathbb{D}_K^n$.

[7] To transform Euclidean gradients into Riemannian ones, simply multiply by inverse of the matrix of the metric.

[8] We quote directly from the paper Chami et al. (2019): "An analog of mean aggregation in hyperbolic space is the Fréchet mean, which, however, has no closed form solution. Instead, we propose to..."

6.1. Hyperbolic Graph Convolutional Neural Networks (HGCNs)

6.1.1. Original Framework

Introduced in Chami et al. (2019), Hyperbolic Graph Convolutional Networks (GCNs) provide generalizations of Euclidean GCNs to hyperbolic space. The proposed network architecture is based on three different layer types: feature transformation, activation, and attention-based aggregation.

Feature transformation: The hyperbolic feature transformation consists of a gyrovector matrix multiplication followed by a gyrovector addition.

$$h_i^l = (W^l \otimes^{K_{l-1}} x_i^{l-1}) \oplus^{K_{l-1}} b^l \tag{13}$$

Attention-based aggregation: Neighborhood aggregation combines local data at a node. It does so by projecting the neighbors using the logarithmic map at the node, averaging in the tangent space, and projecting back with the exponential map at the node. Note that the weights w_{ij} are positive and can be trained or defined by the graph adjacency matrix.

$$AGG^K(x_i) = \exp_{x_i}^K \left(\sum_{j \in \mathcal{N}(i)} w_{ij} \log_{x_i}^K x_j \right) \tag{14}$$

Activation: The activation layer applies a hyperbolic activation function.

$$x_i^l = \sigma^{\otimes^{K_{l-1}, K_l}}(y_i^l) \tag{15}$$

6.1.2. Proposed Changes

The usage of tangent space aggregation in the HGCN framework stemmed from the lack of a differentiable Fréchet mean operation. As a natural extension, we substitute our Fréchet mean in place of the aggregation layer.

6.1.3. Experimental Results

We use precisely the same architecture as in Chami et al. (2019), except we substitute all hyperbolic aggregation layers with our differentiable Fréchet mean layer. This allows us to achieve new state-of-the-art results on the Disease and Disease-M graph datasets (Chami et al., 2019) using the same data splits. These datasets induce ideal test tasks for hyperbolic learning since they have very low Gromov δ-hyperbolicity (Adcock et al., 2013), which indicates the structure is highly tree-like. Our results and comparison to the baseline are given in Table 4. We run experiments for 5 trials and report the mean and standard deviation. Due to practical considerations, we only test with the Poincaré model[9]. For reference, the strongest baseline results with the hyperboloid model are reported from Chami et al. (2019) (note that we outperform these results as well). On the rather non-hyperbolic CoRA (Sen et al., 2008) dataset, our performance is comparable to that of the best baseline. Note that this is similar to the performance exhibited by the vanilla HGCN. Hence we conjecture that when the underlying dataset is not hyperbolic in nature, we do not observe improvements over the best Euclidean baseline methods.

Table 4. ROC AUC results for Link Prediction (LP) on various graph datasets, averaged over 5 trials (with standard deviations). Graph hyperbolicity values are also reported (lower δ is more hyperbolic). Results are given for models learning in Euclidean (E), Hyperboloid (H), and Poincaré (P) spaces. Note that the best Euclidean method is GAT (Veličković et al., 2018) and is shown below for fair comparison on CoRA.

		Disease $\delta = 0$	Disease-M $\delta = 0$	CoRA $\delta = 11$
E	MLP	$72.6_{\pm 0.6}$	$55.3_{\pm 0.5}$	$83.1_{\pm 0.5}$
	GAT	$69.8_{\pm 0.3}$	$69.5_{\pm 0.4}$	$\mathbf{93.7}_{\pm 0.1}$
H	HNN	$75.1_{\pm 0.3}$	$60.9_{\pm 0.4}$	$89.0_{\pm 0.1}$
	HGCN	$90.8_{\pm 0.3}$	$78.1_{\pm 0.4}$	$92.9_{\pm 0.1}$
P	HGCN	$76.4_{\pm 8.9}$	$81.4_{\pm 3.4}$	$93.4_{\pm 0.4}$
	Ours	$\mathbf{93.7}_{\pm 0.4}$	$\mathbf{91.0}_{\pm 0.6}$	$92.9_{\pm 0.4}$

6.2. Hyperbolic Batch Normalization

Euclidean batch normalization (Ioffe & Szegedy, 2015) is one of the most widely used neural network operations that has, in many cases, obviated the need for explicit regularization such as dropout (Srivastava et al., 2014). In particular, analysis demonstrates that batch normalization induces a smoother loss surface which facilitates convergence and yields better final results (Santurkar et al., 2018). Generalizing this for Riemannian manifolds is a natural extension, and such a computation would involve a differentiable Fréchet mean.

6.2.1. Theoretical Formulation and Algorithm

In this section we formulate Riemannian Batch Normalization as a natural extension of standard Euclidean Batch Normalization. This concept is, to the best of our knowledge, only touched upon by Brooks et al. (2019) in the specific instance of the manifold of positive semidefinite matrices. However, we argue in Appendix G that, unlike our method, their formulation is incomplete and lacks sufficient generality to be considered a true extension.

Our full algorithm is given in Algorithm 2. Note that in practice we use $\sqrt{\sigma^2 + \epsilon}$ in place of σ as in the original

[9]The code for HGCN included only the Poincaré model implementation at the time this paper was submitted. Hence we use the Poincaré model for our experiments, although our contributions include derivations for both hyperboloid and Poincaré models.

Algorithm 2 Riemannian Batch Normalization

Training Input: Batches of data points $\{x_1^{(t)}, \cdots, x_m^{(t)}\} \subseteq \mathcal{M}$ for $t \in [1, \ldots, T]$, testing momentum $\eta \in [0, 1]$
Learned Paramters: Target mean $\mu' \in \mathcal{M}$, target variance $(\sigma')^2 \in \mathbb{R}$
Training Algorithm:
$\mu_{test} \leftarrow \text{FrechetMean}(\{x_1^{(1)}, \ldots, x_m^{(1)}\})$
$\sigma_{test} \leftarrow 0$
for $t = 1, \ldots, T$:
  $\mu = \text{FrechetMean}(\{x_1^{(t)}, \ldots, x_m^{(t)}\})$
  $\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} d(x_i^{(t)}, \mu)^2$
  $\mu_{test} = \text{FrechetMean}(\{\mu_{test}, \mu\}, \{\eta, 1-\eta\})$
  $\sigma_{test} = \frac{(t-1)\sigma_{test} + \sigma}{t}$
  for $i = 1, \cdots, m$:
    $\tilde{x_i}^{(t)} \leftarrow \exp_{\mu'}\left(\frac{\sigma'}{\sigma} PT_{\mu \to \mu'}(\log_{\mu} x_i^{(t)})\right)$
  return normalized batch $\tilde{x_1}^{(t)}, \cdots, \tilde{x_m}^{(t)}$

Testing Input: Test data points $\{\overline{x_1}, \cdots, \overline{x_s}\} \subseteq \mathcal{M}$, final running mean μ_{test} and running variance σ_{test}
Testing Algorithm:
$\overline{\mu} = \text{FrechetMean}(\{\overline{x_1}, \cdots, \overline{x_s}\})$
$\overline{\sigma}^2 = \frac{1}{m} \sum_{i=1}^{m} d(\overline{x_i}, \overline{\mu})^2$
for $i = 1, \cdots, s$:
  $\tilde{\overline{x_i}} \leftarrow \exp_{\mu_{test}}\left(\frac{\sigma_{test}}{\overline{\sigma}} PT_{\overline{\mu} \to \mu_{test}}(\log_{\overline{\mu}} \overline{x_i})\right)$
return normalized batch $\tilde{\overline{x_1}}, \cdots, \tilde{\overline{x_s}}$

formulation to avoid division by zero.

6.2.2. Experimental Results

We apply batch normalization (specifically for the Riemannian manifold of hyperbolic space) to the encoder network of the HNN (Ganea et al., 2018) baseline architecture in Chami et al. (2019) and run on the CoRA (Sen et al., 2008) dataset. We present the validation loss and ROC diagrams in Figure 2. Note that our method allows for both significantly faster initial convergence and a better final result. This translates to the test set, where our final ROC score is 93.0 as compared to the baseline 91.5. This is the training dynamic we expect to see when applying Euclidean batch normalization on Euclidean neural networks; hence, this indicates our generalization is successful for this test task.

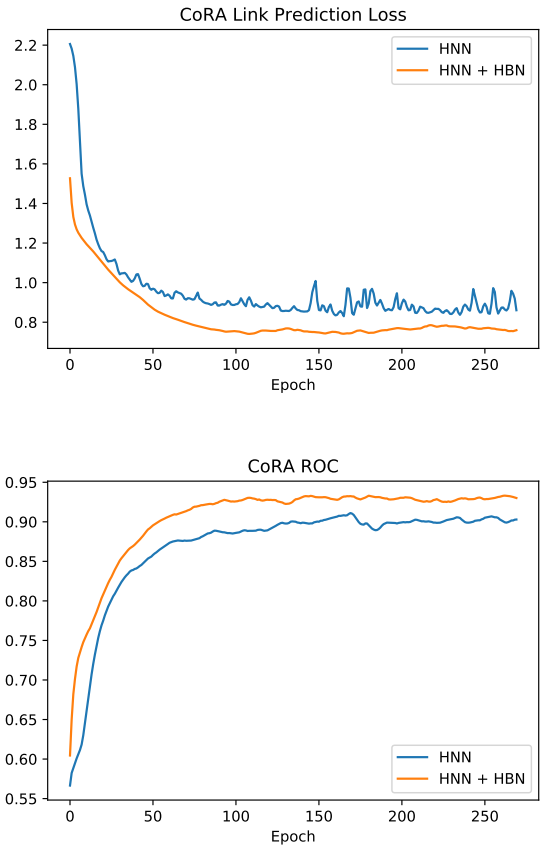

Figure 2. The graphs above correspond to a comparison of the HNN baseline, which uses a two-layer hyperbolic neural network encoder, and the baseline augmented with hyperbolic batch normalization after each layer. The top plot shows the comparison in terms of validation loss, and the bottom plot shows the comparison in terms of validation ROC AUC. Both figures show that we converge faster and attain better performance with respect to the relevant metric.

7. Conclusion and Future Work

We have presented a fully differentiable Fréchet mean operation for use in any differentiable programming setting. Concretely, we introduced differentiation theory for the general Riemannian case, and for the demonstrably useful case of hyperbolic space, we provided a fast forward pass algorithm and explicit derivative computations. We demonstrated that using the Fréchet mean in place of tangent space aggregation yields state-of-the-art performance on link prediction tasks in graphs with tree-like structure. Additionally, we extended batch normalization (a standard Euclidean operation) to the realm of hyperbolic space. On a graph link prediction test task, we showed that hyperbolic batch normalization gives benefits similar to those experienced in the Euclidean setting.

We hope our work paves the way for future developments in geometric representation learning. Potential future work can focus on speeding up our computation of the Fréchet mean gradient, finding applications of our theory on manifolds beyond hyperbolic space, and applying the Fréchet mean to generalize more standard neural network operations.

References

Adcock, A. B., Sullivan, B. D., and Mahoney, M. W. Tree-like structure in large social and information networks. *2013 IEEE 13th International Conference on Data Mining*, pp. 1–10, 2013.

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pp. 9558–9570, 2019.

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145, 2017.

Bacák, M. Computing medians and means in hadamard spaces. *SIAM Journal on Optimization*, 24(3):1542–1566, 2014.

Brooks, D., Schwander, O., Barbaresco, F., Schneider, J.-Y., and Cord, M. Riemannian batch normalization for spd neural networks. In *Advances in Neural Information Processing Systems*, pp. 15463–15474, 2019.

Casado, M. L. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems*, pp. 9154–9164, 2019.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4869–4880, 2019.

Charlier, B. Necessary and sufficient condition for the existence of a fréchet mean on the circle. *ESAIM: Probability and Statistics*, 17:635–649, 2013.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Fréchet, M. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pp. 215–310, 1948.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, pp. 5345–5355, 2018.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *ArXiv*, abs/1607.05447, 2016.

Gu, A., Sala, F., Gunel, B., and R, C. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019.

Gulcehre, C., 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 *International Conference on Learning Representations*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Karcher, H. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977.

Kleinberg, R. D. Geographic routing using hyperbolic space. *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 1902–1909, 2007.

Law, M., Liao, R., Snell, J., and Zemel, R. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pp. 3672–3681, 2019.

Lee, J. *Riemannian Manifolds: An Introduction to Curvature*. Graduate Texts in Mathematics. Springer New York, 1997.

Lee, J. M. Introduction to smooth manifolds. *Graduate Texts in Mathematics*, 218, 2003.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 8228–8239, 2019.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, 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. 3779–3788, 2018.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Pogančić, M. V., Paulus, A., Musil, V., Martius, G., and Rolinek, M. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.

Sala, F., Sa, C. D., Gu, A., and Ré, C. Representation tradeoffs for hyperbolic embeddings. *Proceedings of Machine Learning Research*, 80:4460–4469, 2018.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In *Proceedings of the 19th International Conference on Graph Drawing*, GD'11, pp. 355366, Berlin, Heidelberg, 2011. Springer-Verlag.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29:93–106, 2008.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Tifrea, A., Becigneul, G., and Ganea, O.-E. Poincaré glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2019.

Udrişte, C. *Convex functions and optimization methods on Riemannian manifolds*. Mathematics and Its Applications. Springer, Dordrecht, 1994. doi: 10.1007/978-94-015-8390-9.

Ungar, A. A. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wolter, F.-E. Distance function and cut loci on a complete riemannian manifold. *Archiv der Mathematik*, 32(1):92–96, 1979. doi: 10.1007/BF01238473.

Yu, T. and De Sa, C. M. Numerically accurate hyperbolic embeddings using tiling-based models. In *Advances in Neural Information Processing Systems*, pp. 2021–2031, 2019.

A. Proof of Correctness of Fréchet Mean as Generalization of Euclidean Mean

In this section, we show that the Fréchet mean and variance are natural generalizations of Euclidean mean and variance.

Proposition A.1. *On the manifold $\mathcal{M} = \mathbb{R}^n$, equations (3) and (4) are equivalent to the Euclidean mean and variance.*

Proof. Expanding the optimization function gives:

$$\begin{aligned}
\frac{1}{t}\sum_{l=1}^{t} d(x^{(l)},\mu)^2 &= \frac{1}{t}\sum_{l=1}^{t}\left\|\mu - x^{(l)}\right\|_2^2 = \frac{1}{t}\sum_{l=1}^{t}\sum_{i=1}^{n}(\mu_i - x_i^{(l)})^2 \\
&= \frac{1}{t}\sum_{l=1}^{t}\left[\sum_{i=1}^{n}\mu_i^2 - 2\sum_{i=1}^{n}\mu_i x_i^{(l)} + \sum_{i=1}^{n}(x_i^{(l)})^2\right] \\
&= \sum_{i=1}^{n}\mu_i^2 - \sum_{i=1}^{n}\frac{2}{t}\left(\sum_{l=1}^{t}x_i^{(l)}\right)\mu_i + \frac{1}{t}\sum_{i=1}^{n}\sum_{l=1}^{t}(x_i^{(l)})^2 \\
&= \sum_{i=1}^{n}\left(\mu_i - \left(\frac{1}{t}\sum_{l=1}^{t}x_i^{(l)}\right)\right)^2 + \sum_{i=1}^{n}\left(\frac{1}{t}\sum_{l=1}^{t}(x_i^{(l)})^2 - \left(\frac{1}{t}\sum_{l=1}^{t}x_i^{(l)}\right)^2\right)
\end{aligned}$$

Thus, optimizing the above quadratic function in μ gives (by a simple gradient computation):

$$\arg\min_{\mu\in\mathbb{R}^n}\frac{1}{t}\sum_{l=1}^{t} d(x^{(l)},\mu)^2 = \frac{1}{t}\sum_{l=1}^{t}x^{(l)}$$

$$\min_{\mu\in\mathbb{R}^n}\frac{1}{t}\sum_{l=1}^{t} d(x^{(l)},\mu)^2 = \sum_{i=1}^{n}\left(\frac{1}{t}\sum_{l=1}^{t}(x_i^{(l)})^2 - \left(\frac{1}{t}\sum_{l=1}^{t}x_i^{(l)}\right)^2\right)$$

We note that these are the mean and variance function in the standard Euclidean sense (where the total variance is the sum of variances on each coordinate). □

B. General Theorems on Differentiating through the Argmin

In this section, we provide generalizations of theorems in Gould et al. (2016) that will be useful in our gradient derivations when differentiating through the Fréchet mean.

We first consider the case of differentiating through an unconstrained optimization problem. This is a generalization of Lemma 3.2 in Gould et al. (2016). Note that again we use $\widetilde{\nabla}_{x^{(i)}}$ to represent the total derivative.

Theorem B.1. *Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be a twice differentiable function. Let $g : \mathbb{R}^n \to \mathbb{R}^m$ be given by $g(x) = \arg\min_{y\in\mathbb{R}^m} f(x,y)$. Then*

$$\widetilde{\nabla}_x g(x) = -\nabla_{yy}^2 f(x,g(x))^{-1}\widetilde{\nabla}_x\nabla_y f(x,g(x)) \tag{16}$$

Proof. From our definition of the optimization problem, we know that

$$\nabla_y f(x,y)\Big|_{y=g(x)} = 0$$

Taking the derivative with respect to x gives

$$\begin{aligned}
0 &= \widetilde{\nabla}_x\left(\nabla_y f(x,g(x))\right) \\
&= \widetilde{\nabla}_x\nabla_y f(x,g(x))\cdot\widetilde{\nabla}_x(x) + \nabla_{yy}^2 f(x,g(x))\cdot\widetilde{\nabla}_x g(x) \\
&= \widetilde{\nabla}_x\nabla_y f(x,g(x)) + \nabla_{yy}^2 f(x,g(x))\cdot\widetilde{\nabla}_x g(x)
\end{aligned}$$

and rearranging gives the desired result

$$\widetilde{\nabla}_x g(x) = -\nabla^2_{yy} f(x, g(x))^{-1} \widetilde{\nabla}_x \nabla_y f(x, g(x))$$

□

We then consider the case of differentiating through a constrained optimization problem. This is a generalization of Lemma 4.2 in Gould et al. (2016).

Theorem B.2. *Let $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ be continuous with second derivatives. Let $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ with $\operatorname{rank}(A) = m$. Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be defined by $g(x) = \underset{y\in\mathbb{R}^n : Ay=b}{\arg\min} f(x, y)$, and let $H = \nabla^2_{YY} f(x, g(x)) \in \mathbb{R}^{n\times n}$, then we have*

$$\widetilde{\nabla}_x g(x) = \left(H^{-1}A^\top (AH^{-1}A^\top)^{-1} AH^{-1} - H^{-1}\right) \nabla^2_{XY} f(x, g(x)) \tag{17}$$

where $\nabla^2_{XY} f(x, y) = \widetilde{\nabla}_x \nabla_y f(x, y)$, and $\nabla^2_{YY} f(x, y) = \nabla^2_{yy} f(x, y)$.

Proof. The proof is essentially the same as Gould's proof, and the only thing we need to be careful about is to ensure that the dimension of all quantities still make sense when we pass the partial derivative with respect to $x \in \mathbb{R}$ into the gradient with respect to $x \in \mathbb{R}^n$. We will carefully reproduce the steps in Gould's proof and make modifications if needed:

(1) **Formulating the Lagrangian**: The optimization problem that we are trying to solve is $g(x) = \underset{y\in\mathbb{R}^n : Ay=b}{\arg\min} f(x, y)$. We formulate its Lagrangian to be $L(x, y, \lambda) = f(x, y) + \lambda^\top (Ay - b)$.

Let $\tilde{g}(x) = (y^*(x), \lambda^*(x))$ be the optimal primal-dual pair, and write $\widetilde{\nabla}_x \tilde{g}(x) = (\widetilde{\nabla}_x y^*(x), \nabla_x \lambda^*(x)) = (\tilde{g}_Y(x), \tilde{g}_\Lambda(x))$. Note that we have $\tilde{g}_Y(x) \in \mathbb{R}^{n\times n}$ and $\tilde{g}_\Lambda(x) \in \mathbb{R}^{m\times n}$.

(2) **Derivative conditions from the Lagrangian**: From choice of optimal points $(y^*(x), \lambda^*(x))$, we have

$$\begin{cases} \nabla_y L(x, y, \lambda) = 0 \\ \nabla_\lambda L(x, y, \lambda) = 0 \end{cases} \Rightarrow \begin{cases} \nabla_Y f(x, y^*(x)) + A^\top \lambda^*(x) = 0 \\ Ay^*(x) - b = 0 \end{cases}$$

We note that the first equation has both sides in $\mathbb{R}^n$, and the second equation has both sides in $\mathbb{R}^m$.

Now we take the Jacobian $\widetilde{\nabla}_x$ for both equations[10]. For the first equation, applying the chain rule will result in

$$\begin{aligned} 0 &= \widetilde{\nabla}_x \left(\nabla_Y f(x, y^*(x)) + A^\top \lambda^*(x)\right) \\ &= \nabla^2_{XY} f(x, y^*(x)) \cdot \widetilde{\nabla}_x(x) + \nabla^2_{YY} f(x, y^*(x)) \cdot \widetilde{\nabla}_x(y^*(x)) + A^\top \widetilde{\nabla}_x(\lambda^*(x)) \\ &= \nabla^2_{XY} f(x, y^*(x)) + \nabla^2_{YY} f(x, y^*(x)) \cdot \tilde{g}_Y(x) + A^\top \tilde{g}_\Lambda(x) \end{aligned}$$

For the second equation, this will result in

$$0 = \widetilde{\nabla}_x \left(Ay^*(x) - b\right) = A \cdot \widetilde{\nabla}_x(y^*(x)) = A\tilde{g}_Y(x)$$

The above two equations give the following system:

$$\begin{cases} \nabla^2_{XY} f(x, g(x)) + \nabla^2_{YY} f(x, g(x)) \cdot \tilde{g}_Y(x) + A^\top \tilde{g}_\Lambda(x) = 0 \\ A\tilde{g}_Y(x) = 0 \end{cases}$$

where the first equation has both sides in $\mathbb{R}^{n\times n}$, and the second equation has both sides in $\mathbb{R}^{m\times n}$.

[10] Note that ∇_x (taking gradient over variable x) is different from ∇_X (taking the gradient over the first variable of the function).

(3) **Computing the Jacobian matrix**: Now we will solve for $\tilde{g}_Y(x)$ based on the above equations. We will denote $H = \nabla^2_{YY} f(x, g(x))$.

We first solve for $\tilde{g}_Y(x)$ in the first equation:

$$\tilde{g}_Y(x) = -H^{-1}\left(\nabla^2_{XY} f(x, g(x)) + A^\top \tilde{g}_\Lambda(x)\right)$$

We then substitute this value in the second equation to get

$$0 = A\left(-H^{-1}\left(\nabla^2_{XY} f(x, g(x)) + A^\top \tilde{g}_\Lambda(x)\right)\right) = -AH^{-1}\nabla^2_{XY} f(x, g(x)) - AH^{-1}A^\top \tilde{g}_\Lambda(x)$$

So we can solve for $\tilde{g}_\Lambda(x)$:

$$\tilde{g}_\Lambda(x) = -(AH^{-1}A^\top)^{-1}AH^{-1}\nabla^2_{XY} f(x, g(x))$$

We finally plug this into the first equation again:

$$\begin{aligned}
\tilde{g}_Y(x) &= -H^{-1}\left(\nabla^2_{XY} f(x, g(x)) + A^\top\left(-(AH^{-1}A^\top)^{-1}AH^{-1}\nabla^2_{XY} f(x, g(x))\right)\right) \\
&= -H^{-1}\nabla^2_{XY} f(x, g(x)) + H^{-1}A^\top(AH^{-1}A^\top)^{-1}AH^{-1}\nabla^2_{XY} f(x, g(x)) \\
&= \left(H^{-1}A^\top(AH^{-1}A^\top)^{-1}AH^{-1} - H^{-1}\right)\nabla^2_{XY} f(x, g(x))
\end{aligned}$$

From our definition of $\tilde{g}_Y$, we know that $\widetilde{\nabla}_x y^*(x) = \tilde{g}_Y(x)$, where $y^*(x) = g(x)$ is the optimal solution for the original optimization problem. Thus, we have

$$\widetilde{\nabla}_x g(x) = \left(H^{-1}A^\top(AH^{-1}A^\top)^{-1}AH^{-1} - H^{-1}\right)\nabla^2_{XY} f(x, g(x))$$

which is what we want to show. □

C. In-depth Differential Geometry Background

In this section we give a more formal introduction to differential geometry, which will be critical in the proof of and understanding of our differentiation theorem. Most of these definitions originate from (Lee, 2003) or (Lee, 1997).

C.1. Additional Manifold Definitions

Manifolds: An n-dimensional manifold $\mathcal{M}$ is a second countable Hausdorff space that is locally homeomorphic to $\mathbb{R}^n$. On open subsets of $\mathcal{M}$, we define a coordinate chart (U, φ) where $\varphi : U \to \tilde{U} \subseteq \mathbb{R}^n$ is the homeomorphism.

Smooth manifolds: A manifold $\mathcal{M}$ with dimension n is smooth if it has a smooth atlas, i.e. a collection of charts (U, φ) such that for any two charts (U, φ) and (V, ψ), either $\psi \circ \varphi^{-1}$ is a diffeomorphism or $U \cap V = \emptyset$.

Tangent spaces: A tangent vector v at a point $p \in \mathcal{M}$ is a linear map $v : C^\infty(M) \to \mathbb{R}$ which satisfies $v(fg) = f(p)vg + g(p)vf$. This linear map is also commonly called a derivation at p. The tangent space T_pM is the collection of these derivations. The tangent space is isomorphic to $\mathbb{R}^n$ as a vector space and there exist bases $\frac{\partial}{\partial x^i}|_p$.

Coordinate systems: Instead of writing out charts (U, φ), we use local coordinates (x^i) (which constitute maps from $U \to \mathbb{R}^n$ but provide a cleaner notation). We associate local coordinates x^i with their induced tangent vectors $\frac{\partial}{\partial x^i}|_p$, which form a basis for the tangent space.

Pushforward: A smooth map between manifolds $F : \mathcal{M} \to \mathcal{N}$ admits a push-forward $F_* : T_p\mathcal{M} \to T_{F(p)}\mathcal{N}$ given by $(F_* v)(f) = v(f \circ F)$. When the bases for $T_p\mathcal{M}$ and $T_{f(p)}\mathcal{N}$ are constructed by local coordinates, we can represent F by $\tilde{F}$, which computes in these coordinates. Then F_* corresponds to $\widetilde{\nabla}\tilde{F}$. Note that since the pushfoward is defined without respect to local coordinates, when given local coordinates, we oftentimes use the above two formulations interchangeably as $\widetilde{\nabla}F$.

Differential: The exterior derivative of a function $f : \mathcal{M} \to \mathbb{R}$ in terms of local coordinates $(x_1, \ldots, x_n)$ is given by $df = \begin{bmatrix} \partial_1 f \\ \vdots \\ \partial_n f \end{bmatrix}$ where $\partial_i f = \frac{\partial f}{\partial x^i}$. This can be generalized to differential forms (in this case we only consider 0-forms), but that is outside the scope of our paper. We will oftentimes just write this as ∇f to stress the matrix value version.

C.2. Additional Riemannian Geometry Definitions

Riemannian gradient: On a Riemannian manifold $(\mathcal{M}, \rho)$, the Riemannian gradient ∇^r of a function $f : \mathcal{M} \to \mathbb{R}$ is defined as $\nabla^r f = \rho^{-1} \circ \nabla f$, where ρ^{-1} is taken as a matrix and the gradient is taken with respect to local coordinates.

Geodesics: Geodesics are formally given as curves which have zero acceleration w.r.t. the Levi-Civita Connection. This material is outside the scope of this paper, but it is important to note that geodesics can be non-minimizing.

Conjugate points: Conjugate points (p, q) can be thought of as places where the geodesic between p and q is non-minimizing. The more formal definition involves Jacobi fields, where conjugate points are points where the nontrivial Joacbia field vanishes. However, this is outside the scope of this paper.

Hadamard manifolds: A Hadamard manifold is a manifold with everywhere non-positive sectional curvature. This space enables convex analysis/optimization since it is topologically trivial (similar to $\mathbb{R}^n$). Some canonical examples of Hadamard manifolds include Euclidean space $\mathbb{R}^n$, hyperbolic space $\mathbb{H}^n$, and the space of symmetric positive definite (SPD) matrices $\mathbb{S}^n_+$. A Hadamard manifold is geodesically complete and has no conjugate points.

C.3. A Few Useful Lemmas

Lemma C.1. *Around each point $x \in \mathcal{M}$, $\exp_x$ is a local diffeomorphism.*

Proof. This is given in Lemma 5.10 of (Lee, 1997) as a consequence of the inverse function theorem. □

Lemma C.2 (Chain Rule)**.** *Suppose $g : \mathcal{M} \to \mathcal{N}$ and $f : \mathcal{N} \to \mathcal{L}$ be smooth maps between manifolds. The $\widetilde{\nabla}(f \circ g)(x) = \widetilde{\nabla} f(g(x)) \circ \widetilde{\nabla} g(x)$.*

Proof. This follows directly from application of the standard chain rule. □

Corollary C.3. *We note that if $g : \mathcal{M} \to \mathcal{N}$ and $f : \mathcal{N} \to \mathbb{R}$ be smooth maps between Riemannian manifolds, with $\mathcal{M}$ having metric ρ, then*

$$\nabla^r_x (f \circ g)(x) = \rho_x^{-1} \circ (\widetilde{\nabla} g(x))^\top \circ \nabla f(g(x)) \tag{18}$$

Proof. This is a direct consequence of Lemma C.2 above and definition of Riemannian gradient. □

D. Differentiating Argmin on Manifolds

We now extend the results in Theorem B.1 to product of Riemannian manifolds.

Theorem D.1. *Let $(\mathcal{M}, \rho)$ be an m-dimensional Riemannian manifold, $(\mathcal{N}, \phi)$ be an n-dimensional Riemannian manifold, and $f : \mathcal{M} \times \mathcal{N} \to \mathbb{R}$ be twice differentiable. Let $g(x) = \arg\min_{y \in \mathcal{N}} f(x, y)$. With respect to local coordinates on $\mathcal{M}$ and $\mathcal{N}$, we have*

$$\widetilde{\nabla} g(x) = -\nabla_{yy} f(x, g(x))^{-1} \circ \widetilde{\nabla}_x \nabla_y f(x, g(x)) \tag{19}$$

In particular, if $\mathcal{L} : \mathcal{M} \to \mathbb{R}$ is some differentiable function, then we can calculate the Riemannian gradient as

$$\nabla^r_x (\mathcal{L} \circ g)(x) = -\rho_x^{-1} \circ (\widetilde{\nabla}_x \nabla_y f(x, g(x)))^\top \circ \nabla_{yy} f(x, g(x))^{-1} \circ \nabla \mathcal{L}(g(x)) \tag{20}$$

Proof. (1) **We first show the validity of equation (19).** Note that this is the same expression as equation (16), so all that remains to be proven is that we can apply Theorem B.1.

The proof for Theorem B.1 only depends on the gradient of f being 0. We know that $g(x)$ is a global minimum, so for any chart on $\mathcal{N}$ around $g(x)$, $g(x)$ is a local minimum. Therefore, we can take local coordinates and apply Theorem B.1 on these coordinates to obtain our result.

(2) **We then show the validity of equation (20) from the definition of Riemannian gradient.** This follows immediately from the definition of Riemannian gradient:

$$\begin{aligned}\nabla^r_x(\mathcal{L}\circ g)(x) &= \rho_x^{-1}\circ\widetilde{\nabla}_x g(x)^\top\circ\nabla\mathcal{L}(g(x))\\ &= \rho_x^{-1}\circ(-\nabla_{yy}f(x,g(x))^{-1}\circ\widetilde{\nabla}_x\nabla_y f(x,g(x)))^\top\circ\nabla\mathcal{L}(g(x))\\ &= -\rho_x^{-1}\circ(\widetilde{\nabla}_x\nabla_y f(x,g(x)))^\top\circ(\nabla_{yy}f(x,g(x))^{-1})^\top\circ\nabla\mathcal{L}(g(x))\\ &= -\rho_x^{-1}\circ(\widetilde{\nabla}_x\nabla_y f(x,g(x)))^\top\circ\nabla_{yy}f(x,g(x))^{-1}\circ\nabla\mathcal{L}(g(x))\end{aligned}$$

where we note that $\nabla_{yy}f(x,g(x))^{-1}$ is symmetric (since the Jacobian is symmetric). This is the desired proof. □

Remark. *While the above theorem is sufficient for establishing the existence of a local computation of gradient of an argmin on a manifold, it is not conducive for actual computation due to difficulty in obtaining a local coordinate chart. For practical purposes, we present two more computationally tractable versions below, one of which relies on an exponential map reparameterization inspired by Casado (2019) and one for when the manifold exists in ambient Euclidean space.*

Theorem D.2. *Let $(\mathcal{M},\rho)$ be an m-dimensional Riemannian manifold, $(\mathcal{N},\phi)$ be an n-dimensional Riemannian manifold, and $f:\mathcal{M}\times\mathcal{N}\to\mathbb{R}$ be a twice differentiable function. Let $g(x)=\arg\min_{y\in\mathcal{N}}f(x,y)$. Suppose we identify each tangent space $T_x\mathcal{M}$ with $\mathbb{R}^m$ and $T_y\mathcal{N}$ with $\mathbb{R}^n$ for all $x\in\mathcal{M}, y\in\mathcal{N}$. Fix x',y' as inputs and let $y'=g(x')$. Construct $\widehat{f}:\mathbb{R}^m\times\mathbb{R}^n\to\mathbb{R}$ to be given by $\widehat{f}(x,y)=f(\exp_{x'}x,\exp_{y'}y)$. Then*

$$\widetilde{\nabla}g(x') = -\widetilde{\nabla}\exp_{y'}(\mathbf{0})\circ\nabla^2_{yy}\widehat{f}(\mathbf{0},\mathbf{0})^{-1}\circ\widetilde{\nabla}_x\nabla_y\widehat{f}(\mathbf{0},\mathbf{0})\circ\widetilde{\nabla}\log_{x'}(x') \tag{21}$$

where $\log_{x'}$ is a local inverse of $\exp_{x'}$ around $\mathbf{0}$.

Proof. Define $\widehat{g}(x)=\arg\min_{y\in\mathbb{R}^n}\widehat{f}(x,y)$. Then we see that, locally, $g=\exp_{y'}\circ\widehat{g}\circ\log_{x'}$. Applying chain rule gives us

$$\widetilde{\nabla}g(x') = \widetilde{\nabla}\exp_{y'}(\mathbf{0})\circ\widetilde{\nabla}\widehat{g}(\mathbf{0})\circ\widetilde{\nabla}\log_{x'}(x')$$

where we plugged in $\log_{x'}x'=\mathbf{0}$. We can apply Theorem B.1 because $\widehat{g}:\mathbb{R}^m\to\mathbb{R}^n$ and since x' is an argmin then 0 is a local argmin. We substitute

$$\widetilde{\nabla}\widehat{g}(\mathbf{0}) = -\nabla^2_{yy}\widehat{f}(\mathbf{0},\mathbf{0})^{-1}\circ\widetilde{\nabla}_x\nabla_y\widehat{f}(\mathbf{0},\mathbf{0})$$

gives us the desired result. □

Theorem D.3. *Let $(\mathcal{M},\rho)$ be an m-dimensional Riemannian manifold, and $(\mathcal{N},\phi)$ be an n-dimensional Riemannian manifold. Suppose $\mathcal{M}$ is embedded in $\mathbb{R}^M$ and $\mathcal{N}$ is embedded in $\mathbb{R}^N$ where $M>m$ and $N>n$. Let $f:\mathcal{M}\times\mathcal{N}\to\mathbb{R}$ be a twice differentiable function and $g(x)=\arg\min_{y\in\mathcal{N}}f(x,y)$. We have*

$$\widetilde{\nabla}g(x) = -\operatorname{proj}_{T_x\mathcal{M}}\left(\left(\widetilde{\nabla}^{euc}_y\circ\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x))\right)^{-1}\left(\widetilde{\nabla}^{euc}_x\circ\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x))\right)\right) \tag{22}$$

where $\widetilde{\nabla}^{euc}$ is the total derivative w.r.t. the ambient (Euclidean) space.

Proof. **We will reproduce the steps in the proof of Theorem B.1 for this general setting.** Note that

$$\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x)) = 0$$

as the gradient in the tangent space $T_{g(x)}\mathcal{N}$ is 0. By taking the total derivative we know that

$$\begin{aligned}0 &= \widetilde{\nabla}^{euc}_x\left(\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x))\right)\\ &= \widetilde{\nabla}^{euc}_x\circ\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x))\cdot\widetilde{\nabla}^{euc}_x(x) + \widetilde{\nabla}^{euc}_y\circ\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x))\cdot\widetilde{\nabla}^{euc}_x g(x)\\ &= \widetilde{\nabla}^{euc}_x\circ\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x)) + \widetilde{\nabla}^{euc}_y\circ\operatorname{proj}_{T_{g(x)}\mathcal{N}}\circ\nabla^{euc}_y f(x,g(x))\cdot\widetilde{\nabla}^{euc}_x g(x)\end{aligned}$$

Rearranging gives us

$$\widetilde{\nabla}_x^{euc} g(x) = -\left(\widetilde{\nabla}_y^{euc} \circ \mathrm{proj}_{T_{g(x)}\mathcal{N}} \circ \nabla_y^{euc} f(x, g(x))\right)^{-1} \left(\widetilde{\nabla}_x^{euc} \circ \mathrm{proj}_{T_{g(x)}\mathcal{N}} \circ \nabla_y^{euc} f(x, g(x))\right)$$

and to obtain the derivative in the tangent space, we simply project to $T_{g(x)}\mathcal{M}$. □

Remark (Differentiating through the Fréchet Mean)**.** *Here we examine the formulas given in Theorems 4.1 and 4.2. The Fréchet mean objective function is not differentiable in general, as this requires the manifold to be diffeomorphic to $\mathbb{R}^n$ (Wolter, 1979). In the main paper, this complex formulation is sidestepped (as we define geodesics to be the unique curves that minimize length). Here, we discuss the problem of conjugate points and potential difficulties.*

We note that a sufficient condition for differentiability of the Fréchet mean is that its objective function is smooth, which occurs precisely when the squared Riemannian distance $d(x, y)^2$ is smooth. The conditions for this are that the manifold is geodesically complete and x and y are not conjugate points. This means that on Hadamard spaces, we can differentiate the Fréchet mean everywhere. Similarly, in cases when the manifold is not geodesically complete, such as when the manifold is not connected, we can still differentiate in places where the Fréchet mean is well defined. Finally, in the canonical case of spherical geometry, the set of conjugate points is a set of measure 0, meaning we can differentiate almost everywhere. Hence we see that in most practical settings the difficulties of non-differentiability do not arise.

E. Derivations of Algorithms to Compute the Fréchet Mean in Hyperbolic Space

E.1. Derivation of Algorithm for Hyperboloid Model Fréchet Mean Computation

In this section, we derive an algorithm that is guaranteed to converge to the Fréchet mean of points in the hyperboloid model.

Algorithm 3 Hyperboloid model Fréchet mean algorithm

Inputs: Data points $x^{(1)}, \cdots, x^{(t)} \in \mathbb{H}^n_K \subseteq \mathbb{R}^{n+1}$, weights $w_1, \cdots, w_t \in \mathbb{R}$.
Algorithm:
$y_0 = x^{(1)}$
for $k = 0, 1, \cdots, T$:

$$u_{k+1} = \sum_{l=1}^{t} \left(w_l \cdot \frac{2 \operatorname{arccosh}\left(-(x^{(l)})^\top M y_k\right)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \cdot x^{(l)} \right)$$

$$y_{k+1} = \frac{u_{k+1}}{\sqrt{-u_{k+1}^\top M u_{k+1}}}$$

return y_T

We now prove that this algorithm indeed converges to the Fréchet mean for points in the Lorentz model.

Theorem E.1. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{H}^n_K \subseteq \mathbb{R}^{n+1}$ be t points in the hyperboloid space, $w_1, \ldots, w_t \in \mathbb{R}^+$ be their weights, and let their weighted Fréchet mean be the solution to the following optimization problem*

$$y^* = \arg\min_{y \in \mathbb{H}^n} \sum_{l=1}^{t} w_l \cdot d_{\mathbb{H}^n}(x^{(l)}, y)^2 = \arg\min_{y \in \mathbb{H}^n} \sum_{l=1}^{t} w_l \cdot \operatorname{arccosh}^2(-(x^{(l)})^\top M y) \tag{23}$$

Then Algorithm 3 gives a sequence of points $\{y_k\}$ such that their limit $\lim_{k \to \infty} y_k = y^*$ converges to the Fréchet mean solution.

Proof. (1) **Apply concavity of** $\operatorname{arccosh}$ **to give an upper bound on the objective function**: Since $f(x) = \operatorname{arccosh}^2(x)$ is concave, its graph lies above the tangent, so we have

$$\begin{aligned} \operatorname{arccosh}^2(x) &\leq \operatorname{arccosh}^2(y) + (\operatorname{arccosh}^2(y))'(x - y) \\ &= \operatorname{arccosh}^2(y) + (x - y)\frac{2 \operatorname{arccosh}(y)}{\sqrt{y^2 - 1}} \end{aligned}$$

Let us denote

$$g(y) = \sum_{l=1}^{t} w_l \cdot \operatorname{arccosh}^2(-x^\top M y)$$

to be the objective function. Applying our concavity property to this objective function with respect to some fixed y_k at iteration k gives

$$\begin{aligned} g(y) &= \sum_{l=1}^{t} w_l \cdot \operatorname{arccosh}^2(-(x^{(l)})^\top M y) \\ &\leq \sum_{l=1}^{t} w_l \cdot \left(\operatorname{arccosh}^2\left(-(x^{(l)})^\top M y_k\right) + \left((-(x^{(l)})^\top M y) - (-(x^{(l)})^\top M y_k)\right) \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \right) \\ &= g(y_k) + \sum_{l=1}^{t} w_l \cdot (x^{(l)})^\top M (y_k - y) \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \end{aligned}$$

(2) **Finding solution to the minimization problem of upper bound**: Now consider the following minimization problem:

$$y_{k+1}^* = \underset{y \in \mathbb{H}^n}{\arg\min} \left(g(y_k) + \sum_{l=1}^{t} w_l \cdot (x^{(l)})^\top M (y_k - y) \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \right) \tag{24}$$

We will show that the solution of this optimization problem satisfies the computation in the algorithm:

$$y_{k+1}^* = \frac{u_{k+1}}{\sqrt{-u_{k+1}^\top M u_{k+1}}} \text{ with } u_{k+1} = \sum_{l=1}^{t} \left(w_l \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \cdot x^{(l)} \right)$$

(2-1) We first note that we can remove the terms that don't depend on y in the optimization problem:

$$\begin{aligned} &\underset{y \in \mathbb{H}^n}{\arg\min} \left(g(y_k) + \sum_{l=1}^{t} \left(w_l \cdot (x^{(l)})^\top M (y_k - y) \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \right) \right) \\ &= \underset{y \in \mathbb{H}^n}{\arg\min} \left(-y^\top M \cdot \sum_{l=1}^{t} \left(w_l \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \cdot x^{(l)} \right) \right) \end{aligned} \tag{25}$$

(2-2) We now propose a general method of solving optimization problems in the form (2-1).

For any u such that $u^\top M u < 0$, we have

$$\begin{aligned} \underset{y \in \mathbb{H}^n}{\arg\min} \left(-y^\top M u\right) &= \underset{y \in \mathbb{H}^n}{\arg\min} \frac{-y^\top M u}{\sqrt{-u^\top M u}} = \underset{y \in \mathbb{H}^n}{\arg\min} \left(\operatorname{arccosh}\left(-y^\top M \frac{u}{\sqrt{-u^\top M u}} \right) \right) \\ &= \underset{y \in \mathbb{H}^n}{\arg\min} \left(d_{\mathbb{H}^n}\left(y, \frac{u}{\sqrt{-u^\top M u}} \right) \right) = \frac{u}{\sqrt{-u^\top M u}} \end{aligned}$$

(2-3) Specifically in the k-th iteration, we can define

$$u_{k+1} = \sum_{l=1}^{t} \left(w_l \cdot \frac{2\operatorname{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \cdot x^{(l)} \right) \tag{26}$$

In order to apply the statement in (2-2), we now show that u_{k+1} satisfies $u_{k+1}^T M u_{k+1} < 0$. We can expand this as a sum:

$$u_{k+1}^T M u_{k+1} = \sum_{l=1}^{t}\sum_{j=1}^{t}\left(w_l \cdot \frac{2\,\mathrm{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}}\right)\left(w_j \cdot \frac{2\,\mathrm{arccosh}(-(x^{(j)})^\top M y_k)}{\sqrt{\left(-(x^{(j)})^\top M y_k\right)^2 - 1}}\right)(x^{(l)})^T M x^{(j)}$$

We know that the two constant blocks are both greater than 0. We also have $(x^{(l)})^T M x^{(l)} = -1$ strictly smaller than 0. Thus, we only need to show that $(x^{(l)})^T M x^{(j)} \leq 0$ for $l \neq j$.

We can expand

$$\begin{cases}(x^{(l)})^T M x^{(l)} = -1\\ (x^{(j)})^T M x^{(j)} = -1\end{cases} \Rightarrow \begin{cases}(x_0^{(l)})^2 = (x_1^{(l)})^2 + \cdots + (x_n^{(l)})^2 + 1\\ (x_0^{(j)})^2 = (x_1^{(j)})^2 + \cdots + (x_n^{(j)})^2 + 1\end{cases}$$

Multiplying them and applying Cauchy's inequality gives

$$(x_0^{(l)} x_0^{(j)})^2 = ((x_1^{(l)})^2 + \cdots + (x_n^{(l)})^2 + 1)((x_1^{(j)})^2 + \cdots + (x_n^{(j)})^2 + 1) \geq (|x_1^{(l)} x_1^{(j)}| + \cdots + |x_n^{(l)} x_n^{(j)}| + 1)^2$$

This implies

$$x_0^{(l)} x_0^{(j)} = |x_0^{(l)} x_0^{(j)}| \geq |x_1^{(l)} x_1^{(j)}| + \cdots + |x_n^{(l)} x_n^{(j)}| + 1 \geq x_1^{(l)} x_1^{(j)} + \cdots + x_n^{(l)} x_n^{(j)} + 1$$

if we assume $x_0^{(l)} > 0$ for all $1 \leq l \leq n$ (i.e. we always pick points on the same connected component of the hyperboloid, either with x_0 always positive or always negative). Thus, we have $(x^{(l)})^\top M x^{(j)} \leq -1 < 0$ for $l \neq j$ as well. This together our above results ensure that we have $u_{k+1}^\top M u_{k+1} < 0$.

(2-4) Since we have verified $u_{k+1}^\top M u_{k+1} < 0$ in (2-3), we know that we can apply the result in (2-2) to the optimization problem (25) to obtain

$$y_{k+1}^* = \underset{y \in \mathbb{H}^n}{\arg\min}\left(-y^\top M \cdot \sum_{l=1}^{t}\left(w_l \cdot \frac{2\,\mathrm{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \cdot x^{(l)}\right)\right) = \frac{u_{k+1}}{\sqrt{-u_{k+1}^\top M u_{k+1}}}$$

and this together with equation (26) gives exactly the same process as in the algorithm. Thus, the algorithm generates the solution $y_{k+1} = y_{k+1}^*$ of this minimization problem.

(3) **Proof of convergence**: We now show that the sequence $\{y_k\}$ converges to the Fréchet mean y^* as $k \to \infty$.

To show this, we consider the objective function minimized in (24):

$$y_{k+1}^* = \underset{y \in \mathbb{H}^n}{\arg\min}\, h(y)\text{ , where } h(y) = g(y_k) + \sum_{l=1}^{t} w_l \cdot (x^{(l)})^\top M(y_k - y) \cdot \frac{2\,\mathrm{arccosh}(-(x^{(l)})^\top M y_k)}{\sqrt{\left(-(x^{(l)})^\top M y_k\right)^2 - 1}} \tag{27}$$

We know that $h(y_k) = g(y_k)$ for $y_k \in \mathbb{H}^n$, so we must have $g(y_{k+1}^*) \leq g(y_k)$ with equality only if y_k is already the minimizer of g (in which case, by definition of g, we have already reached a Fréchet mean). Thus, if we are not at the Fréchet mean, we must have $g(y_{k+1}) < g(y_k)$ strictly decreasing.

This means the sequence $\{y_k\}$ must converge to the Fréchet mean, as this is the only fixed point that we can converge to.

□

E.2. Derivation of Algorithm for Poincaré Model Fréchet Mean Computation

In this section, we derive an algorithm that is guaranteed to converge to the Fréchet mean of points in the Poincaré ball model. This is the same algorithm as Algorithm 1 in the main paper.

Algorithm 4 Poincaré model Fréchet mean algorithm

Inputs: $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}_K^n \subseteq \mathbb{R}^{n+1}$.
Algorithm:
$y_0 = x^{(1)}$
Define $g(y) = \frac{2\,\mathrm{arccosh}(1+2y)}{\sqrt{y^2+y}}$
for $k = 0, 1, \cdots, T$:
  for $l = 1, 2, \cdots, t$:
    $\alpha_l = w_l \cdot g\left(\frac{\|x^{(l)}-y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \cdot \frac{1}{1-\|x^{(l)}\|^2}$
  $a = \sum_{l=1}^{t} \alpha_l$, $b = \sum_{l=1}^{t} \alpha_l x^{(l)}$, $c = \sum_{l=1}^{t} \alpha_l \|x^{(l)}\|^2$
  $y_{k+1} = \left(\frac{a+c-\sqrt{(a+c)^2-4\|b\|^2}}{2\|b\|^2}\right) b$
return y_T

We now prove that this algorithm indeed converges to the Fréchet mean for points in the Poincaré ball model.

Theorem E.2. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}_K^n$ be t points in the Poincaré ball, $w_1, \ldots, w_t \in \mathbb{R}^+$ be their weights, and let their weighted Fréchet mean be the solution to the following optimization problem*

$$y^* = \arg\min_{y \in \mathbb{D}^n} \sum_{l=1}^{t} w_l \cdot d_{\mathbb{D}^n}(x^{(l)}, y)^2 = \arg\min_{y \in \mathbb{D}^n} \sum_{l=1}^{t} w_l \cdot \mathrm{arccosh}^2\left(1 + 2 \cdot \frac{\|x^{(l)} - y\|^2}{(1-\|x^{(l)}\|^2)(1-\|y\|^2)}\right) \tag{28}$$

Then Algorithm 4 gives a sequence of points $\{y_k\}$ such that their limit $\lim_{k\to\infty} y_k = y^*$ converges to the Fréchet mean solution.

Proof. (1) **Apply concavity of** arccosh **to give an upper bound on the objective function**: Let us denote $g(y) = \mathrm{arccosh}(1+2y)^2$, and also denote $h(y)$ to be the objective function:

$$h(y) = \sum_{l=1}^{t} w_l \cdot g\left(\frac{\|x^{(l)} - y\|^2}{(1-\|x^{(l)}\|^2)(1-\|y\|^2)}\right)$$

We know that $g(y)$ is concave, and this means

$$g(x) \le g(y) + g'(y)(x - y)$$

Applying this to our objective function with respect to some fixed y_k at iteration k gives:

$$\begin{aligned} g\left(\frac{\|x^{(l)} - y\|^2}{(1-\|x^{(l)}\|^2)(1-\|y\|^2)}\right) &\le g\left(\frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \\ &+ g'\left(\frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \cdot \left(\frac{\|x^{(l)} - y\|^2}{(1-\|x^{(l)}\|^2)(1-\|y\|^2)} - \frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \end{aligned}$$

Summing this up over all $0 \le l \le t$ gives us

$$\begin{aligned} h(y) &\le h(y_k) \\ &+ \sum_{l=0}^{t} w_l \cdot g'\left(\frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \cdot \left(\frac{\|x^{(l)} - y\|^2}{(1-\|x^{(l)}\|^2)(1-\|y\|^2)} - \frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \end{aligned} \tag{29}$$

(2) **Finding solution to the minimization problem of upper bound**: Our goal is to solve the optimization problem of the RHS of equation (29). Following similar ideas as in Theorem E.1, we can remove terms unrelated to the variable y and simplify this optimization problem:

$$y_{k+1}^* = \arg\min_{y \in \mathbb{D}^n} \sum_{l=1}^{t} w_l \cdot g'\left(\frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \cdot \left(\frac{\|x^{(l)} - y\|^2}{(1-\|x^{(l)}\|^2)(1-\|y\|^2)}\right)$$

Let us denote

$$\alpha_l = w_l \cdot g'\left(\frac{\|x^{(l)} - y_k\|^2}{(1-\|x^{(l)}\|^2)(1-\|y_k\|^2)}\right) \cdot \frac{1}{1-\|x^{(l)}\|^2}$$

Then we can simplify the optimization problem to

$$\arg\min_{y\in\mathbb{D}^n}\left(\sum_{l=1}^{t}\alpha_l \cdot \frac{\|x^{(l)}-y\|^2}{1-\|y\|^2}\right) = \arg\min_{y\in\mathbb{D}^n}\left(\sum_{l=1}^{t}\alpha_l \cdot \frac{\|x^{(l)}\|^2 - 2(x^{(l)})^\top y + \|y\|^2}{1-\|y\|^2}\right)$$

Now let

$$a = \sum_{l=1}^{t}\alpha_l \;,\; b = \sum_{l=1}^{t}\alpha_l x^{(l)} \;,\; c = \sum_{l=1}^{t}\alpha_l \|x^{(l)}\|^2$$

Then the above optimization problem further simplifies to

$$\arg\min_{y\in\mathbb{D}^n}\left(\frac{a\|y\|^2 - 2b^T y + c}{1-\|y\|^2}\right)$$

Say we fix some length $\|y\|$, then to minimize the above expression, we must choose $y = \eta b$ (so that $b^\top y$ achieves maximum). Plugging this into the above expression gives

$$\arg\min_{\eta b\in\mathbb{D}^n}\left(\frac{a\|b\|^2\eta^2 - 2\|b\|^2\eta + c}{1-\|b\|^2\eta^2}\right)$$

Now we'll consider

$$f(\eta) = \frac{a\|b\|^2\eta^2 - 2\|b\|^2\eta + c}{1-\|b\|^2\eta^2}$$

$$\begin{aligned}
\Rightarrow f'(\eta) &= \frac{(1-\|b\|^2\eta^2)(a\|b\|^2\eta^2 - 2\|b\|^2\eta + c)' - (a\|b\|^2\eta^2 - 2\|b\|^2\eta + c)(1-\|b\|^2\eta^2)'}{(1-\|b\|^2\eta^2)^2} \\
&= \frac{(1-\|b\|^2\eta^2)(2a\|b\|^2\eta - 2\|b\|^2) - (a\|b\|^2\eta^2 - 2\|b\|^2\eta + c)(-2\|b\|^2\eta)}{(1-\|b\|^2\eta^2)^2} \\
&= \frac{-2\|b\|^2 + 2a\|b\|^2\eta + 2\|b\|^4\eta^2 - 2a\|b\|^4\eta^3 + 2\|b\|^2 c\eta - 4\|b\|^4\eta^2 + 2a\|b\|^4\eta^3}{(1-\|b\|^2\eta^2)^2} \\
&= \frac{-2\|b\|^2 + 2(a+c)\|b\|^2\eta - 2\|b\|^4\eta^2}{(1-\|b\|^2\eta^2)^2} = \frac{-2\|b\|^2}{(1-\|b\|^2\eta^2)^2}\left(\|b\|^2\eta^2 - (a+c)\eta + 1\right)
\end{aligned}$$

Thus, to achieve minimum, we will have $f'(\eta) = 0$, so

$$f'(\eta) = 0 \Rightarrow \|b\|^2\eta^2 - (a+c)\eta + 1 = 0 \Rightarrow \eta = \frac{a+c \pm \sqrt{(a+c)^2 - 4\|b\|^2}}{2\|b\|^2}$$

Moreover, we know that $f'(\eta) < 0$ for $\eta < \frac{a+c-\sqrt{(a+c)^2-4\|b\|^2}}{2\|b\|^2}$ and $f'(\eta) > 0$ for $\frac{a+c-\sqrt{(a+c)^2-4\|b\|^2}}{2\|b\|^2} < \eta < \frac{a+c+\sqrt{(a+c)^2-4\|b\|^2}}{2\|b\|^2}$. This means the actual η that achieves the minimum is

$$\eta = \frac{a+c-\sqrt{(a+c)^2-4\|b\|^2}}{2\|b\|^2}$$

and thus we have

$$y^*_{k+1} = \left(\frac{a+c-\sqrt{(a+c)^2-4\|b\|^2}}{2\|b\|^2}\right) b$$

and this is exactly what we computed for y_{k+1} in Algorithm 4.

(3) **Proof of convergence**: We now show that the sequence $\{y_k\}$ converges to the Fréchet mean y^* as $k \to \infty$.

We know that if we pick the minimizing y for the RHS of equation (29), the RHS is smaller than or equal to the case where we pick $y = y_k$ (in which case the RHS becomes $g(y_k)$).

Thus, we know that $g(y_{k+1}^*) \leq g(y_k)$. Similar to the case in Theorem E.1, we know that equality only holds when we're already at the Fréchet mean, so if we are not at the Fréchet mean, we must have $g(y_{k+1}) < g(y_k)$ strictly decreasing. This means the sequence $\{y_k\}$ must converge to the Fréchet mean, as this is the only fixed point that we can converge to.

□

F. Explicit Derivations of Backpropagation of Fréchet Mean for Hyperbolic Space

F.1. Differentiating through all Parameters of the Fréchet Mean

With our constructions from Appendix D, we derive gradient expressions for the Fréchet mean in hyperbolic space. In particular, we wish to differentiate with respect to input points, weights, and curvature. To see that these are all differentiable values, we note that the squared distance function in the Fréchet mean objective function admits derivatives for all of these variables. In particular, we see that the Fréchet mean for hyperbolic space w.r.t. these input parameters is effectively a smooth function from $(\mathbb{H}^n)^t \times \mathbb{R}^t \times \mathbb{R} \to \mathbb{H}^n$, where the input variables are points, weights, and curvature respectively.

F.2. Derivation of Formulas for Hyperboloid Model Backpropagation

In this section, we derive specific gradient computations for the Fréchet mean in hyperboloid model; we assume the Fréchet mean is already provided from the forward pass. This gradient computation is necessary since, at the time of writing, all machine learning auto-differentiation packages do not support manifold-aware higher-order differentiation.

Our first goal is to recast our original problem into an equaivalent optimization problem that can be easily differentiated.

Theorem F.1. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{H}^n_K \subseteq \mathbb{R}^{n+1}$ be t points in the hyperboloid space, $w_1, \ldots, w_t \in \mathbb{R}^+$ be their weights, and let their weighted Fréchet mean y^* be the solution to the following optimization problem.*

$$y^*(x^{(1)}, \cdots, x^{(t)}) = \underset{y \in \mathbb{H}^n}{\arg\min} \sum_{l=1}^{t} w_l \cdot \operatorname{arccosh}^2(-(x^{(l)})^\top M y) \tag{30}$$

Then we can recast this using the reparametrization map $y = h(\overline{x}) = u + \overline{x} \cdot \sqrt{1 + u^\top M u}$ to obtain the equivalent problem:

$$u^*(x^{(1)}, \cdots, x^{(t)}) = \underset{u \in \mathbb{R}^n, \bar{x}^\top M u = 0}{\arg\min} F(x^{(1)}, \cdots, x^{(t)}, u) \tag{31}$$

$$\text{where } F(x^{(1)}, \cdots, x^{(t)}, u) = \sum_{l=1}^{t} w_l \cdot g\left((x^{(l)})^\top M u + (x^{(l)})^\top M \bar{x} \cdot \sqrt{1 + u^\top M u}\right) \tag{32}$$

$$g(x) = \operatorname{arccosh}(-y)^2 \tag{33}$$

Proof. (1) **We first show that $h : \mathbb{H}^n \to \mathbb{H}^n$ is a bijection.**

For completeness, we derive the re-parameterization together with intuition. We let $\overline{x}$ be any point in $\mathbb{H}^n$ and u a point in the tangent space satisfying $\overline{x}^\top M u = 0$. We wish to solve for a constant c so that $h(\overline{x}) = u + \overline{x} \cdot c$ lies on the manifold. Note that this corresponds to re-scaling $\overline{x}$ so that the induced shift by the vector u does not carry $\overline{x}$ off the manifold. Algebraically, we require $h(\overline{x})^\top M h(\overline{x}) = -1$:

$$(u + \overline{x} \cdot c)^\top M (u + \overline{x} \cdot c) = -1$$
$$u^\top M u + u^\top M \overline{x} \cdot c + c \overline{x}^\top M u + c^2 \cdot \overline{x}^\top M \overline{x} = -1$$
$$u^\top M u + 1 = c^2 \Rightarrow c = \sqrt{u^\top M u + 1}$$

Note that since we are on the positive sheet of the hyperboloid, we take the positive root at the final step. Since the map is non-degenerate, i.e. $\sqrt{u^\top M u + 1} \neq 0$ since u is in the tangent space, observe that for any point $h(\overline{x})$ on the hyperboloid

we can solve for $\overline{x} = \frac{h(\overline{x})-u}{\sqrt{u^\top Mu+1}}$. Hence the map is surjective. Moreover, note that the value of $\overline{x}$ is unique; hence we have injectivity and conclude that h is a bijection.

(2) The rest of the proof follows from plugging the reparametrization map into equation (30) and simplifying the result. □

We then present how to backpropagate through this equivalent problem.

Theorem F.2. *For each $1 \leq i \leq n$, consider the function $u_i^* : \mathbb{R}^n \to \mathbb{R}^n$ defined by*

$$u_i^*(x^{(i)}) = u^*(\tilde{x}^{(1)}, \cdots, \tilde{x}^{(i-1)}, x^{(i)}, \tilde{x}^{(i+1)}, \cdots \tilde{x}^{(n)}) \tag{34}$$

which only varies $x^{(i)}$ and fixing all other input variables in u^*.

Then the Jacobian matrix of u_i^* can be computed in the following way:

$$\widetilde{\nabla}_{x^{(i)}} u^*(x^{(i)}) = \left(H^{-1}A^\top(AH^{-1}A^\top)^{-1}AH^{-1} - H^{-1}\right) \widetilde{\nabla}_{x^{(i)}} \nabla_u F(x^{(1)}, \cdots, x^{(t)}, u)\Big|_{u=0} \tag{35}$$

where we plug in $A = x^\top M$, the Hessian evaluated at $u = 0$:

$$\begin{aligned} H &= \nabla^2_{uu} F(x^{(1)}, \ldots, x^{(t)}; u)\Big|_{u=0} \\ &= \sum_{l=1}^{t} w_l \cdot \left(g''\left((x^{(l)})^\top M\bar{x}\right) \cdot \left(Mx^{(l)}\right)\left(Mx^{(l)}\right)^\top + g'\left((x^{(l)})^\top M\bar{x}\right) \cdot \left((x^{(l)})^\top M\bar{x}\right) \cdot M \right) \end{aligned} \tag{36}$$

and the mixed gradient evaluated at $u = 0$:

$$\widetilde{\nabla}_{x^{(i)}} \nabla_u F(x^{(1)}, \cdots, x^{(t)}, u)\Big|_{u=0} = w_i \cdot \left(g''\left((x^{(i)})^\top M\bar{x}\right) \cdot \left(Mx^{(i)}\right)(M\bar{x})^\top + g'\left((x^{(i)})^\top M\bar{x}\right) \cdot M \right) \tag{37}$$

Proof. (1) **Application of Gould's theorem**: Recall from Theorem F.1 that our minimization problem has the form

$$u^*(x^{(1)}, \cdots, x^{(t)}) = \underset{u \in \mathbb{R}^n, \bar{x}^\top Mu = 0}{\arg\min} F(x^{(1)}, \cdots, x^{(t)}, u)$$

where

$$F(x^{(1)}, \cdots, x^{(t)}, u) = \sum_{l=1}^{t} w_l \cdot g\left((x^{(l)})^\top Mu + (x^{(l)})^\top M\bar{x} \cdot \sqrt{1 + u^\top Mu}\right)$$

$$g(x) = \operatorname{arccosh}(-y)^2$$

We now apply Theorem B.2 on the constrained optimization problem (31), noting that we can write $x^\top Mx = 0$ in the form $Au = b$ for $A = x^\top M \in \mathbb{R}^{1\times n}$ and $b = 0 \in \mathbb{R}$. This gives us

$$\widetilde{\nabla} u^*(x^{(i)}) = \left(H^{-1}A^\top(AH^{-1}A^\top)^{-1}AH^{-1} - H^{-1}\right) \widetilde{\nabla}_{x^{(i)}} \nabla_u F(x^{(1)}, \cdots, x^{(t)}, u)$$

where $A = x^\top M$ and $H = \nabla^2_{uu} F(x^{(1)}, \cdots, x^{(t)}, u)$, so we recover equation (35). We also note that after we compute the Fréchet mean in the forward pass, we can set $u = 0$ and this will simplify the expressions of both partial derivatives.

(2) **Computing the gradient $\nabla_u F$**: We first compute $\nabla_u F(x^{(1)}, \cdots, x^{(n)}, u) \in \mathbb{R}^n$ using the chain rule.

$$\nabla_u F(x^{(1)}, \cdots, x^{(t)}; u) = \sum_{l=1}^{t} w_l \cdot g'\left((x^{(l)})^\top Mu + (x^{(l)})^\top M\bar{x} \cdot \sqrt{1 + u^\top Mu}\right) \cdot \left(Mx^{(l)} + (x^{(l)})^\top M\bar{x} \cdot \frac{Mu}{\sqrt{1 + u^\top Mu}}\right)$$

(3) **Computing the Hessian** $\nabla^2_{uu} F$: We then evaluate the Hessian $\nabla^2_{uu} F(x^{(1)}, \cdots, x^{(t)}, u) \in \mathbb{R}^{n \times n}$.

$$
\begin{aligned}
&H = \nabla^2_{uu} F(x^{(1)}, \ldots, x^{(t)}; u) \\
&= \sum_{l=1}^{t} w_l \cdot g''\left((x^{(l)})^\top M u + (x^{(l)})^\top M \bar{x} \cdot \sqrt{1 + u^\top M u}\right) \cdot \left(M x^{(l)} + (x^{(l)})^\top M \bar{x} \cdot \frac{M u}{\sqrt{1 + u^\top M u}}\right) \\
&\qquad \cdot \left(M x^{(l)} + (x^{(l)})^\top M \bar{x} \cdot \frac{M u}{\sqrt{1 + u^\top M u}}\right)^\top \\
&\quad + \sum_{l=1}^{t} w_l \cdot g'\left((x^{(l)})^\top M u + (x^{(l)})^\top M \bar{x} \cdot \sqrt{1 + u^\top M u}\right) \cdot (x^{(l)})^\top M \bar{x} \cdot \left(\frac{M}{\sqrt{1 + u^\top M u}} - \frac{M u u^\top M}{(1 + u^\top M u)^{3/2}}\right)
\end{aligned}
$$

(4) **Computing** $\widetilde{\nabla}_{x^{(i)}} \nabla_u F$: We then evaluate $\widetilde{\nabla}_{x^{(i)}} \nabla_u F(x^{(1)}, \cdots, x^{(t)}, u) \in \mathbb{R}^{n \times n}$.

We first note that we can rewrite $(x^{(l)})^\top M \bar{x} \cdot \frac{M u}{\sqrt{1 + u^\top M u}} = \frac{M u \bar{x}^\top M x^{(l)}}{1 + u^\top M u}$ where the numerator is a matrix multiplication. We then take the derivative of ∇_u computed above:

$$
\begin{aligned}
&\widetilde{\nabla}_{x^{(i)}} \nabla_u F(x^{(1)}, \cdots, x^{(t)}, u) \\
&= w_i \cdot g''\left((x^{(i)})^\top M u + (x^{(i)})^\top M \bar{x} \cdot \sqrt{1 + u^\top M u}\right) \cdot \left(M x^{(i)} + (x^{(i)})^\top M \bar{x} \cdot \frac{M u}{\sqrt{1 + u^\top M u}}\right) \left(M u + M \bar{x} \sqrt{1 + u^\top M u}\right)^\top \\
&\qquad + w_i \cdot g'\left((x^{(i)})^\top M u + (x^{(i)})^\top M \bar{x} \cdot \sqrt{1 + u^\top M u}\right) \cdot \left(M + \frac{M u \bar{x}^\top M}{\sqrt{1 + u^\top M u}}\right)
\end{aligned}
$$

(5) **Evaluating the above functions at** $u = 0$: In our above computations, the parameter u would be set to 0 after our forward pass finds the Fréchet mean. Thus, we will evaluate our results in (6), (7) at $u = 0$.

The Hessian evaluated at $u = 0$ gives

$$
\begin{aligned}
H &= \nabla^2_{uu} F(x^{(1)}, \ldots, x^{(t)}; u)\Big|_{u=0} \\
&= \sum_{l=1}^{t} w_l \cdot \left(g''\left((x^{(l)})^\top M \bar{x}\right) \cdot \left(M x^{(l)}\right)\left(M x^{(l)}\right)^\top + g'\left((x^{(l)})^\top M \bar{x}\right) \cdot \left((x^{(l)})^\top M \bar{x}\right) \cdot M\right)
\end{aligned}
$$

The mixed gradient evaluated at $u = 0$ gives

$$
\widetilde{\nabla}_{x^{(i)}} \nabla_u F(x^{(1)}, \cdots, x^{(t)}, u)\Big|_{u=0} = w_i \cdot \left(g''\left((x^{(i)})^\top M \bar{x}\right) \cdot \left(M x^{(i)}\right)(\bar{x})^\top + g'\left((x^{(i)})^\top M \bar{x}\right) \cdot M\right)
$$

and the above two equations give exactly equations (36) and (37). □

Remark. *In the above formulation we have assumed that $K = -1$ for simplicity. For different curvatures, the above process can be generalized by constructing a general reparametrization formula (based off of scaling the hyperboloid).*

Remark. *We also omit derivations for curvature and weights, as this process can be seen in the following section.*

F.3. Derivation of Formulas for Poincaré Ball Model Backpropagation

In this section, we derive specific gradient computations for the Poincaré ball. This is necessary since many machine learning auto-differentiation do not support higher order differentiation.

Theorem F.3. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}^n_K \subseteq \mathbb{R}^n$ be t points in the Poincaré ball, $w_1, \ldots, w_t \in \mathbb{R}$ be their weights, and let their weighted Fréchet mean y^* be the solution to the following optimization problem.*

$$
y^*(x^{(1)}, \cdots, x^{(t)}) = \arg\min_{y \in \mathbb{D}^n_K} f(\{x\}, y) \tag{38}
$$

$$\textit{where } f(\{x\},y)=\frac{1}{K}\sum_{l=1}^{t}w_l\cdot\operatorname{arccosh}\left(1-\frac{2K||x^{(l)}-y||_2^2}{(1+K||x^{(l)}||_2^2)(1+K||y||_2^2)}\right)^2 \tag{39}$$

Then the gradient of y^* with respect to $x^{(i)}$ is given by

$$\widetilde{\nabla}_{x^{(i)}}y^*(\{x\})=-\nabla_{yy}^2 f(\{x\},y^*(\{x\}))^{-1}\widetilde{\nabla}_{x^{(i)}}\nabla_y f(\{x\},y^*(\{x\})) \tag{40}$$

(1) Terms of $\widetilde{\nabla}_{x^{(i)}}\nabla_y f(\{x\},y^*(\{x\}))$ are given by

$$\frac{\partial}{\partial x_{ik}}\frac{\partial}{\partial y_j}f(\{x\},y^*(\{x\}))=\frac{1}{K}w_i\cdot\left(\left(\frac{2}{(v^{(i)})^2-1}-\frac{2(v^{(i)})\operatorname{arccosh}(v^{(i)})}{((v^{(i)})^2-1)^{\frac{3}{2}}}\right)M_{ik}T_{ij}+\frac{2\operatorname{arccosh}(v^{(i)})}{\sqrt{(v^{(i)})^2-1}}\frac{\partial T_{ij}}{\partial x_{ik}}\right) \tag{41}$$

where

$$T_{ij}=\frac{4K}{D^{(i)}}(x_j^{(i)}-y_j)+\frac{4K^2}{(D^{(i)})^2}y_j\|x^{(i)}-y\|_2^2\cdot(1+K\|x^{(i)}\|_2^2) \tag{42}$$

$$M_{ik}=\frac{4K}{D^{(i)}}(y_k-x_k^{(i)})+\frac{4K^2}{(D^{(i)})^2}\cdot x_k^{(i)}\|x^{(i)}-y\|_2^2\cdot(1+K\|y\|_2^2) \tag{43}$$

$$\frac{\partial T_{ij}}{\partial x_{ij}}=\frac{4K}{D^{(i)}}-\frac{8K^2}{(D^{(i)})^2}x_j^{(i)}(x_j^{(i)}-y_j)(1+K\|y\|_2^2)+\frac{8K^2y_j}{(D^{(i)})^2}(x_j^{(i)}-y_j)(1+K\|x^{(i)}\|_2^2)-\frac{8K^3y_j}{(D^{(i)})^2}x_j^{(i)}\|x^{(i)}-y\|_2^2 \tag{44}$$

and for $k\neq j$,

$$\frac{\partial T_{ij}}{\partial x_{ik}}=-\frac{8K^2}{(D^{(i)})^2}x_k^{(i)}(x_j^{(i)}-y_j)(1+K\|y\|_2^2)+\frac{8K^2y_j}{(D^{(i)})^2}(x_k^{(i)}-y_k)(1+K\|x^{(i)}\|_2^2)-\frac{8K^3y_j}{(D^{(i)})^2}x_k^{(i)}\|x^{(i)}-y\|_2^2 \tag{45}$$

(2) Terms of $\nabla_{yy}^2 f(\{x\},y^*(\{x\}))$ is given by

$$\frac{\partial}{\partial y_i}\frac{\partial}{\partial y_j}f(\{x\},y^*(\{x\}))=\frac{1}{K}\sum_{l=1}^{t}w_l\cdot\left(\left(\frac{2}{(v^{(l)})^2-1}-\frac{2(v^{(l)})\operatorname{arccosh}(v^{(l)})}{((v^{(l)})^2-1)^{\frac{3}{2}}}\right)T_{li}T_{lj}+\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\frac{\partial T_{lj}}{\partial y_i}\right) \tag{46}$$

where

$$v^{(l)}=1-\frac{2K||x^{(l)}-y||_2^2}{(1+K||x^{(l)}||_2^2)(1+K||y||_2^2)}\,,\ D^{(l)}=(1+K||x^{(l)}||_2^2)(1+K||y||_2^2) \tag{47}$$

$$\frac{\partial T_{li}}{\partial y_i}=-\frac{4K}{D^{(l)}}-\frac{16K^2}{(D^{(l)})^2}y_i(x_i^{(l)}-y_i)(1+K\|x^{(l)}\|_2^2)+\frac{4K^2(1+K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\|x^{(l)}-y\|_2^2-\frac{16K^3(1+K\|x^{(l)}\|_2^2)^2}{(D^{(l)})^3}y_i^2\|x^{(l)}-y\|_2^2 \tag{48}$$

and for $i\neq j$

$$\frac{\partial T_{lj}}{\partial y_i}-\frac{8K^2(1+K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\left[y_j(x_i^{(l)}-y_i)+y_i(x_j^{(l)}-y_j)\right]-\frac{16K^3(1+K\|x^{(l)}\|_2^2)^2}{(D^{(l)})^3}y_iy_j\|x^{(l)}-y\|_2^2 \tag{49}$$

Remark. *While the above computation is challenging to verify theoretically, to ensure the correctness of our computation, we implemented a gradient check on test cases for which it was simple to generate the correct gradient with a simple ϵ-perturbation, and ensured that this value coincided with the gradient provided by our formulas above.*

Proof. (1) **Application of Gould's theorem**: For the minimization problem

$$y^*(x^{(1)},\cdots,x^{(t)})=\underset{y\in\mathbb{D}_K^n}{\arg\min}\, f(\{x\},y)$$

$$\text{where } f(\{x\},y)=\frac{1}{K}\sum_{l=1}^{t} w_l\cdot \operatorname{arccosh}\left(1-\frac{2K||x^{(l)}-y||_2^2}{(1+K||x^{(l)}||_2^2)(1+K||y||_2^2)}\right)^2$$

we can apply Theorem B.2 to find the the gradient of y^* with respect to each $x^{(i)}$:

$$\widetilde{\nabla}_{x^{(i)}}y^*(\{x\})=-\nabla_{yy}^2 f(\{x\},y^*(\{x\}))^{-1}\widetilde{\nabla}_{x^{(i)}}\nabla_y f(\{x\},y^*(\{x\}))$$

and this is exactly in the form of equation (51). Thus, our goal is to compute the Hessian $\nabla_{yy}^2 f(\{x\},y^*(\{x\}))$ and the mixed gradient $\widetilde{\nabla}_{x^{(i)}}\nabla_y f(\{x\},y^*(\{x\}))$.

(2) **Computing gradient $\nabla_y f$**: Denote

$$v^{(l)}=1-\frac{2K||x^{(l)}-y||_2^2}{(1+K||x^{(l)}||_2^2)(1+K||y||_2^2)}\,,\ D^{(l)}=(1+K||x^{(l)}||_2^2)(1+K||y||_2^2)$$

Then we have:

$$\nabla_y f(\{x\},y)=\frac{1}{K}\left[\sum_{l=1}^{t} w_l\cdot\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial v^{(l)}}{\partial y_1},\sum_{l=1}^{t} w_l\cdot\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial v^{(l)}}{\partial y_2},\cdots\right] \tag{50}$$

where we replace $\frac{\partial v^{(i)}}{\partial y_j}$ with the following expression T_{ij}:

$$\begin{aligned}T_{ij}=\frac{\partial v^{(i)}}{\partial y_j}&=-2K\frac{\partial}{\partial y_j}\left(\frac{\|x^{(i)}-y\|_2^2}{D^{(l)}}\right)=-\frac{2K}{(D^{(l)})^2}\left(D^{(i)}\frac{\partial}{\partial y_j}\left(\|x^{(i)}-y\|_2^2\right)-\|x^{(i)}-y\|_2^2\frac{\partial}{\partial y_j}(D^{(i)})\right)\\&=-\frac{2K}{(D^{(i)})^2}\left(D^{(i)}\cdot 2(y_j-x_j^{(i)})-\|x^{(i)}-y\|_2^2\cdot(1+K\|x^{(i)}\|_2^2)\cdot 2Ky_j\right)\\&=\frac{4K}{D^{(i)}}(x_j^{(i)}-y_j)+\frac{4K^2}{(D^{(i)})^2}y_j\|x^{(i)}-y\|_2^2\cdot(1+K\|x^{(i)}\|_2^2)\end{aligned}$$

which is exactly the formula of T_{ij} in equation (42).

(3) **Now we derive the formula for $\frac{\partial}{\partial x_{ik}}\nabla_y f$.**

(3-1) From our previous equation (50) to evaluate $\nabla_y f$, our goal is to evaluate all terms of the form

$$\begin{aligned}&\frac{\partial}{\partial x_{ik}}\left(\sum_{l=1}^{t} w_l\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\frac{\partial v^{(l)}}{\partial y_j}\right)=\sum_{l=1}^{t} w_l\cdot\frac{\partial}{\partial x_{ik}}\left(\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\right)\cdot\frac{\partial v^{(l)}}{\partial y_j}+\sum_{l=1}^{t} w_l\cdot\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial}{\partial x_{ik}}\left(\frac{\partial v^{(l)}}{\partial y_j}\right)\\&=\sum_{l=1}^{t} w_l\cdot\left(\frac{2}{(v^{(l)})^2-1}-\frac{2(v^{(l)})\operatorname{arccosh}(v^{(l)})}{((v^{(l)})^2-1)^{\frac{3}{2}}}\right)\cdot\frac{\partial v^{(l)}}{\partial x_{ik}}\cdot\frac{\partial v^{(l)}}{\partial y_j}+\sum_{l=1}^{t} w_l\cdot\frac{2\operatorname{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial}{\partial x_{ik}}\left(\frac{\partial v^{(l)}}{\partial y_j}\right)\end{aligned}$$

We already have the expression for $T_{lj}=\frac{\partial v^{(l)}}{\partial y_j}$ above, so we just need the expression for $\frac{\partial v^{(l)}}{\partial x_{ik}}$ and $\frac{\partial}{\partial x_{ik}}\left(\frac{\partial v^{(l)}}{\partial y_j}\right)$.

(3-2) We first evaluate $\frac{\partial}{\partial x_{ik}}\left(\frac{\partial v^{(l)}}{\partial y_j}\right)=\frac{\partial T_{lj}}{\partial x_{ik}}$.

$$\begin{aligned}\frac{\partial T_{lj}}{\partial x_{ik}}&=\frac{\partial}{\partial x_{ik}}\left(\frac{4K(x_j^{(l)}-y_j)}{D^{(l)}}\right)+\frac{\partial}{\partial x_{ik}}\left(\frac{4K^2y_j\|x^{(l)}-y\|_2^2\cdot(1+K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\right)\\&=4K\frac{\partial}{\partial x_{ik}}\left(\frac{x_j^{(l)}-y_j}{D^{(l)}}\right)+4K^2y_j\frac{\partial}{\partial x_{ik}}\left(\frac{\|x^{(l)}-y\|_2^2\cdot(1+K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\right)\end{aligned}$$

This expression would become zero if $l \neq i$, so we only need to consider the case $l = i$.

For the first expression, when $k = j$, we have

$$\begin{aligned}
4K\frac{\partial}{\partial x_{ij}}\left(\frac{x_j^{(i)} - y_j}{D^{(i)}}\right) &= \frac{4K}{(D^{(i)})^2}\left(D^{(i)}\frac{\partial}{\partial x_{ij}}(x_j^{(i)} - y_j) - (x_j^{(i)} - y_j)\frac{\partial}{\partial x_{ij}}D^{(i)}\right) \\
&= \frac{4K}{(D^{(i)})^2}\left(D^{(i)} - (x_j^{(i)} - y_j)\cdot 2Kx_j^{(i)}(1 + K\|y\|_2^2)\right) \\
&= \frac{4K}{D^{(i)}} - \frac{8K^2}{(D^{(i)})^2}x_j^{(i)}(x_j^{(i)} - y_j)(1 + K\|y\|_2^2)
\end{aligned}$$

For the first expression, when $k \neq j$, we have

$$\begin{aligned}
4K\frac{\partial}{\partial x_{ik}}\left(\frac{x_j^{(i)} - y_j}{D^{(i)}}\right) &= \frac{4K}{(D^{(i)})^2}\left(-(x_j^{(i)} - y_j)\frac{\partial}{\partial x_{ik}}D^{(i)}\right) = \frac{4K}{(D^{(i)})^2}\left(-(x_j^{(i)} - y_j)\cdot 2Kx_k^{(i)}(1 + K\|y\|_2^2)\right) \\
&= -\frac{8K^2}{(D^{(i)})^2}x_k^{(i)}(x_j^{(i)} - y_j)(1 + K\|y\|_2^2)
\end{aligned}$$

For the second expression, we have

$$\begin{aligned}
&4K^2y_j\frac{\partial}{\partial x_{ik}}\left(\frac{\|x^{(i)} - y\|_2^2\cdot(1 + K\|x^{(i)}\|_2^2)}{(D^{(i)})^2}\right) \\
&= \frac{4K^2y_j}{(D^{(i)})^4}((D^{(i)})^2\|x^{(i)} - y\|_2^2\frac{\partial}{\partial x_{ik}}(1 + K\|x^{(i)}\|_2^2) + (D^{(i)})^2(1 + K\|x^{(i)}\|_2^2)\frac{\partial}{\partial x_{ik}}(\|x^{(i)} - y\|_2^2) \\
&\qquad - \|x^{(i)} - y\|_2^2\cdot(1 + K\|x^{(i)}\|_2^2)\frac{\partial}{\partial x_{ik}}(D^{(i)})^2) \\
&= \frac{4K^2y_j}{(D^{(i)})^4}\cdot(D^{(i)})^2\|x^{(i)} - y\|_2^2\cdot 2Kx_k^{(i)} + \frac{4K^2y_j}{(D^{(i)})^4}\cdot(D^{(i)})^2(1 + K\|x^{(i)}\|_2^2)\cdot 2(x_k^{(i)} - y_k) \\
&\qquad - \frac{4K^2y_j}{(D^{(i)})^4}\cdot\|x^{(i)} - y\|_2^2\cdot(1 + K\|x^{(i)}\|_2^2)\cdot 2D^{(i)}\cdot 2Kx_k^{(i)}(1 + K\|y\|_2^2) \\
&= \frac{8K^3y_j}{(D^{(i)})^2}x_k^{(i)}\|x^{(i)} - y\|_2^2 + \frac{8K^2y_j}{(D^{(i)})^2}(x_k^{(i)} - y_k)(1 + K\|x^{(i)}\|_2^2) - \frac{16K^3y_j}{(D^{(i)})^3}x_k^{(i)}\|x^{(i)} - y\|_2^2(1 + K\|x^{(i)}\|_2^2)(1 + K\|y\|_2^2) \\
&= \frac{8K^3y_j}{(D^{(i)})^2}x_k^{(i)}\|x^{(i)} - y\|_2^2 + \frac{8K^2y_j}{(D^{(i)})^2}(x_k^{(i)} - y_k)(1 + K\|x^{(i)}\|_2^2) - \frac{16K^3y_j}{(D^{(i)})^2}x_k^{(i)}\|x^{(i)} - y\|_2^2 \\
&= \frac{8K^2y_j}{(D^{(i)})^2}(x_k^{(i)} - y_k)(1 + K\|x^{(i)}\|_2^2) - \frac{8K^3y_j}{(D^{(i)})^2}x_k^{(i)}\|x^{(i)} - y\|_2^2
\end{aligned}$$

Thus, we have for $k = j$,

$$\frac{\partial T_{ij}}{\partial x_{ij}} = \frac{4K}{D^{(i)}} - \frac{8K^2}{(D^{(i)})^2}x_j^{(i)}(x_j^{(i)} - y_j)(1 + K\|y\|_2^2) + \frac{8K^2y_j}{(D^{(i)})^2}(x_j^{(i)} - y_j)(1 + K\|x^{(i)}\|_2^2) - \frac{8K^3y_j}{(D^{(i)})^2}x_j^{(i)}\|x^{(i)} - y\|_2^2$$

which matches equation (44). Also, for $k \neq j$, we have

$$\frac{\partial T_{ij}}{\partial x_{ik}} = -\frac{8K^2}{(D^{(i)})^2}x_k^{(i)}(x_j^{(i)} - y_j)(1 + K\|y\|_2^2) + \frac{8K^2y_j}{(D^{(i)})^2}(x_k^{(i)} - y_k)(1 + K\|x^{(i)}\|_2^2) - \frac{8K^3y_j}{(D^{(i)})^2}x_k^{(i)}\|x^{(i)} - y\|_2^2$$

which matches equation (45).

(3-3) We now evaluate $\frac{\partial v^{(l)}}{\partial x_{ik}}$. We have

$$\frac{\partial v^{(l)}}{\partial x_{ik}} = -2K\frac{\partial}{\partial x_{ik}}\left(\frac{\|x^{(l)}-y\|_2^2}{D^{(l)}}\right) = -\frac{2K}{(D^{(i)})^2}\left(D^{(l)}\frac{\partial}{\partial x_{ik}}\left(\|x^{(l)}-y\|_2^2\right) - \|x^{(l)}-y\|_2^2\frac{\partial}{\partial x_{ik}}(D^{(l)})\right)$$

Note that is also zero when $l \neq i$, so we'll assume $l = i$. This will give

$$\begin{aligned}
M_{ik} = \frac{\partial v^{(i)}}{\partial x_{ik}} &= -2K\frac{\partial}{\partial x_{ik}}\left(\frac{\|x^{(i)}-y\|_2^2}{D^{(i)}}\right) = -\frac{2K}{(D^{(i)})^2}\left(D^{(i)}\frac{\partial}{\partial x_{ik}}\left(\|x^{(i)}-y\|_2^2\right) - \|x^{(i)}-y\|_2^2\frac{\partial}{\partial x_{ik}}(D^{(i)})\right) \\
&= -\frac{2K}{(D^{(i)})^2}\left(D^{(i)}2(x_k^{(i)}-y_k) - \|x^{(i)}-y\|_2^2\cdot 2Kx_k^{(i)}(1+K\|y\|_2^2)\right) \\
&= \frac{4K}{D^{(i)}}(y_k - x_k^{(i)}) + \frac{4K^2}{(D^{(i)})^2}\cdot x_k^{(i)}\|x^{(i)}-y\|_2^2\cdot(1+K\|y\|_2^2)
\end{aligned}$$

which matches equation (43).

(3-4) Now we put everything together:

$$\begin{aligned}
&\frac{\partial}{\partial x_{ik}}\left(\sum_{l=1}^{t} w_l\cdot\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\frac{\partial v^{(l)}}{\partial y_j}\right) \\
&= \sum_{l=1}^{t} w_l\cdot\left(\frac{2}{(v^{(l)})^2-1} - \frac{2(v^{(l)})\,\mathrm{arccosh}(v^{(l)})}{((v^{(l)})^2-1)^{\frac{3}{2}}}\right)\cdot\frac{\partial v^{(l)}}{\partial x_{ik}}\cdot\frac{\partial v^{(l)}}{\partial y_j} + \sum_{l=1}^{t} w_l\cdot\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial}{\partial x_{ik}}\left(\frac{\partial v^{(l)}}{\partial y_j}\right) \\
&= w_l\cdot\left(\left(\frac{2}{(v^{(i)})^2-1} - \frac{2(v^{(i)})\,\mathrm{arccosh}(v^{(i)})}{((v^{(i)})^2-1)^{\frac{3}{2}}}\right)\cdot\frac{\partial v^{(i)}}{\partial x_{ik}}\cdot\frac{\partial v^{(i)}}{\partial y_j} + \frac{2\,\mathrm{arccosh}(v^{(i)})}{\sqrt{(v^{(i)})^2-1}}\frac{\partial}{\partial x_{ik}}\left(\frac{\partial v^{(i)}}{\partial y_j}\right)\right) \\
&= w_l\cdot\left(\left(\frac{2}{(v^{(i)})^2-1} - \frac{2(v^{(i)})\,\mathrm{arccosh}(v^{(i)})}{((v^{(i)})^2-1)^{\frac{3}{2}}}\right)M_{ik}T_{ij} + \frac{2\,\mathrm{arccosh}(v^{(i)})}{\sqrt{(v^{(i)})^2-1}}\frac{\partial T_{ij}}{\partial x_{ik}}\right)
\end{aligned}$$

where T_{ij} is given by equation (42), $\frac{\partial T_{ij}}{\partial x_{ik}}$ is given by equations (44) and (45), and M_{ik} is given by equation (43).

Now from equation (50), to obtain the gradient for the original function, we need to multiply the above expression by an additional factor of $\frac{1}{K}$, and then the RHS would match what we want in equation (41).

(4) **Finally, we derive the formula for Hessian $\nabla^2_{yy}f$:**

(4-1) From our formula for $\nabla_y f$ in (50), our goal is to evaluate all terms of the form

$$\begin{aligned}
&\frac{\partial}{\partial y_i}\left(\sum_{l=1}^{t} w_l\cdot\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\frac{\partial v^{(l)}}{\partial y_j}\right) = \sum_{l=1}^{t} w_l\cdot\frac{\partial}{\partial y_i}\left(\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\right)\cdot\frac{\partial v^{(l)}}{\partial y_j} + \sum_{l=1}^{t} w_l\cdot\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial}{\partial y_i}\left(\frac{\partial v^{(l)}}{\partial y_j}\right) \\
&= \sum_{l=1}^{t} w_l\cdot\left(\frac{2}{(v^{(l)})^2-1} - \frac{2(v^{(l)})\,\mathrm{arccosh}(v^{(l)})}{((v^{(l)})^2-1)^{\frac{3}{2}}}\right)\cdot\frac{\partial v^{(l)}}{\partial y_i}\cdot\frac{\partial v^{(l)}}{\partial y_j} + \sum_{l=1}^{t} w_l\cdot\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial}{\partial y_i}\left(\frac{\partial v^{(l)}}{\partial y_j}\right)
\end{aligned}$$

(4-2) We now consider evaluating $\frac{\partial}{\partial y_i}\left(\frac{\partial v^{(l)}}{\partial y_j}\right) = \frac{\partial T_{lj}}{\partial y_i}$.

$$\begin{aligned}
\frac{\partial T_{lj}}{\partial y_i} &= \frac{\partial}{\partial y_i}\left(\frac{4K(x_j^{(l)}-y_j)}{D^{(l)}}\right) + \frac{\partial}{\partial y_i}\left(\frac{4K^2y_j\|x^{(l)}-y\|_2^2\cdot(1+K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\right) \\
&= 4K\frac{\partial}{\partial y_i}\left(\frac{x_j^{(l)}-y_j}{D^{(l)}}\right) + 4K^2(1+K\|x^{(l)}\|_2^2)\frac{\partial}{\partial y_i}\left(\frac{y_j\|x^{(l)}-y\|_2^2\cdot}{(D^{(l)})^2}\right)
\end{aligned}$$

For the first expression, when $i = j$, we have

$$\begin{aligned}
4K\frac{\partial}{\partial y_i}\left(\frac{x_i^{(l)} - y_i}{D^{(l)}}\right) &= \frac{4K}{(D^{(l)})^2}\left(D^{(l)}\frac{\partial}{\partial y_i}(x_i^{(l)} - y_i) - (x_i^{(l)} - y_i)\frac{\partial}{\partial y_i}D^{(l)}\right) \\
&= \frac{4K}{(D^{(l)})^2}\left(-D^{(l)} - (x_i^{(l)} - y_i)\cdot 2Ky_i(1 + K\|x^{(l)}\|_2^2)\right) \\
&= -\frac{4K}{D^{(l)}} - \frac{8K^2}{(D^{(l)})^2}y_i(x_i^{(l)} - y_i)(1 + K\|x^{(l)}\|_2^2)
\end{aligned}$$

For the first expression, when $i \neq j$, we have

$$\begin{aligned}
4K\frac{\partial}{\partial y_i}\left(\frac{x_j^{(l)} - y_j}{D^{(l)}}\right) &= \frac{4K}{(D^{(l)})^2}\left(-(x_j^{(l)} - y_j)\frac{\partial}{\partial y_i}D^{(l)}\right) = \frac{4K}{(D^{(l)})^2}\left(-(x_j^{(l)} - y_j)\cdot 2Ky_i(1 + K\|x^{(l)}\|_2^2)\right) \\
&= -\frac{8K^2}{(D^{(l)})^2}y_i(x_j^{(l)} - y_j)(1 + K\|x^{(l)}\|_2^2)
\end{aligned}$$

For the second expression, when $i = j$, we have

$$\begin{aligned}
&4K^2(1 + K\|x^{(l)}\|_2^2)\frac{\partial}{\partial y_i}\left(\frac{y_i\|x^{(l)} - y\|_2^2}{(D^{(l)})^2}\right) \\
&= \frac{4K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^4}\left((D^{(l)})^2 y_i\frac{\partial}{\partial y_i}(\|x^{(l)} - y\|_2^2) + (D^{(l)})^2\|x^{(l)} - y\|_2^2 - y_i\|x^{(l)} - y\|_2^2\frac{\partial}{\partial y_i}(D^{(l)})^2\right) \\
&= \frac{4K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^4}\left((D^{(l)})^2 y_i\cdot 2(y_i - x_i^{(l)}) + (D^{(l)})^2\|x^{(l)} - y\|_2^2 - y_i\|x^{(l)} - y\|_2^2\cdot 2D^{(l)}\cdot 2Ky_i(1 + K\|x^{(l)}\|_2^2)\right) \\
&= \frac{8K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}y_i(y_i - x_i^{(l)}) + \frac{4K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\|x^{(l)} - y\|_2^2 - \frac{16K^3(1 + K\|x^{(l)}\|_2^2)^2}{(D^{(l)})^3}y_i^2\|x^{(l)} - y\|_2^2
\end{aligned}$$

For the second expression, when $i \neq j$, we have

$$\begin{aligned}
&4K^2(1 + K\|x^{(l)}\|_2^2)\frac{\partial}{\partial y_i}\left(\frac{y_j\|x^{(l)} - y\|_2^2\cdot}{(D^{(l)})^2}\right) \\
&= \frac{4K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^4}\left((D^{(l)})^2 y_j\frac{\partial}{\partial y_i}(\|x^{(l)} - y\|_2^2) - y_j\|x^{(l)} - y\|_2^2\frac{\partial}{\partial y_i}(D^{(l)})^2\right) \\
&= \frac{4K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^4}\left((D^{(l)})^2 y_j\cdot 2(y_i - x_i^{(l)}) - y_j\|x^{(l)} - y\|_2^2\cdot 2D^{(l)}\cdot 2Ky_i(1 + K\|x^{(l)}\|_2^2)\right) \\
&= \frac{8K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}y_j\cdot(y_i - x_i^{(l)}) - \frac{16K^3(1 + K\|x^{(l)}\|_2^2)^2}{(D^{(l)})^3}y_iy_j\|x^{(l)} - y\|_2^2
\end{aligned}$$

Thus, for $i = j$, we have

$$\frac{\partial T_{li}}{\partial y_i} = -\frac{4K}{D^{(l)}} - \frac{16K^2}{(D^{(l)})^2}y_i(x_i^{(l)} - y_i)(1 + K\|x^{(l)}\|_2^2) + \frac{4K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\|x^{(l)} - y\|_2^2 - \frac{16K^3(1 + K\|x^{(l)}\|_2^2)^2}{(D^{(l)})^3}y_i^2\|x^{(l)} - y\|_2^2$$

For $i \neq j$, we have

$$\frac{\partial T_{lj}}{\partial y_i} = -\frac{8K^2(1 + K\|x^{(l)}\|_2^2)}{(D^{(l)})^2}\left[y_j(x_i^{(l)} - y_i) + y_i(x_j^{(l)} - y_j)\right] - \frac{16K^3(1 + K\|x^{(l)}\|_2^2)^2}{(D^{(l)})^3}y_iy_j\|x^{(l)} - y\|_2^2$$

which are exactly equations (48) and (49).

(4-3) Finally, we combine the results together:

$$\begin{aligned}
&\frac{\partial}{\partial y_i}\left(\sum_{l=1}^{t} w_l \cdot \frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\frac{\partial v^{(l)}}{\partial y_j}\right)\\
&=\sum_{l=1}^{t} w_l \cdot \left(\left(\frac{2}{(v^{(l)})^2-1}-\frac{2(v^{(l)})\,\mathrm{arccosh}(v^{(l)})}{((v^{(l)})^2-1)^{\frac{3}{2}}}\right)\cdot\frac{\partial v^{(l)}}{\partial y_i}\cdot\frac{\partial v^{(l)}}{\partial y_j}+\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot\frac{\partial}{\partial y_i}\left(\frac{\partial v^{(l)}}{\partial y_j}\right)\right)\\
&=\sum_{l=1}^{t} w_l \cdot \left(\left(\frac{2}{(v^{(l)})^2-1}-\frac{2(v^{(l)})\,\mathrm{arccosh}(v^{(l)})}{((v^{(l)})^2-1)^{\frac{3}{2}}}\right)T_{li}T_{lj}+\frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\frac{\partial T_{lj}}{\partial y_i}\right)
\end{aligned}$$

where T_{li}, T_{lj} are given in equation (42) and $\frac{\partial T_{lj}}{\partial y_i}$ is given in equation (48) and (49).

Similarly, to obtain the gradient for the original function, we need to multiply the above expresssion by an additional factor of $\frac{1}{K}$, and the resulting RHS would match what we want in equation (46).

□

Theorem F.4. *Assume the same construction as the above theorem. However, consider f as a function of the weights w. Then the gradient of y^* with respect to the weights w is given by*

$$\widetilde{\nabla}_w y^*(\{x\}) = -\nabla^2_{yy} f(\{x\}, y^*(\{x\}))^{-1}\widetilde{\nabla}_w \nabla_y f(\{x\}, y^*(\{x\})) \tag{51}$$

Note that we assume implicitlyt that the weight is an input.

(1) Terms of $\widetilde{\nabla}_w \nabla_y f(\{x\}, y^*(\{x\}))$ are given by

$$\frac{\partial}{\partial w_i}\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\})) = \frac{1}{K}\cdot\frac{2\,\mathrm{arccosh}(v^{(i)})}{\sqrt{(v^{(i)})^2-1}}\cdot T_{ij} \tag{52}$$

(2) Terms of $\nabla_{yy} f(\{x\}, y^*(\{x\}))$ are given in Equation 46

Proof. We note that Equation 50 gives values for $\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\}))$. These are given by

$$\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\})) = \frac{1}{K}\sum_{l=1}^{t} w_l \cdot \frac{2\,\mathrm{arccosh}(v^{(l)})}{\sqrt{(v^{(l)})^2-1}}\cdot T_{lj}$$

It follows immediately that we have

$$\frac{\partial}{\partial w_i}\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\})) = \frac{1}{K}\cdot\frac{2\,\mathrm{arccosh}(v^{(i)})}{\sqrt{(v^{(i)})^2-1}}\cdot T_{ij}$$

Note that we have already shown the computation for the Hessian in F.3 □

Theorem F.5. *Assume the same construction but instead let f be a function of curvature K. The derivative with respect to the curvature is given by*

$$\widetilde{\nabla}_K y^*(\{x\}) = -\nabla^2_{yy} f(\{x\}, y^*(\{x\}))^{-1}\widetilde{\nabla}_K \nabla_y f(\{x\}, y^*(\{x\})) \tag{53}$$

where we again implicitly assume curvature is an input.

(1) Terms of $\widetilde{\nabla}_K \nabla_y f(\{x\}, y^*(\{x\}))$ are given by

$$\frac{d}{dK}\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\})) = \sum_{i=1}^{t} -\frac{w_i \operatorname{arccosh}(v^{(i)}) T_{ij}}{K^2\sqrt{(v^{(i)})^2-1}} + \frac{w_i T_{ij}}{K}\left(\frac{2}{((v^{(i)})^2+1)^{3/2}} - \frac{4v^{(i)}\operatorname{arccosh}(v^{(i)})}{((v^{(i)})^2-1)^2}\right)\frac{dv^{(i)}}{dK} + \frac{w_i}{K}\frac{2\operatorname{arccosh}((v^{(i)})^2)}{\sqrt{(v^{(i)})^2-1}}\frac{dT_{ij}}{dK} \tag{54}$$

where

$$\frac{dv^{(i)}}{dK} = \frac{\|2(v^{(i)})^2 - y_2^2\|(\|x^{(i)}\|_2^2\|y\|_2^2K^2 - 1)}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2} \tag{55}$$

and

$$\frac{dT_{ij}}{dK} = \frac{4(x_j^{(i)} - y_j)(1 - \|x^{(i)}\|_2^2\|y\|_2^2K^2)}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2} + \frac{8Ky_i\|x^{(i)}-y\|_2^2}{(1+K\|y\|_2^2)^3} \tag{56}$$

(2) Terms of $\nabla_{yy}^2 f(\{x\}, y^*(\{x\}))^{-1}$ are given in Equation 46

Proof. We note that Equation 50 gives values for $\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\}))$. These are given by

$$\frac{\partial}{\partial y_j} f(\{x\}, y^*(\{x\})) = \frac{1}{K}\sum_{i=1}^{t} w_i \cdot \frac{2\operatorname{arccosh}(v^{(i)})}{\sqrt{(v^{(i)})^2-1}} \cdot T_{ij}$$

By applying product rule, we see that Equation 56 is valid. To check the explicit derivatives, as a reminder we write the equations for $v^{(i)}$ and T_{ij}.

$$v^{(i)} = 1 - \frac{2K\|x^{(i)}-y\|_2^2}{(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)}$$

$$T_{ij} = \frac{4K(x_j^{(i)} - y_j)}{(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)} + \frac{4K^2y_i\|x^{(i)}-y\|_2^2}{(1+K\|y\|_2^2)^2}$$

We can calculate the derivative of of $v^{(i)}$ by applying quotient rule. In particular

$$\begin{aligned}
\frac{dv^{(i)}}{dK} &= \frac{d}{dK}\left(1 - \frac{2K\|x^{(i)}-y\|_2^2}{(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)}\right) \\
&= -\frac{d}{dK}\left(\frac{2K\|x^{(i)}-y\|_2^2}{(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)}\right) \\
&= -\frac{\left((1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)\right)2\|v^{(i)}-y\|_2^2 - 2K\|v^{(i)}-y\|_2^2\left(\|x^{(i)}\|+\|y\|_2^2+2K\|x^{(i)}\|_2^2\|y\|_2^2\right)}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2} \\
&= \frac{\|2(v^{(i)})^2 - y_2^2\|(\|x^{(i)}\|_2^2\|y\|_2^2K^2 - 1)}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2}
\end{aligned}$$

Similarly, we can calculate the derivative of T_{ij} by applying similar rules. In particular we see that

$$\frac{d}{dK}\left(\frac{4K(x_j^{(i)}-y_j)}{(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)}\right)$$
$$=\frac{4(x_j^{(i)}-y_j)(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)-4K(x_j^{(i)}-y_j)\left(\|x^{(i)}\|+\|y\|_2^2+2K\|x^{(i)}\|_2^2\|y\|_2^2\right))}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2}$$
$$=\frac{4(x_j^{(i)}-y_j)(1-\|x^{(i)}\|_2^2\|y\|_2^2K^2)}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2}$$

and

$$\frac{d}{dK}\left(\frac{4K^2y_i\|x^{(i)}-y\|_2^2}{(1+K\|y\|_2^2)^2}\right)=\frac{8Ky_i\|x^{(i)}-y\|_2^2(1+K\|y\|_2^2)-4K^2y_i\|x^{(i)}-y\|_2^2\|y\|_2^2}{(1+K\|y\|_2^2)^3}$$
$$=\frac{8Ky_i\|x^{(i)}-y\|_2^2}{(1+K\|y\|_2^2)^3}$$

Putting it all together, we have that

$$\frac{dT_{ij}}{dK}=\frac{d}{dK}\left(\frac{4K(x_j^{(i)}-y_j)}{(1+K\|x^{(i)}\|_2^2)(1+K\|y\|_2^2)}+\frac{4K^2y_i\|x^{(i)}-y\|_2^2}{(1+K\|y\|_2^2)^2}\right)$$
$$=\frac{4(x_j^{(i)}-y_j)(1-\|x^{(i)}\|_2^2\|y\|_2^2K^2)}{(1+K\|x^{(i)}\|_2^2)^2(1+K\|y\|_2^2)^2}+\frac{8Ky_i\|x^{(i)}-y\|_2^2}{(1+K\|y\|_2^2)^3}$$

which gives us the derivation for $\widetilde{\nabla}_K\nabla_y f(\{x\},y^*(\{x\}))$. We have already shown derivations for the Hessian, so this gives us the desired formulation of the derivation $\widetilde{\nabla}_K y^*(\{x\})$ with respect to curvature. □

G. Riemannian Batch Normalization as a Generalization of Euclidean Bach Normalization

In this section, we present the proof that our Riemannian batch normalization algorithm formulated in Algorithm 2 is a natural generalization of Euclidean batch normalization. In doing so, we also derive an explicit generalization which allows for vector-valued variance. To introduce this notion, we first define product manifolds. As opposed to previous methods, this allows us to introduce variance to complete the so-called "shift-and-scale" algorithm.

Product manifold: We can define a product manifold $\mathcal{M}=\prod_{i=1}^n\mathcal{M}_i$. If $\mathcal{M}_i$ is a k_i-dimensional manifold then $\dim\mathcal{M}=\sum_{i=1}^n k_i$ and $T_m\mathcal{M}=\prod_{i=1}^n T_{m_i}\mathcal{M}_i$ where $m=(m_i)_{i=1}^n$. If these manifolds are Riemannian with metrics g_i, this inherits a metric $g=\begin{pmatrix} g_1 & \cdots & \cdots & \cdots \\ \vdots & g_2 & \cdots & \cdots \\ \vdots & \vdots & \ddots & \\ \vdots & \vdots & & g_n \end{pmatrix}$ where the values are 0 for elements not in the diagonal matrices.

Fréchet Mean on Product Manifold: First note that the definition of Fréchet mean and variance given in equations (3), (4) can be extended to the case of product manifolds $\mathcal{M}=\prod_{i=1}^n\mathcal{M}_i$, where we will have $\mu_{fr}\in\mathcal{M}$, $\sigma_{fr}^2\in\mathbb{R}^n$. For such product manifolds, we can define the Fréchet mean and variance element-wise as

$$\mu_{fr:prod}=(\mu_{fr:\mathcal{M}_i}))_{i\in[n]} \tag{57}$$
$$\sigma_{fr:prod}^2=(\sigma_{fr:\mathcal{M}_i}^2)_{i\in[n]} \tag{58}$$

Proposition G.1. *The product Fréchet mean formula given in equation (57) is equivalent with the Fréchet mean given in equation (3) when applied on the product manifold.*

Proof. Note that on the product manifold $d^2_{\mathcal{M}}(x, y) = \sum_{i=1}^{m} d^2_{\mathcal{M}_i}(x, y)$. The proposition follows immediately since we are optimizing the disjoint objectives $\sigma^2_{fr}(\mathcal{M}_i)$, and the values in our product are disjoint and unrelated. □

Corollary G.2. *Consider $\mathbb{R}^n$ as a product manifold $\mathbb{R} \times \mathbb{R} \cdots \times \mathbb{R}$ n times, then the values in equation (57) and equation (58) correspond to the vector-valued Euclidean variance and mean.*

Proof. This follows almost directly from A.1. In particular, we recall that the Euclidean vector-valued mean and variance of input points $x^{(1)}, \ldots, x^{(t)}$ are defined by

$$\mu_{euc} = \frac{1}{t} \sum_{i=1}^{t} x^{(i)} \tag{59}$$

$$\sigma^2_{euc} = \frac{1}{t} \sum_{i=1}^{t} (x^{(i)})^2 - \left(\frac{1}{t} \sum_{i=1}^{t} x^{(i)} \right)^2 \tag{60}$$

Define μ_i to be the standard Euclidean mean and σ_i^2 to be the standard Euclidean variance for points $\{x_i^{(1)}, \ldots, x_i^{(t)}\}$. We see that the mean coincides as

$$\begin{aligned} \mu_{euc} &= \frac{1}{t} \sum_{i=1}^{t} x^{(i)} \\ &= \left(\frac{1}{t} \sum_{i=1}^{t} x_1^{(i)}, \ldots, \frac{1}{t} \sum_{i=1}^{t} x_n^{(i)} \right) \\ &= (\mu_1, \ldots, \mu_n) \end{aligned}$$

Similarly, we have that, for the variance

$$\begin{aligned} \sigma^2_{euc} &= \frac{1}{t} \sum_{i=1}^{t} (x^{(i)})^2 - \left(\frac{1}{t} \sum_{i=1}^{t} x^{(i)} \right)^2 \\ &= \left(\frac{1}{t} \sum_{i=1}^{t} (x_1^{(i)})^2 - \left(\frac{1}{t} \sum_{i=1}^{t} x_1^{(i)} \right)^2, \ldots, \frac{1}{t} \sum_{i=1}^{t} (x_n^{(i)})^2 - \left(\frac{1}{t} \sum_{i=1}^{t} x_n^{(i)} \right)^2 \right) \\ &= (\sigma_1^2, \ldots, \sigma_n^2) \end{aligned}$$

which is what was desired. □

Theorem G.3. *The Riemannian batch normalization algorithm presented in Algorithm 2 is equivalent to the Euclidean batch normalization algorithm when $\mathcal{M} = \mathbb{R}^n$ for all n during training time.*

Proof. We know from Corollary G.2 that the Fréchet mean and variance computed in the first two steps correspond to Euclidean mean and variance.

The core of this proof lies in the fact that the exponential and logarithmic maps on $\mathbb{R}^n$ are trivial. This is because $T_x\mathbb{R}^n = \mathbb{R}^n$, and as a manifold $\mathbb{R}^n$ exhibits the same distance function as its tangent space. Consider the Euclidean batch normalization formula given below:

$$x_i' = \gamma \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta \tag{61}$$

where we remove the ϵ term in (Ioffe & Szegedy, 2015) for clarity. We know that in Euclidean space, $\exp_x \overrightarrow{v} = x + \overrightarrow{v}$ and $\log_x y = y - x$ and $PT_{x \to x'}(\overrightarrow{v}) = \overrightarrow{v}$. Then we can rewrite equation (61) as:

$$x_i' = \exp_\beta \left(\gamma \frac{\log_{\mu_{\mathcal{B}}} x_i}{\sigma_{\mathcal{B}}} \right) = \exp_\beta \left(\frac{\gamma}{\sigma_{\mathcal{B}}} PT_{\mu_{\mathcal{B}} \to \beta}(\log_{\mu_{\mathcal{B}}} x_i) \right) \tag{62}$$

which corresponds to the "shifting-and-scaling" part of Algorithm 2, which is the only action during train time.

□

Remark. *We note that in our testing procedure (when acquiring the mean set statistics), we utilize the notion of momentum to define the test statistics instead of an iterative averaging method. To understand why this is necessary, we note that this is due to the lack of a non-closed formulation for the Fréchet mean. In particular if we have an iterative averaging procedure defined by*

$$\mu(\{x^{(1)}, \dots, x^{(n+1)}\}) = f(\mu(\{x^{(1)}, \dots, x^{(n)}\}, x^{(n+1)})$$

for some function f, then with this iterative procedure we can define the Fréchet mean by

$$\mu(\{x^{(1)}, \dots, x^{(n)}\}) = f(\dots f(f(x^{(1)}, x^{(2)}), x^{(3)}), \dots, f^{(n)})$$

If one can solve this, then we would have a closed form for the Fréchet mean and a way to iterativly update, allowing us to fully generalize the Euclidean Batch Normalization algorithm (with both training and testing algorithms) to general Riemannian manifolds.

Remark. *In this construction, we utilize product manifolds as a substitute for our Riemannian manifold $\mathbb{R}^n$. However, another natural formulation would be to normalize in the tangent space $T_m\mathcal{M}$. To see why this works, at least in the Euclidean space, note that $T_x\mathbb{R}^n = \mathbb{R}^n$ and so normalizing variance in the tangent space would be equivalent. However, this is due to the fact that $T_x\mathbb{R}^n$ has a coordinate basis which is invariant under parallel transport.*

However, on general Riemannian manifolds, this is not necessarily the case. For example, on the ball $\mathbb{D}^n_{-1}$, we normally define the basis as vectors of $\mathbb{R}^n$. But, we note that when we transport these vectors (say from two points on a non-diameter geodesic) then the bases vectors change. This means that it becomes difficult to establish a standard coordinate basis for variance normalization. This is a property called ***holonomy*** *which informally examines how much information is lost under parallel transport because of curvature.*

H. Additional Experimental Results and Details

H.1. Training and Architectural Details

All experiments on link-prediction and batch normalization were run using modifications of the code provided by Chami et al. (2019). Every experiment was run on a NVIDIA RTX 2080Ti GPU until improvement, as measured by the ROC AUC metric, had not occurred over the course of 100 epochs (this is also the default stopping criterion for the experiments in Chami et al. (2019)).

For clarity, we describe the original proposed hyperbolic graph convolution proposed by HGCN (Chami et al., 2019) and our modification. The original hyperbolic convolution involves a hyperbolic linear transformation (down from the number of dataset features to 128), followed by a hyperbolic aggregation step, followed by a hyperbolic activation. Our modification uses instead a hyperbolic linear transformation (down from the number of input features to 128), followed by the differentiable Fréchet mean operation we developed, followed by a hyperbolic activation.

For the batch normalization experiments, the baseline encoder was a hyperbolic neural network (Ganea et al., 2018) with only two layers (the first going from the input feature dimension to 128, and the second going from 128 to 128). Our modification instead used two hyperbolic linear layers of the same dimensions with hyperbolic batch normalization layers following each layer.

H.2. Additional Hyperbolic Batch Normalization Results

We present additional batch normalization results to supplement the results on CoRA (Sen et al., 2008) we provided in the main paper. Specifically, we run with the same HNN (Ganea et al., 2018) and HNN with batch normalization models we ran with previously, except now we run on the Disease and Disease-M datasets (Chami et al., 2019). Validation results are shown in Figures 3 and 4. Notice that we see both faster convergence in loss and ROC AUC, as well as significantly better final results.

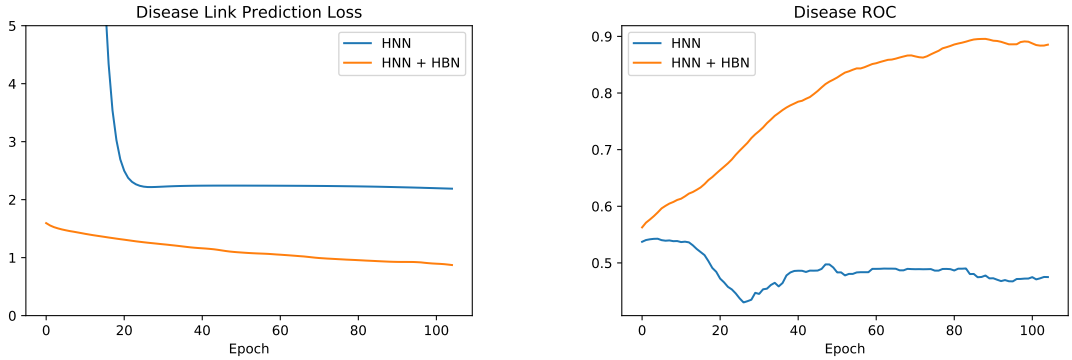

Figure 3. The graphs above correspond to a comparison of the HNN baseline, which uses a two-layer hyperbolic neural network encoder, and the baseline augmented with hyperbolic batch normalization after each layer. The experiments are run with the Disease (Chami et al., 2019) dataset. The top plot shows the comparison in terms of validation loss, and the bottom plot shows the comparison in terms of validation ROC AUC. Both figures show that we converge faster and attain better performance with respect to the relevant metric.

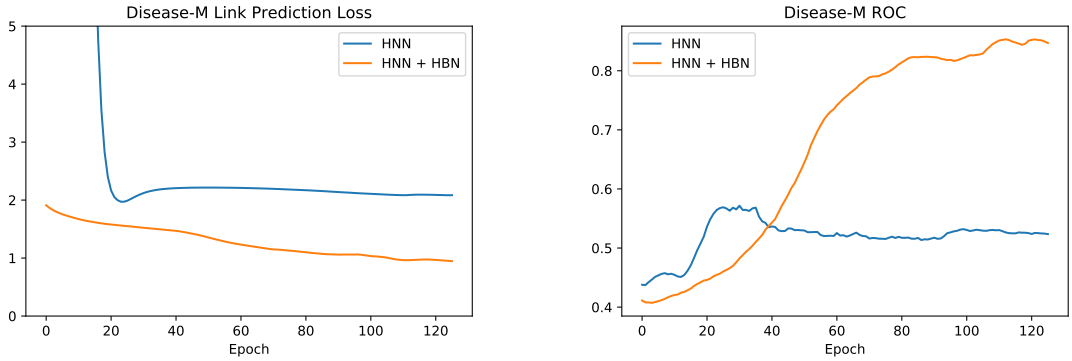

Figure 4. The graphs above correspond to a comparison of the HNN baseline, which uses a two-layer hyperbolic neural network encoder, and the baseline augmented with hyperbolic batch normalization after each layer. The experiments are run with the Disease-M dataset (Chami et al., 2019). As for the figure above, a comparison of validation loss and validation ROC AUC between the two approaches is given. Note that our batch normalization augmented network performs better.

H.3. Pseudo-means Warp Geometry

In our main paper, we frequently state that pseudo-Fréchet means warp geometry and are hence less desirable than the true Fréchet mean. Here we describe two common pseudo-means used in the literature and illustrate evidence for this fact.

One of these means is the tangent space aggregation proposed by (Chami et al., 2019). We saw from the main paper that this mean yielded worse performance on link prediction tasks when compared to the Fréchet mean. We can view this as evidence of the fact that taking the mean in the tangent space does not yield a representation that lets the model aggregate features optimally. Another general closed form alternatives to the Fréchet mean that has gained some traction is the Einstein midpoint; this has been used, due to its convenient closed form, by the Hyperbolic Attention Networks paper (Gulcehre et al., 2019). Although it has a closed form, this mean is not guaranteed to solve the naturally desirable Fréchet variance minimization.

Here we experimentally illustrate that both of these means do not in general attain the minimum Fréchet variance, and present the relative percentage by which both methods perform are worse (than the optimal variance obtained by the true mean). We conduct mean tests on ten randomly generated 16-dimensional on-manifold points for 100 trials. The points are generated in the Klein model of hyperbolic space, the Einstein midpoint is taken, then the points are translated into the hyperboloid model where we compute the mean using tangent space aggregation and our method. This is done for fair comparison, so that all methods deal with the same points. The results are shown in Table 5. Notice that the tangent space aggregation method is very off, but this is somewhat expected since it trades global geometry for local geometry. The Einstein midpoint performs much better in this case, but can still be quite off as is demonstrated by its $> 5\%$ relative error average and high 13% relative error standard deviation.

Table 5. Fréchet variance of various pseudo-means; we run our approach to become accurate within $\epsilon = 10^{-12}$ of the true Fréchet mean, and then report how much worse the other means are in terms of attaining the minimum variance. The average over 100 trials and corresponding standard deviation is reported. The primary baselines are the tangent space aggregation (Chami et al., 2019) and the Einstein midpoint (Gulcehre et al., 2019) methods.

.

Mean	Relative Error (Fréchet Variance)	Distance in Norm from True Mean
Tangent space aggregation (Chami et al., 2019)	$246.7\%_{\pm 4.5\%}$	$28207146.1_{\pm 8300242.0}$
Einstein midpoint (Gulcehre et al., 2019)	$6.8\%_{\pm 13.5\%}$	$30.3_{\pm 65.3}$
Ours	0.0%	10^{-12}