6.8410: Shape Analysis, 2023
Instructor: Justin Solomon
TA: Theia Henderson

Manifold Optimization for Embedding Geometric Graphs

Kiril Bangachev[1]

[1]MIT Department of Electrical Engineering and Computer Science, kirilb@mit.edu

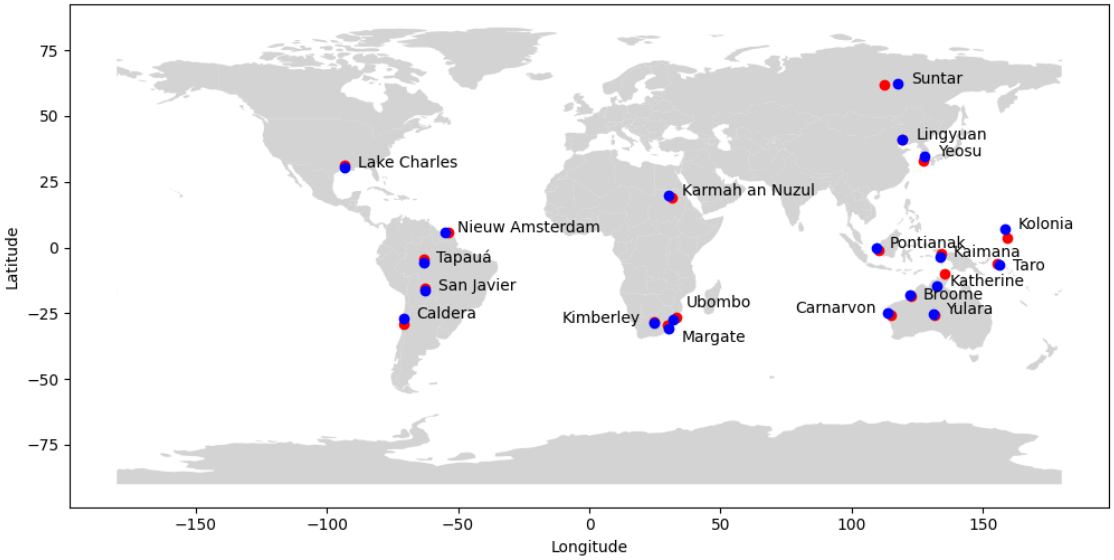

Figure 1: Red dots indicate inferred city locations from geometric graph after an optimal rotation and blue dots indicate true city locations.

Abstract

A geometric graph is defined by a manifold $\mathcal{M}$, threshold τ, distance function $d : \mathcal{M} \times \mathcal{M} \longrightarrow \mathbb{R}_{\geq 0}$, and n points $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ on $\mathcal{M}$. Then, $\mathbf{gg}(\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n)$ is a graph on vertex set $[n] = \{1, 2, \ldots, n\}$, where vertices i and j are connected if and only if $d(\mathbf{x}^i, \mathbf{x}^j) \leq \tau$. In the problem of embedding a geometric graph, one is given $\mathcal{M}, \tau, d$ and an adjacency matrix $\mathbf{G}$. The goal is to find n points $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^n$ on $\mathcal{M}$ such that $\mathbf{gg}(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^n)$ has adjacency matrix (approximately) equal to $\mathbf{G}$ [DDHM22, OMF20].

We propose a solution by formulating a corresponding minimzation problem over the product manifold $\mathcal{M}^{\times n}$ and running first- and second-order (Riemannian) optimization algorithms. Our approach improves on previous work in two different ways. 1) Generalizability. First-order optimization already achieves meaningful embeddings for a variety of different manifolds and distance functions, including spherical manifolds with the induced L_2 metric, hyperbolic space with the Minkowski metric, and tori with the Manhattan metric. Most previous work is restricted to L_2-distances in Euclidean space. 2) Robustness. The optimization approach performs extremely well when as much as a constant fraction of the graph edges are hidden. Again, this seems out of reach for most previous algorithms.

Our approach has applications to estimating the dimension of geometric graphs, estimating geodesic distances between latent vectors, recovering missing edges in geometric graphs, and reducing the amount of noise in geometric graphs.

1. Introduction

A geometric graph is defined by a Riemannian manifold $\mathcal{M}$, threshold τ, distance function $d : \mathcal{M} \times \mathcal{M} \longrightarrow \mathbb{R}_{\geq 0}$ and n points $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ on $\mathcal{M}$. Then, $\mathbf{gg}(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n)$ is a graph on vertex set $[n] = \{1, 2, \dots, n\}$, where vertices i and j are connected if and only if $d(\mathbf{x}^i, \mathbf{x}^j) \leq \tau$. The distance function does not need to be related to the extrinsic or intrinsic geodesic distance of $\mathcal{M}$, we only assume that it is continuous and, furthermore, is twice differentiable almost everywhere on $\mathcal{M} \times \mathcal{M}$.

Common in the literature is the related concept of random geometric graphs. Suppose that, in addition to $\mathcal{M}, \tau, d(\cdot,\cdot)$, one is also provided with a distribution $\mathcal{D}$ over $\mathcal{M}$. Then, the random geometric graph $\mathsf{RGG}(n, \mathcal{M}, \tau, d(\cdot,\cdot), \mathcal{D})$ is the distribution over geometric graphs $\mathbf{gg}(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n)$, where $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n \sim_{iid} \mathcal{D}$.

Random geometric graphs have been widely used to model networks across the sciences [DdC22]. Specific applications include protein-protein interactions and viral spread in the biological sciences [HRP08, PJ09], wireless networks and motion planning in engineering [HAB*09, SSH18], consensus dynamics and citation networks in the social sciences [XOLL16, ES16].

Associated to (random) geometric graphs are a wide range of inference tasks.

Estimating Geodesic Distances. The input is $\mathcal{M}, \tau, d(\cdot,\cdot), \mathcal{D}$ and an adjacency matrix $\mathbf{G}$ sampled from $\mathsf{RGG}(n, \mathcal{M}, \tau, d(\cdot,\cdot), \mathcal{D})$. One needs to approximately infer one (or all) of the geodesic distances between the hidden vectors $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$. This problem has applications to machine learning and is usually studied in the context when $d(\cdot,\cdot)$ is the extrinsic distance and one wants to infer the intrinsic distances [AVL12]. Usually, the geodesic distance is computed as a (weighted) shortest path distance in the graph. More sophisticated estimators also appear for specific setups (for example, in [DDHM22]).

Distinguishing From Erdős-Rényi. On input an $n \times n$ adjacency matrix $\mathbf{G}$ and $\mathcal{M}, \tau, d(\cdot,\cdot), \mathcal{D}$, one needs to test between $\mathsf{RGG}(n, \mathcal{M}, \tau, d(\cdot,\cdot), \mathcal{D})$ and the Erdős-Rényi graph $\mathsf{G}(n, q)$, in which each edge appears independently with probability q. The parameter q is chosen so that the expected density of the two graph models is the same. This problem has received a lot of attention in recent years in the high-dimensional statistics community when $\mathcal{M} = \mathbb{S}^{p-1}$. It turns out that as p grows, $\mathsf{RGG}(n, \mathbb{S}^{p-1}, \tau, \|\cdot\|_2, \mathsf{Unif})$ converges to $\mathsf{G}(n, q)$ in total variation and the problem becomes information-theoretically intractable ([DGLU11, BDER14] and a lot of subsequent work).

Estimating Dimension. The input is an $n \times n$ adjacency matrix $\mathbf{G}$ sampled from $\mathsf{RGG}(n, \mathcal{M}, \tau, \|\cdot\|_2, \mathsf{Unif})$, where $\mathcal{M}$ comes from some known family of manifolds such as $\{\mathbb{S}^{p-1}\}_{p \in \mathbb{N}}$ or $\{\mathbb{T}^p\}_{p \in \mathbb{N}}$. The goal is to recover the hidden dimension p. Recent empirical work focusing on this problem has surprisingly shown that many real-world networks have extremely low-dimension [AMS22]. There are also theoretical guarantees for specific low-dimensional models [FGKS23b]. Few computational and information-theoretic lower bounds are known for this problem [BDER14].

Embedding. The input is $\mathcal{M}, \tau, d(\cdot,\cdot)$ and an adjacency matrix $\mathbf{G}$ corresponding to $\mathbf{gg}(\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n)$, but the vectors $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ are hidden. The goal is to find n points $\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^n$ on $\mathcal{M}$ such that $\mathbf{gg}(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^n)$ has (approximately) adjacency matrix $\mathbf{G}$ [DDHM22, OMF20]. This is the focus of the current paper. We note that the embedding problem actually captures (at least) two different problems:

1. **Strong Embedding Problem.** The goal is to find and embedding which furthermore satisfies $(\mathbf{y}^1, \mathbf{y}^2, \dots, \mathbf{y}^n) \approx (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n)$ up to symmetries of the underlying space. The notion of $\approx$ can depend on the context.
2. **Weak Embedding Problem.** The goal is to find any (approximate) embedding of $\mathbf{G}$. One should note that in the weak embedding problem, one does not even require $\mathbf{G}$ to be generated as a geometric graph over $(\mathcal{M}, \tau, d(\cdot,\cdot))$ at the first place.

1.1. Applications of Embedding

Relation to Other Inference Tasks. Heuristically, an algorithm solving the strong embedding problem can be applied to some of the aforementioned tasks.

1. **Estimating Geodesic Distances** can be performed by using a strong embedding algorithm. One simply needs to compute the geodesic distances between embedded vectors.
2. **Estimating Dimension** can be performed by binary-searching over the smallest dimension in which an approximate embedding is successfully recovered. Concretely, one first chooses an embedding algorithm $\mathcal{A}$, error tolerance tol, and number of attempts $\kappa_{\mathsf{attempt}}$. Then, starting from dimension 2, one performs a binary search over dimensions, accepting the smallest dimension for which an embedding with at most tol incorrect edges is found out of $\kappa_{\mathsf{attempt}}$ attempts.

Robust Embeddings. In case the embedding algorithm is robust to noise and missing data, it can also be used to recover data.

1. **Missing Data.** Suppose that one is instead given the matrix $\mathbf{G}$ in which some of the entries are hidden. Can one recover the missing entries? This questions appears naturally in the context of social networks when one tries to recover a friendship graph by pooling data from several different social media - there could exist a pair of people who are not simultaneously part of any single platform, but we may still want to know if they are friends.
2. **Noise.** Suppose that one is given a corrupted version of $\mathbf{G}$, in which each entry is randomly replaced with probability q_e (the noise rate) with a Bernoulli random variable. Can one determine the corrupted entries? Can one reduce the amount of noise? These questions can be naturally interpreted in the context of social media, when one tries to identify "fake" friendships or followers in an online platform.

Graph Representations. Sometimes, it could be useful to find a representation of a graph as the adjacency matrix of a geometric graph, even if this is not how the adjacencies were originally generated. So far, this direction has mainly been utilized in theoretical work, for example, in proving hardness results for explicitly constructing matrices with the restricted-isometry propertry [Gam19]. Note that for this task, one only needs a weak embedding algorithm.

1.2. Related Problems

The problem of embedding geometric graphs is closely related to several other tasks in computational geometry.

Geometric Graph Recognition. Suppose that instead of solving the embedding problem, we simply want to decide whether there exists points $\mathbf{y}^1,\mathbf{y}^2,\dots,\mathbf{y}^n \in \mathcal{M}$ such that $\mathbf{gg}(\mathbf{y}^1,\mathbf{y}^2,\dots,\mathbf{y}^n)$ has adjacency matrix $\mathbf{G}$. It turns out that this problem is already NP-hard in one of the most natural cases - when $\mathcal{M}=\mathbb{R}^2, d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2, \tau = 1$ [BK98].

Sensor Network Localization. Suppose that there are n hidden vectors $\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n \in \mathcal{M}$. One is provided a set $\mathcal{E} \subseteq [n]\times[n]$ and a list $\mathcal{D} = \{(ij : d_{ij}),\ ij \in \mathcal{E}\}$, where d_{ij} is the (potentially noisy) distance $d(\mathbf{x}^i,\mathbf{x}^j)$. The goal is to recover $\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n$ on input $\mathcal{D}$ (up to the symmetries of the underlying space). This problem has received significant attention in the community (for example, [GK04, CLS12]). The corresponding decision problems are also known to be NP-hard [Sax79, Yem79].

1.3. This Work

Our approach to geometric graph embedding is based on minimizing a logit loss, which captures a relaxation of the number of mismatched edges. The loss function is motivated by [DDHM22].

To gain some intuition, consider first the number of mismatched edges:

$$\gamma(\mathbf{y}^1,\mathbf{y}^2,\dots,\mathbf{y}^n) = \frac{1}{2}\sum_{i,j}\left|\mathbb{1}[d(\mathbf{y}^i-\mathbf{y}^j)\le\tau] - G_{ij}\right|.$$

The problem with directly optimizing this quantity is that the indicator $\mathbb{1}[x\le\tau]$ has derivative 0 everywhere, except at τ, where the derivative is $+\infty$. Thus, derivative-based optimization approaches are unsuitable.

As common in machine learning, we replace this indicator with a soft version of it. Specifically, let $\beta>0$ be a "large" constant and consider the sigmoid $\sigma(x) = 1/(1+\exp(\beta x))$. The sigmoid leads us to optimizing a logit loss as in logistic regression. We massage the loss function one final time. Let $h:\mathbb{R}_{\ge 0}\longrightarrow\mathbb{R}$ be an increasing function. The role of h is to make the function $\mathbf{d} := h\circ d$ easy to differentiate. Let $\tau_h = h(\tau)$. We discuss how to choose β and $h(\cdot)$ later. Using these parameters, we can compute the logit loss

$$\begin{aligned} f(\mathbf{y}^1,\mathbf{y}^2,\dots,\mathbf{y}^n) := \\ -\sum_{i\neq j}\Bigg(G_{i,j}\ln\sigma(\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j)-\tau_h) \\ +(1-G_{i,j})\ln\Big(1-\sigma(\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j)-\tau_h)\Big)\Bigg). \end{aligned} \tag{1}$$

In the limit $\beta\longrightarrow+\infty$, the sigmoid $\sigma(x)$ converges to the indicator $\mathbb{1}[x\le 0]$. In this limit, $f=0$ exactly at points which satisfy $\mathbf{d}(\mathbf{x}^i,\mathbf{x}^j)\gg\tau_h \Longleftrightarrow G_{i,j}=0$ (equivalently $d(\mathbf{x}^i,\mathbf{x}^j)\gg\tau\Longleftrightarrow G_{i,j}=0$). Otherwise $f=+\infty$.

Since f is differentiable, we can minimize it over $(\mathcal{M})^{\times n}$ using simple first- and second- order methods. To do so, we use the `manopt` package [BMAS14].

We apply the optimization technique to three different pairs $(\mathcal{M},d(\cdot,\cdot))$ motivated by ongoing research on geometric graphs.

1. **Unit Sphere With L_2-Distance.** That is, $\mathcal{M}=\mathbb{S}^{p-1} = \{(u_1,u_2,\dots,u_p)\in\mathbb{R}^p : \sum_{i=1}^p u_i^2 = 1\}$ and $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2$. Note that in that case, $d(\cdot,\cdot)$ is an L_2-distance. Furthermore, even though $d(\cdot,\cdot)$ is defined extrinsically (via the L_2 distance in the ambient $\mathbb{R}^p$), geometric graphs are produced by the intrinsic distance. This is true because, for the spherical distance $d_{\mathbb{S}^{p-1}}(\cdot,\cdot)$,we have $d_{\mathbb{S}^{p-1}}(\mathbf{x},\mathbf{y}) = \arccos(\langle\mathbf{x},\mathbf{y}\rangle) = \arccos(2-\|\mathbf{x}-\mathbf{y}\|_2^2)$, so $\|\mathbf{x}-\mathbf{y}\|_2\le\tau\longleftrightarrow d_{\mathbb{S}^{p-1}}(\mathbf{x},\mathbf{y})\le\arccos(2-\tau^2)$. The spherical model is one of the most popular in the literature on random geometric graphs, especially in the high-dimensional regime ([DGLU11, BDER14, DdC22]).
2. **Hyperbolic Space with Minkowski Distance.** We use the Lorenizan formulation $\mathcal{M}=\mathbb{H}^p = \{(x_0,x_1,x_2,\dots,x_p)\in\mathbb{R}^{p+1} : x_0>0, -x_0^2+\sum_{i=1}^p x_i^2 = -1\}$. The intrinsic (Minkowski) distance function is given by

$$d_{\mathbb{H}^p}(\mathbf{x},\mathbf{y}) = 2\,\mathrm{arcsinh}\left(\frac{1}{2}\sqrt{-(x_0-y_0)^2+\sum_{i=1}^p(x_i-y_i)^2}\right).$$

   This is an intrinsic model with a hyperbolic metric, which is qualitatively very different from the L_2 metric (for example, it leads to constant negative curvature -1). Hyperbolic models are becoming increasingly popular in network science as they can capture important hierarchical structure and degree inhomogeneity [NK18].
3. **Tori with Manhattan Distance.** That is, $\mathcal{M}=\mathbb{T}^p=(\mathbb{S}^1)^{\times p}$ and

$$d_{\mathbb{T}^p,1}(\mathbf{x},\mathbf{y}) = \sum_{i=1}^p d_{\mathbb{S}^1}(x_i,y_i) = \sum_{i=1}^p \arccos(\langle x_i,y_i\rangle).$$

   Tori with the Manhattan L_1 distance and, more generally L_q distances, that is $d_q(\mathbf{x},\mathbf{y}) = (\sum_{u=1}^p d_{\mathbb{S}^1}(\mathbf{x}_u,\mathbf{y}_u)^q)^{1/q}$, are also a commonly studied model in the literature (for example, [FGKS23a, FGKS23b]).

The performance of our approach in these three different settings can be summarized as follows.

Manifold and Distance	Type	Strong Embedding	Weak Embedding
$(\mathbb{S}^{p-1}, d_{\mathbb{S}^{p-1}})$	Intrinsic, L_2 Distance	Only in low dimensions	Yes
$(\mathbb{H}^p, d_{\mathbb{H}^p})$	Intrinsic, Non-L_2 Distance	Only in low dimensions	Yes
$(\mathbb{T}^p, d_{\mathbb{T}^p,1})$	Neither intrinsic nor extrinsic, Non-L_2 Distance	No	Only In High Dimension

In Eq. (11), we explain why one could only hope to solve the strong embedding problem in low dimensions, so our algorithm has

the correct qualitative behaviour for $\mathbb{S}^{p-1}$and $\mathbb{H}^p$. Of course, the precise meaning of "low dimension" remains to be quantified.

In addition, in Section 4.2, we demonstrate that in the spherical case, our algorithm can be successfully applied to a variety of problems.

1. **Estimating Geodesic Distances** in the low-dimensional setting Section 4.2.1.
2. **Estimating Dimensions** in the low-dimensional setting Section 4.2.2.
3. **Recovering Missing Data** in the low-dimensional setting even when a constant fraction of the edges are erased Section 4.2.3.
4. **Recovering Corrupted Data** in the low-dimensional setting. Noise is consistently reduced, even if by a small factor Section 4.2.4.
5. **Representing Erdos-Renyi Graphs** as geometric graphs in dimension $d \approx n/3$ Section 4.2.6.

2. Related Work

Two different lines of research in the literature have addressed the problem of geometric graph embeddings.

Spectral Methods. In [OMF20], the authors present a spectral algorithm and show that it is asymptotically equivalent to an MLE embedding. More concretely, the authors solve the following problem. Suppose that $\mathbf{G}$ is the adjacency matrix of a "soft" geometric graph with vertices $\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n \in \mathbb{R}^p$. In it, edges are random functions depending on the latent vectors. Namely, $\mathbf{P}[\mathbf{G}_{ij}=1] = 1/(1+\exp(\tau - \langle \mathbf{x}_i,\mathbf{x}_j\rangle)) = \sigma^{1,\tau}(\mathbf{x}^i,\mathbf{x}^j)$. Then, upon observing $\mathbf{G}$, the authors want to find the maximizer $((\mathbf{x}^1)^*,(\mathbf{x}^2)^*,\dots,(\mathbf{x}^n)^*)$ of the log-likelihood

$$\begin{aligned}&\mathbf{LL}(\mathbf{x}^1,\dots,\mathbf{x}^n)\\&= \sum_{ij} G_{ij}\ln\sigma^{1,\tau}(\mathbf{x}^i,\mathbf{x}^j) + (1-G_{ij})\ln(1-\sigma^{1,\tau}(\mathbf{x}^i,\mathbf{x}^j))).\end{aligned} \tag{2}$$

It should be noted that the log-likelihood is the same as our Eq. (1) (for $\beta = 1$). The authors show that under suitable conditions, the following algorithm recovers an approximate maximizer $(\widehat{\mathbf{x}^1},\dots,\widehat{\mathbf{x}^n})$ in the sense that $\mathbf{LL}(\widehat{\mathbf{x}^1},\dots,\widehat{\mathbf{x}^n})/\mathbf{LL}((\mathbf{x}^1)^*,\dots,(\mathbf{x}^n)^*) \geq 1-\varepsilon$.

i Find the top eigenvectors $\mathbf{v}_1,\mathbf{v}_2,\dots,\mathbf{v}_d$ of $B = G - \frac{1}{1+\exp(\tau)}\mathbf{1}\mathbf{1}^T$, where τ can be estimated from G if unknown.
ii Perform logistic regression with strictly positive coefficients $\lambda_1,\lambda_2,\dots,\lambda_p$ of A on $\mathbf{v}_i\mathbf{v}_i^T$ for $1 \leq i \leq p$.
iii Set $\widehat{\mathbf{X}} = (\widehat{\mathbf{x}^1},\dots,\widehat{\mathbf{x}^n})$ to be a solution of $\mathbf{X}^T\mathbf{X} = \sum_{i=1}^p \lambda_i \mathbf{v}_i\mathbf{v}_i^T$.

This method has the clear advantage of a provable performance guarantee. However, it leaves several gaps:

1. The method is clearly only suitable for inner-product distance functions. Thus, instances like the hyperbolic Minkowski and toric Manhattan settings we consider are out of reach.
2. It is unclear how to adapt this approach so that $\widehat{\mathbf{x}^1},\dots,\widehat{\mathbf{x}^n}$ belong to a submanifold of $\mathbb{R}^p$. The simplest case of $\mathbb{S}^{p-1}$ already seems to pose a challenge as when solving the logistic regression in ii., one needs to ensure that $\sum_{i=1}^p \lambda_i \mathbf{v}_i\mathbf{v}_i^T$ has an all-ones diagonal.
3. It is unclear whether the method continues to yield an MLE estimator in the limit $\beta \longrightarrow +\infty$, corresponding to the "hard" geometric graph setting of the current work. The reason is that desired uniform convergence in [OMF20, Proof of Lemma 2] might fail.

Combinatorial Methods. In [DDHM22] and [DMM20], the authors focus on $\mathcal{M} = [0,1]^p$ with the L_2 distance. The main subroutine in their argument is estimating the euclidean distances between vertices from the adjacency matrix. Different estimators are used depending on the graph distance between points. The authors show that their technique produces an embedding $(\widehat{\mathbf{x}^1},\widehat{\mathbf{x}^2},\dots,\widehat{\mathbf{x}^n})$ such that the distortion to the hidden vectors $\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n$ which produced G is small. Formally, they address the objective function $\max_i \inf_\phi \|\phi(\widehat{\mathbf{x}_i}) - \mathbf{x}_i\|_2$, where ϕ is minimized over the symmetries of $(\mathcal{M},d(\cdot,\cdot))$ (which is just the dihedral group in the $[0,1]^p$ case). Their algorithm attains low distortion even if a small number of edges are deleted, thus robustly solving the strong embedding problem. However, this approach also leaves several gaps:

1. The method strongly relies on the choice $(\mathcal{M},d(\cdot,\cdot)) = ([0,1]^p,L_2)$ and is, thus, hard to adapt to other pairs $(\mathcal{M},d(\cdot,\cdot))$.
2. The method very heavily depends on the fact that $\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n$ come from the uniform distribution, which makes it hard to adapt. Perhaps more fundamentally, it depends on *knowing* the distribution of $\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n$ (for example, [DDHM22, Lemma 2.4,Lemma 2.5]).
3. It is unclear how the method adapts to hidden edges.

3. Technical Approach

Our meta-algorithm for solving the embedding problem is the following.

Meta-Algorithm for Embedding $\mathbf{G}$ in $(\mathcal{M},d(\cdot,\cdot),\tau)$

1. **Input Data:** $\mathbf{G},\mathcal{M},d(\cdot,\cdot),\tau$.
2. **Input Parameters:** β,h, and Riemannian Optimizer $\mathcal{O}$ (and required inputs to $\mathcal{O}$).
3. **Construct Loss** $f(\mathbf{y}^1,\mathbf{y}^2,\dots,\mathbf{y}^n)$ as in Eq. (1).
4. **Minimize** f over $\mathcal{M}^{\times n}$using $\mathcal{O}$.
5. **Return** the point $(\widehat{\mathbf{x}^1},\widehat{\mathbf{x}^2},\cdots,\widehat{\mathbf{x}^n})$.

For the most common first and second order methods, the required additional input to $\mathcal{O}$ are the gradient and/or hessian of f. In the case when $\mathcal{M}$ is a Riemannian submanifold of $\mathbb{R}^m$, which is the case for $\mathbb{S}^{p-1}$ and $\mathbb{T}^p$, it is enough to compute the Euclidean gradient and Hessian of f (more formally, of a smooth extension of f to an open set in $\mathbb{R}^m$) and use suitable projections. Specifically, let $\Pi_{T_\mathbf{x}} : \mathbb{R}^m \longrightarrow T_\mathbf{x}\mathcal{M}$ be the projection operator and $\mathcal{P}_{V,\mathbf{x}}$ be the differential of $\Pi_{T_\mathbf{x}}$ along vector V. The following hold [Bou23]:

$$\nabla^R f(\mathbf{x}) = \Pi_{T_\mathbf{x}}\nabla f(\mathbf{x}),$$
$$\mathsf{Hess}^R f(\mathbf{x})[V] = \Pi_{T_\mathbf{x}}(\mathsf{Hess} f(\mathbf{x})[V]) + \mathcal{P}_{V,\mathbf{x}}(\nabla f(\mathbf{x}) - \Pi_{T_\mathbf{x}}\nabla f(\mathbf{x})).$$

Above, ∇,Hess correspond to the Euclidean operators and ∇^R,Hess^R to their Riemannian counterparts. For a wide range of common manifolds, $\Pi_{T_\mathbf{x}},\mathcal{P}_{V,\mathbf{x}}$ are pre-comuputed in the software package `manopt` we use and, thus, it is enough to compute the

Euclidean gradients and Hessians. The case of $\mathbb{H}^p$ is more subtle because $\mathbb{H}^p$ is *not* a Riemannian submanifold of $\mathbb{R}^{p+1}$ since the metric is not the induced metric. Nevertheless, similar projection methods can be used in this case as well [WL18, FX06, BMAS14].

Using $\sigma'(x) = -\beta\sigma(x)(1-\sigma(x))$, we obtain.

$$\nabla_{\mathbf{y}^i} f(\mathbf{y}) = 2\beta \sum_{j\neq i} \left(G_{i,j} - \sigma(\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j) - \tau_h) \right) \nabla_{\mathbf{y}^i} \mathbf{d}(\mathbf{y}^i,\mathbf{y}^j). \quad (3)$$

Using σ as a short-hand for $\sigma(\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j) - \tau_h)$,

$$\begin{aligned}
&\nabla\nabla_{\mathbf{y}^i} f(\mathbf{y})[V] = \\
&= 2\sum_{j\neq i} \beta^2\sigma(1-\sigma)\Big(\langle V_i, \nabla_{\mathbf{y}^i}\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j)\rangle + \\
&\qquad + \langle V_j, \nabla_{\mathbf{y}^j}\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j)\rangle \Big) \nabla_{\mathbf{y}^i}\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j) \\
&+ 2\sum_{j\neq i} \beta \Big(G_{i,j} - \sigma \Big) \nabla_{\mathbf{y}^i}\nabla_{\mathbf{y}^i}\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j)[V_i] \\
&+ 2\sum_{j\neq i} \beta \Big(G_{i,j} - \sigma \Big) \nabla_{\mathbf{y}^j}\nabla_{\mathbf{y}^i}\mathbf{d}(\mathbf{y}^i,\mathbf{y}^j)[V_j].
\end{aligned} \quad (4)$$

We next give explicit formulae for the three cases of interest. Numerical results certifying the correctness of our gradients and Hessians can be found in Appendix A.

3.1. Spherical Manifold with Euclidean Distance

Suppose that $\mathcal{M} = \mathbb{S}^{p-1}$, embedded in $\mathbb{R}^p$ and $d(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2$. We use $h(x) = x^2$, which leads to $\mathbf{d}(\mathbf{x},\mathbf{y}) = h\circ d(\mathbf{x},\mathbf{y}) = 2 - 2\langle \mathbf{x},\mathbf{y}\rangle$. The respective gradient and hessian simplify to the following.

$$\nabla_{\mathbf{y}^i} f(\mathbf{y}) = -4\beta \sum_{j\neq i} \Big(G_{i,j} - \sigma \Big) \mathbf{y}^j. \quad (5)$$

$$\begin{aligned}
&\nabla\nabla_{\mathbf{y}^i} f(\mathbf{y})[V] = \\
&= 4\sum_{j\neq i} \beta^2\sigma(1-\sigma)\Big(\langle V_i, \mathbf{y}^j\rangle + \langle V_j, \mathbf{y}^i\rangle \Big) \mathbf{y}_j \\
&- 4\sum_{j\neq i} \beta \Big(G_{i,j} - \sigma \Big) V_j.
\end{aligned} \quad (6)$$

3.2. Hyperbolic Space with Minkowksi Distance

We use the representation of $\mathbb{H}^p$ in $\mathbb{R}^{p+1}$ and $h(x) = x$. Denote $\chi = (-1,1,1,\ldots,1) \in \mathbb{R}^{n+1}$, $\langle \mathbf{u},\mathbf{v}\rangle_{\text{Mink}} = \langle \mathbf{u},\mathbf{v}\otimes\chi\rangle$, and $\|\mathbf{u}\|^2_{\text{Mink}} = \langle \mathbf{u},\mathbf{u}\rangle_{\text{Mink}}$. Here, $\otimes$ is the entry-wise product. The respective Euclidean gradient and hessian are expressed as follows.

$$\begin{aligned}
&\nabla_{\mathbf{y}^i} f(\mathbf{y}) \\
&= 2\beta \sum_{j\neq i} \frac{(G_{i,j}-\sigma)}{\|\mathbf{y}^i-\mathbf{y}^j\|_{\text{Mink}}\sqrt{1+\|\mathbf{y}^i-\mathbf{y}^j\|^2_{\text{Mink}}/4}} [(\mathbf{y}^i-\mathbf{y}^j)\otimes\chi].
\end{aligned} \quad (7)$$

$$\begin{aligned}
&\nabla\nabla_{\mathbf{y}^i} f(\mathbf{y})[V] \\
&= 2\beta^2 \sum_{j\neq i} \frac{\sigma(1-\sigma)\langle \mathbf{y}^i-\mathbf{y}^j, V^i-V^j\rangle_{\text{Mink}}}{\|\mathbf{y}^i-\mathbf{y}^j\|^2_{\text{Mink}}(1+\|\mathbf{y}^i-\mathbf{y}^j\|^2_{\text{Mink}}/4)} [(\mathbf{y}^i-\mathbf{y}^j)\otimes\chi] \\
&- 2\beta \sum_{j\neq i} \frac{(G_{i,j}-\sigma)\langle \mathbf{y}^i-\mathbf{y}^j, V^i-V^j\rangle_{\text{Mink}}}{\|\mathbf{y}^i-\mathbf{y}^j\|^3_{\text{Mink}}\sqrt{1+\|\mathbf{y}^i-\mathbf{y}^j\|^2_{\text{Mink}}/4}} [(\mathbf{y}^i-\mathbf{y}^j)\otimes\chi] \\
&- 2\beta \sum_{j\neq i} \frac{(G_{i,j}-\sigma)\langle \mathbf{y}^i-\mathbf{y}^j, V^i-V^j\rangle_{\text{Mink}}}{4\|\mathbf{y}^i-\mathbf{y}^j\|_{\text{Mink}}\sqrt{1+\|\mathbf{y}^i-\mathbf{y}^j\|^2_{\text{Mink}}/4}^3} [(\mathbf{y}^i-\mathbf{y}^j)\otimes\chi] \\
&+ 2\beta \sum_{j\neq i} \frac{(G_{i,j}-\sigma)}{\|\mathbf{y}^i-\mathbf{y}^j\|_{\text{Mink}}\sqrt{1+\|\mathbf{y}^i-\mathbf{y}^j\|^2_{\text{Mink}}/4}} [(V^i-V^j)\otimes\chi].
\end{aligned} \quad (8)$$

3.3. Tori With Manhattan Distance

Suppose that $\mathcal{M} = \mathbb{T}^p = (\mathbb{S}^1)^{\times p}$, that is a product of p circles of circumference 2π. We use again $h(x) = x$. Viewing each $\mathbf{y}^i$ as vector of p coordinates in $\mathbb{R}^2$, we compute

$$\nabla_{\mathbf{y}^i_u} f(\mathbf{y}) = -2\sum_{j\neq i} \frac{\beta}{\sqrt{1-\langle \mathbf{x}^i_u, \mathbf{x}^j_u\rangle^2}} \Big(G_{i,j} - \sigma \Big) \mathbf{y}^j_u. \quad (9)$$

$$\begin{aligned}
&\nabla\nabla_{\mathbf{y}^i_u} f(\mathbf{y})[V] = \\
&= 2\sum_{j\neq i} \left(\frac{\beta^2\sigma(1-\sigma)}{\sqrt{1-\langle \mathbf{y}^i_u, \mathbf{y}^j_u\rangle^2}} \sum_{w=1}^{p} \frac{\langle \mathbf{y}^i_u, V^j_u\rangle + \langle \mathbf{y}^j_u, V^i_u\rangle}{\sqrt{1-\langle \mathbf{y}^i_w, \mathbf{y}^j_w\rangle^2}} \right) \mathbf{y}^j_u \\
&- 2\sum_{j\neq i} \frac{\beta\langle \mathbf{y}^i_u, \mathbf{y}^j_u\rangle(\langle \mathbf{y}^i_u, V^j_u\rangle + \langle \mathbf{y}^j_u, V^i_u\rangle)}{(1-\langle \mathbf{y}^i_u, \mathbf{y}^j_u\rangle^2)^{3/2}} (G_{ij} - \sigma)\mathbf{y}^j_u \\
&- 2\sum_{j\neq i} \frac{\beta}{\sqrt{1-\langle \mathbf{x}^i_u, \mathbf{x}^j_u\rangle^2}} \Big(G_{i,j} - \sigma \Big) V^j_u.
\end{aligned} \quad (10)$$

Remark 1 Before turning to experiments, we discuss the case of missing data. Suppose that we only know that value of $\mathbf{G}$ for $(ij) \in \mathcal{E}$. Then, we simply perform the summation in Eq. (1) over pairs $(ij) \in \mathcal{E}$. Gradients and Hessians adapt accordingly.

4. Empirical Results

In our experiments, we focus on embedding samples from random geometric graphs. Specifically, the adjacency matrix is generated from a uniform distribution on $\mathbb{S}^{p-1}$ and $\mathbb{T}^p$. In the case of $\mathbb{H}^p$, we use the default random generator in `manopt`, which takes $(x_1,x_2,\ldots,x_p) \sim \mathcal{N}(0,I_p)$ and sets x_0 so that the defining equation of the Lorenzian formulation is satisfied. In all cases, the density parameter τ is chosen so that the expected density of the resulting random geometric graph is $1/2$. In the case of $\mathbb{S}^{p-1}$, this corresponds to $\tau = \sqrt{2}, \tau_h = 2$ (as $\|\mathbf{x}-\mathbf{y}\|_2 \leq \sqrt{2} \iff \langle \mathbf{x},\mathbf{y}\rangle \geq 0$). In the case of the torus, this corresponds to $\tau = \tau_h = \pi p/2$ as $d_{\mathbb{S}^1}(\mathbf{x}_u,\mathbf{y}_u) \sim \mathsf{Unif}([0,\pi])$ and the sum of p uniform r.v. is symmetric around its mean. In the case of $\mathbb{H}^p$, we find the value of τ corresponding to expected density $1/2$ empirically. We set the parameter β to be inversely proportional to $\mathbf{Var}(\mathbf{d}(\mathbf{x},\mathbf{y}))^{1/2}$ with respect to the latent

vector distributions, so that the order of $\beta(\mathbf{d}(\mathbf{x},\mathbf{y})-\tau)$ is roughly independent of dimension. In the case of large n, we recommend an inverse $(\log n)^{1/2}$ factor as the mximum of n subgaussian variables grows like $(\log n)^{1/2}$. Namely, $\beta \propto (\mathbf{Var}(\mathbf{d}(\mathbf{x},\mathbf{y}))\log n)^{-1/2}$. The code for all of the experiments is on GitHub.

4.1. Empirical Performance: Weak and Strong Embeddings

We fix $n = 100$ and range $p \in \{2,4,8,16,32,64\}$. All results are averaged over 20 iterations. We did not observe any significant difference in terms of the success of the algorithms with respect to first- and second- order methods. All experiments in the spherical and toric case use `steepestdescent`. In the hyperbolic case, we used `trustregions` as it was significantly faster. Our code on Github supports both `steepestdescent` and `trustregions` in all three cases.

4.1.1. Spherical Manifold

The performance on the weak embedding problem was better for larger dimensions, producing no error for $p \geq 16$ and insignificant errors for $p \in \{4,8\}$. In 5 of the 20 runs, the error for $p = 2$ was extremely large, but in the other 15 cases, again a satisfactory embedding was produced. The error for values above 64 was also zero and is not depicted here.

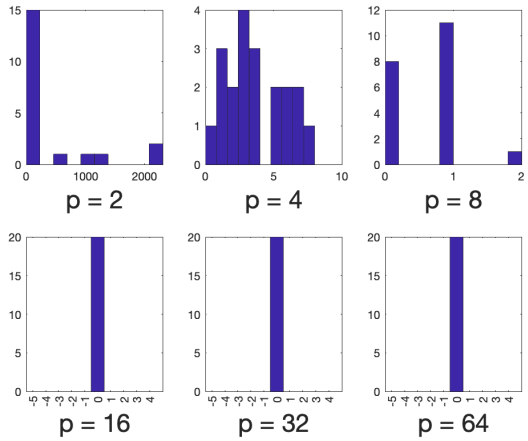

Figure 2: Number of incorrectly embedded edges out of 4950 versus number of iterations in spherical setting. Github.

However, the successful embedding in higher dimensions comes with a cost - the inferred embedding is significantly different from the original vectors which produced the respective adjacency matrix. More concretely, suppose that $\mathbf{G}$ is generated as $\mathbf{gg}(\mathbf{x}^1,\mathbf{x}^2,\ldots,\mathbf{x}^n)$ and $\mathbf{X} = (\mathbf{x}^1,\mathbf{x}^2,\ldots,\mathbf{x}^n)$. Let $\widehat{\mathbf{X}} = (\widehat{\mathbf{x}}^1,\widehat{\mathbf{x}}^2,\ldots,\widehat{\mathbf{x}}^n)$. As the dimension increases, $\widehat{\mathbf{X}}$ is increasingly different from $\mathbf{X}$.

We show this via the relative error of the respective distance matrices. Specifically, we consider the relative difference of distance matrices quantity. It is given by $\sum_{ij}(D_{ij}-\widehat{D_{ij}})^2/\mathbf{E}[\sum_{ij}(D_{ij}-\mathbf{E}D_{ij})^2]$, where D is the true distance matrix and $\widehat{D}$ is the estimated distance matrix. We compare the difference of distance matrices, normalized by the variance $\mathbf{E}[\sum_{ij}(D_{ij}-\mathbf{E}D_{ij})^2]$. This normalization captures the performance of always guessing the expected distance. It is needed especially in high dimension. Over the sphere, the distance between every two points is, with high probability, $\pi/2 \pm \tilde{O}(\sqrt{\log n/d})$, so just making a guess with small absolute error $\tilde{O}(\sqrt{\log n/d})$ is trivial.

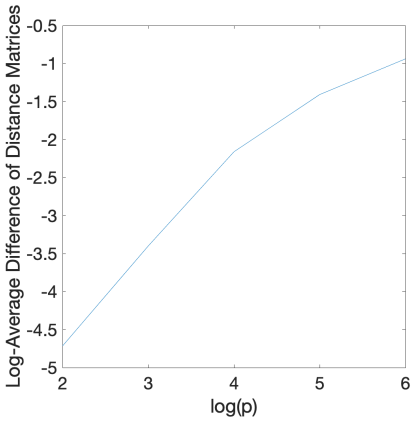

Figure 3: Log-log plot of dimension versus relative difference between distance matrices in spherical setting. Dimension 2 is excluded because of outliers. Github.

Overall, we observe the following trend: as the weak embedding problem becomes easier, the strong embedding problem becomes harder. This is natural - whenever there are more feasible embeddings (so the weak problem is easier), finding the correct one (i.e., solving the strong problem) becomes harder. We observe a similar phenomenon in the toric and hyperbolic settings.

4.1.2. Hyperbolic Manifold

Again, the performance with respect to the weak embedding problem improved with dimension.

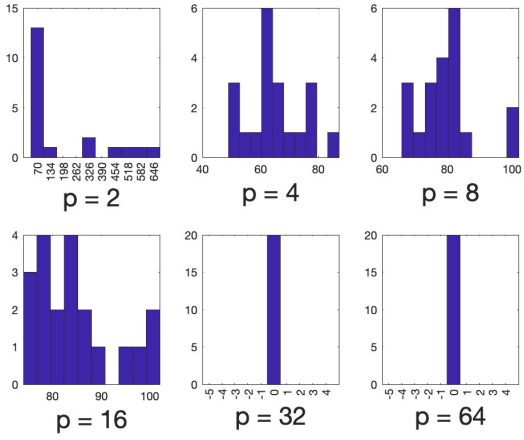

Figure 4: Number of incorrectly embedded edges out of 4950 versus number of iterations in hyperbolic setting. Github.

Similarly, the performance with respect to the strong embedding problem deteriorated with dimension.

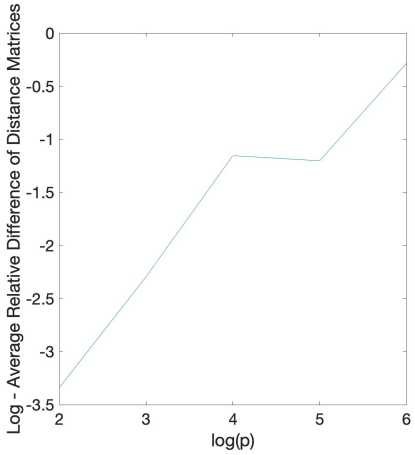

Figure 5: Log-log plot of dimension versus relative difference between distance matrices in hyperbolic setting. Dimension 2 is excluded because of outliers. Github.

4.1.3. Toric Manifold

The performance on the weak embedding problem was significantly worse in the toric setting for low dimensions than it was in the spherical and hyperbolic settings. Nevertheless, once a high-enough dimension was reached ($p = 32$), the performance on the weakembedding problem was perfect.

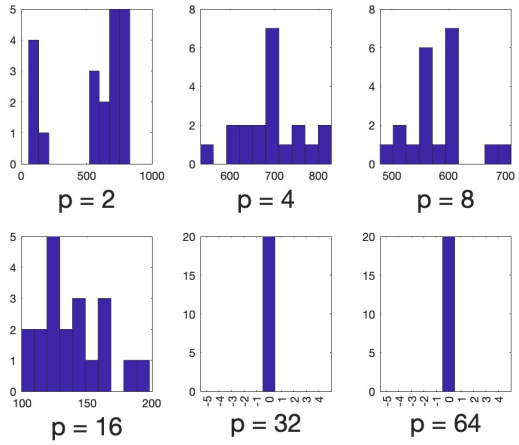

Figure 6: Number of incorrectly embedded edges out of 4950 versus number of iterations in troic setting. Github.

We did not observe any evidence that the strong embedding problem was solved by our approach in any regime. This is not surprising. Fig. 6 shows that in low dimension, the optimization approach does not solve even the weak embedding problem. However, drawing analogy from the Hyperbolic and Spherical settings, we expect that the strong embedding problem will only be solved in low dimension if solved at all.

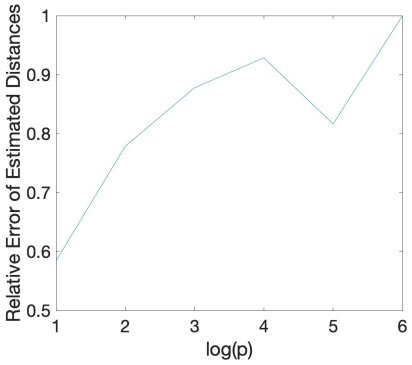

Figure 7: Log-dimension versus relative difference between distance matrices in the toric setting. Note that unlike in Figs. 3 and 5, the relative distance is not on the log scale. Github.

4.2. Applications

We focus on the spherical setting and consider the performance of our approach with respect to the different applications outlined in Section 1.1.

4.2.1. Estimating Geodesic Distances

In the low-dimensional regime when our approach solves the strong embedding problem, a good approximation of the original distance matrix was produced, with the error increasing with dimension. Below, we plot the ratio between the relative error in distance matrices averaged over 20 iterations for $p \in \{3,4,5,6,7,8,9,10\}$. Note that unlike the similar Figs. 3 and 5, this is *not* a log-log plot.

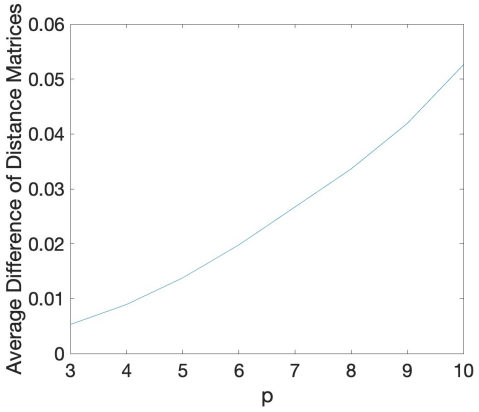

Figure 8: Dimension versus relative difference between distance matrices in spherical setting. Github.

4.2.2. Estimating Dimension

We performed experiments following the binary-search approach outlined in Section 1.1 with $n = 100, \mathsf{tol} = 1/n$ and $\kappa_{\mathsf{attempts}} =$

5.. Our experiments are in the low-dimensional setting $p \in \{2,3,\dots,15\}$ We did 10 iterations.

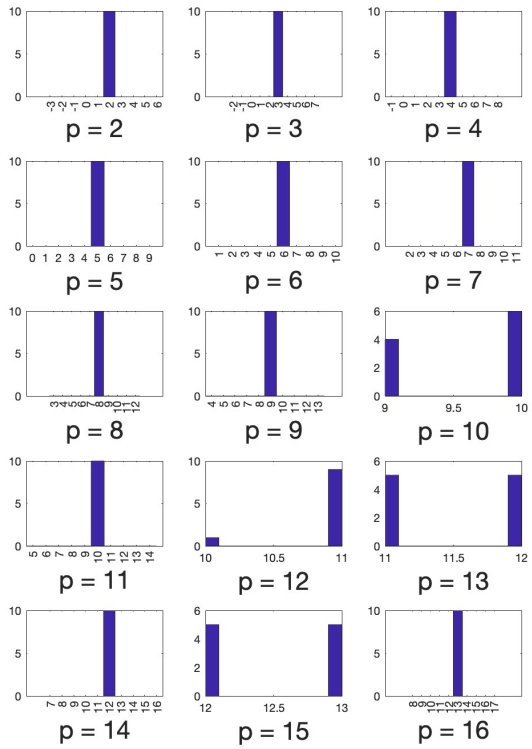

Figure 9: Estimated dimension versus number of iterations Github.

Our algorithm works perfectly for $p \leq 9$, but as p starts to increase, the dimensions are underestimated. This problem can be fixed for small $p \geq 10$ by decreasing tol. However, we will quickly run into the same issue. The reason is that in high enough dimension, it becomes easy to embed graphs even if they are "unlikely" to be generated in this dimension. Specifically, in Section 4.2.6 we empirically show that a sample from the Erdős-Rényi graph $G(n,1/2)$ can be embedded in dimension $p \approx n/3$. However, it is known [BDER14] that

$$d_{TV}\left(G(n,1/2), \mathsf{RGG}(n,\mathbb{S}^{p-1},\sqrt{2},\|\cdot\|_2,\mathsf{Unif})\right) = 1 - o_n(1)$$

whenever $p = o(n^3)$. This means that when $n/3 < p \ll n^3$, our algorithm will correctly embed a sample from $G(n,1/2)$, even though that sample is very unlikely to be generated from $\mathsf{RGG}(n,\mathbb{S}^{p-1},\sqrt{2},\|\cdot\|_2,\mathsf{Unif})$. This poses a fundamental limitation to using embedding algorithms for estimating the dimension of random geometric graphs.

4.2.3. Robustness 1: Missing Data

We performed experiments for $n = 100, p = 10$ in which we mask entries independently with a "mask rate" $q_m \in \{0, 0.033, 0.066, 0.1, \dots, 0.3\}$. We averaged the results over 10 independent iterations. We find that in this regime, our algorithm performs well on synthetic data even if a constant fraction of the entries are hidden Fig. 10.

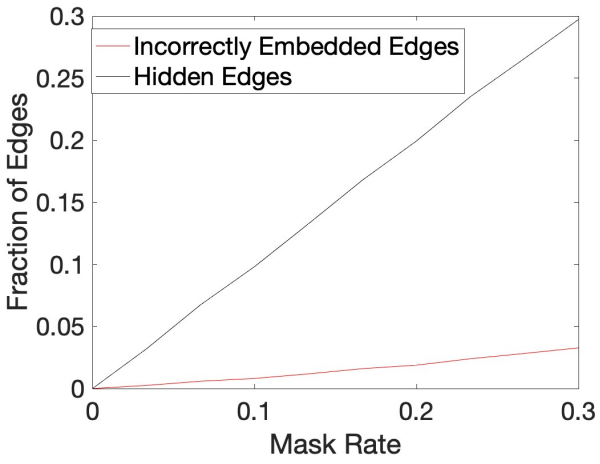

Figure 10: Hidden edges and incorrectly embedded edges as a fraction of all edges versus mask level. Github.

Note that simply filling the missing edges randomly will yield an approximately $q_m/2$ error, so our algorithm vastly outperforms a random guess.

4.2.4. Robustness 2: Noisy Data

We perform experiments in which we independently replace each edge with a uniform bit with "noise rate" $q_n \in \{0, 0.001, 0.002, \dots, 0.01\}$ for $n = 100, p = 10$. We find that in this regime, our algorithm leads to consistent, even if low, levels of reducing noise.

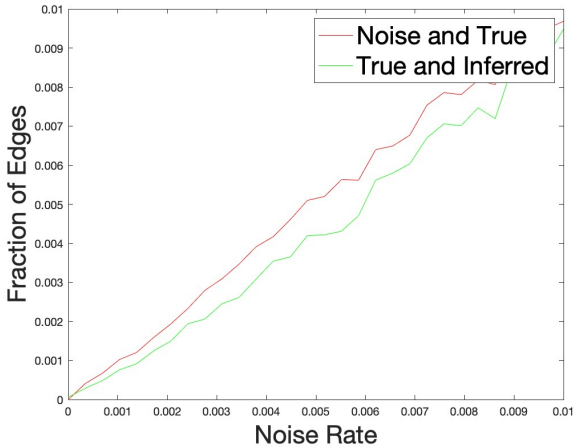

Figure 11: Number of corrupted edges in noisy graph and inferred graph versus noise level. Github.

4.2.5. Robustness 3: Distributional Robustness

One of the drawbacks of combinatorial embeddings is the fact that these methods rely heavily on the distribution $\mathcal{D}$ from which latent vectors in a RGG are generated. While the performance of our approach certainly depends on $\mathcal{D}$, we show that the same algorithm yields meaningful results under a wide range of distributions. We performed experiments analogous to the one in Fig. 1 with different distributions over the 320 cities. A full explanation of the sampling processes is given in Appendix B. In the embedded cities, we apply an optimal rotation for visualisation purposes. That is, if $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n) \in \mathbb{R}^{p \times n}$, $\widehat{\mathbf{X}} = (\widehat{\mathbf{x}^1}, \widehat{\mathbf{x}^2}, \dots, \widehat{\mathbf{x}^n}) \in \mathbb{R}^{p \times n}$,we plot $O\widehat{\mathbf{X}}$, where O is the rotation minimizing $\|\mathbf{X} - O\widehat{\mathbf{X}}\|_F$. It is well known that it is given by $O = UV^T$, where $U\Sigma V^T$ is the SVD decomposition of $\mathbf{X}\widehat{\mathbf{X}}^T$.

Uniformly Random Locations Which Are Cities.

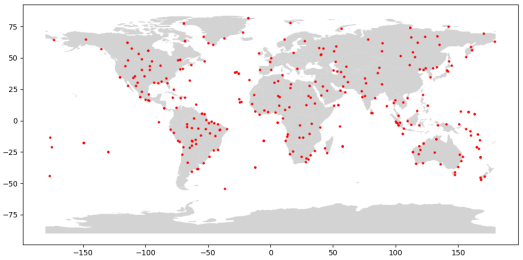

Figure 12: 320 Uniformly random locations which are cities.

The resulting embedding was

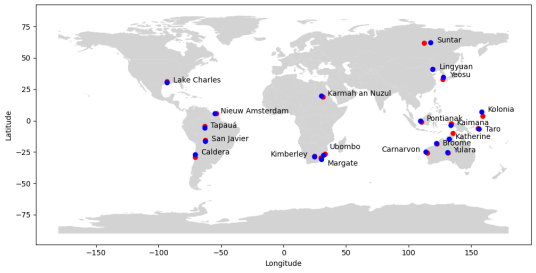

Figure 13: Embedding of first 20 cities.

Random Locations With Coastal Bias Which Are Cities.

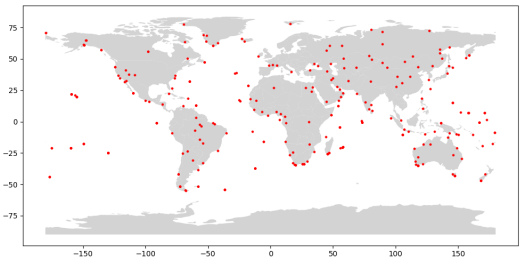

Figure 14: 320 Random city locations with coastal bias.

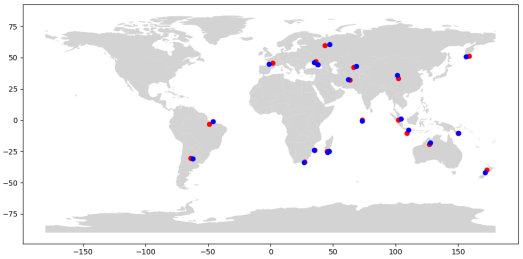

Figure 15: Embedding of first 20 cities with coastal bias after optimal rotation.

Uniformly Random Cities From Dataset.

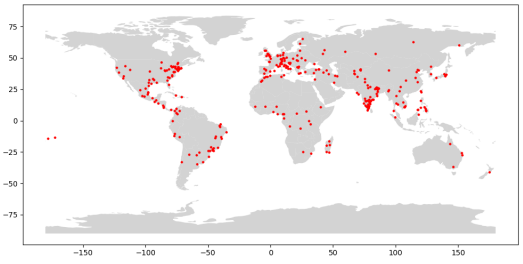

Figure 16: 320 Uniformly sampled cities from dataset.

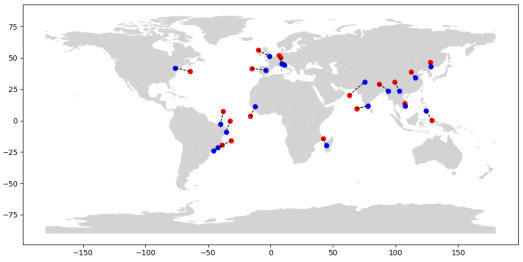

Figure 17: Embedding of first 20 cities sampled uniformly from the dataset after optimal rotation.

Admittedly, in the last case, when the data is furthest from uniformly distributed, the performance is the worst. Nevertheless, it must be acknowledged that in all three cases, an adequate embedding is produced by the *same* algorithm.

4.2.6. Embedding Typical Graphs

Finally, we demonstrate that our algorithm can also embed adjacency matrices that are not generated as geometric graphs. Again,we followed the simple binary-search procedure as in Section 4.2.2.

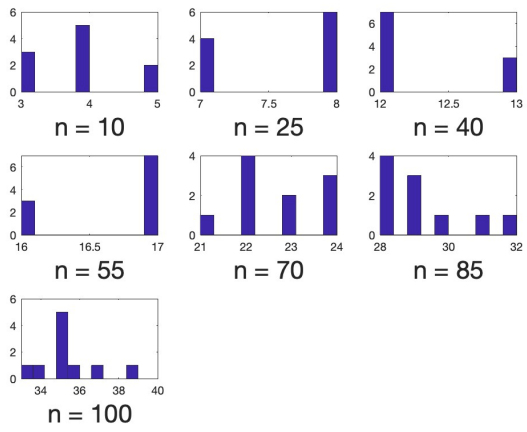

Figure 18: Minimal dimension p for which our optimization-based algorithm embedded perfectly a sample from $G(n,1/2)$. Performed are 10 iterations. Github.

We observe that our algorithm succcessfully embeds samples from $G(n,1/2)$ in $\mathbb{S}^{p-1}$ for $p \approx n/3$. To the best of our knowledge, no constructions for embedding $G(n,1/2)$ as a spherical geometric graph are known for $p \ll n$. When $p \geq Cn\sqrt{\log n}$ for some absolute constant C, one can show that the simple embedding $(\mathbf{x}^i)_{i=1}^p$ given by

$$\begin{aligned} \mathbf{x}_i^i &= \frac{1}{\sqrt{2}}, \\ \mathbf{x}_j^i &= \sqrt{\frac{2}{d-1}}(G_{ij} - 1/2) \text{ for } 1 \leq j \leq n, j \neq i \\ \mathbf{x}_k^i &= 0 \text{ for } k > n \end{aligned} \tag{11}$$

succeeds with high probability. Importantly, this suggests that solving the strong embedding problem is information-theoretically intractable when $p = \Omega(Cn\sqrt{\log n})$ as the latent-vector configurations described by Eq. (11) are already sufficient to embed nearly all graphs, but these configurations are just a small fraction of the possible configurations over $(\mathbb{S}^{p-1})^{\times n}$. This also implies that embedding-based algorithms are inadequate for estimating dimension and geodesic distances in this regime.

5. Future Directions

5.1. Landscape of The Logit Loss

We observe that, especially when the dimension p is large with respect to the number of observations n, gradient descent performs extremely well, oftentimes finding a perfect embedding Figs. 2 and 4. We are curious about the theoretical underpinnings of this phenomenon.

Question 2 For what regimes, if any, of p,n,τ,β is it true that all critical points of the logit loss Eq. (1) are global extrema with high probability? Even more ambitiously, when does the logit loss have only two critical points over $(\mathbb{S}^{p-1})^{\times n}/\mathbb{O}(p)$?

The second part of the question is on the quotient manifold $(\mathbb{S}^{p-1})/\mathbb{O}(p)$, where $\mathbb{O}(p)$ is the orthogonal group acting on $(\mathbb{S}^{p-1})$ by $\mathbf{O}(\mathbf{x}^1,\mathbf{x}^2,\dots,\mathbf{x}^n) = (\mathbf{O}\mathbf{x}^1,\mathbf{O}\mathbf{x}^2,\dots,\mathbf{O}\mathbf{x}^n)$ as the problem is clearly invariant under rotations. We note that we expect a positive answer to the second question *only* in low dimensions. As seen in Section 4.2.6, in high dimensions it is easy to find perfect embeddings, but these typically are very different from the latent vectors. The same question can be asked in the toric and hyperbolic settings.

5.2. The Blessing of Dimensionality for Weak Embeddings

The performance of our algorithms on the weak embedding problem improves in terms of the correctly classified edges as we increase the dimension for all of the $\mathbb{S}^{p-1}$, $\mathbb{H}^p$, and $\mathbb{T}^p$ models Figs. 2 and 6. This leads to the following question.

Question 3 For what regimes, if any, of p,n,τ,β is there an efficient algorithm that finds a perfect embedding of a geometric graph? What if we only require a perfect embedding with high probability over a random geometric graph distribution? What about with high probability over the uniform Erdős-Rényi distribution (see Section 4.2.6)?

The construction in Eq. (11) for $(\mathcal{M},\|\cdot\|_2,\sqrt{2})$ give a positive answer to the worst-case question when $p = \Omega(n^2)$ and to the average case question when $p = \tilde{\Omega}(n)$. Similarly, we ask.

Question 4 For what regimes, if any, of p,n,τ,β is there an efficient algorithm that solves the Geometric Graph Recognition problem (recall Section 1.2)?

Again, we know that the problem is hard in 2 and 3 dimensions [BK98], but becomes easy in the regime $p = \Omega(n^2)$, simply because any graph can be embedded as a geometric graph in $\mathbb{S}^{\Omega(n^2)}$ using the same strategy as in Eq. (11).

5.3. Why Is Strong Embedding in the Toric Case Harder?

While our algorithms work well in low dimension for $\mathbb{S}^{p-1}, \mathbb{H}^p$, solving both the the weak and strong embedding problems satisfactorily, this is not the case for $\mathbb{T}^p$. We leave it as an open problem why this is the case. One potential reason could be the fact that this is the only model in which $d(\cdot,\cdot)$ is not (a transformation of) the intrinsic distance

5.4. Computational Savings

Finally, there are many potential ways to improve the running time of our algorithms. First, we can use the symmetries of $\mathcal{M}$ so that we optimize over a (potentially smaller) manifold. Specifically, in the case of $\mathcal{M} = \mathbb{S}^{p-1}$, we can use the quotient $(\mathbb{S}^{p-1})^{\times n}/\mathbb{O}(p)$ as already discussed. First-order methods are identical over quotient and non-quotient manifolds, but second order methods can benefit from exploiting the quotient structure in practice [Bou23, Chapter 9]. Second, one can split the vertices of the graph into patches, embed each patch individually, and then synchronize the patches using the symmetries of the underlying space $(\mathcal{M}, d(\cdot,\cdot))$. This approach has proven to be extremely powerful in the closely related problem of network sensing localization [GK04, CLS12].

References

[AMS22] ALMAGRO P. B., M. & SERRANO M.: Detecting the ultra low dimensionality of real networks. *Nature Communications* (2022). URL: https://doi.org/10.1038/s41467-022-33685-z. 2

[AVL12] ALAMGIR M., VON LUXBURG U.: Shortest path distance in random k-nearest neighbor graphs. In *Proceedings of the 29th International Coference on International Conference on Machine Learning* (Madison, WI, USA, 2012), ICML'12, Omnipress, p. 1251–1258. 2

[BDER14] BUBECK S., DING J., ELDAN R., RÁCZ M.: Testing for high-dimensional geometry in random graphs. *Random Structures & Algorithms 49* (11 2014). doi:10.1002/rsa.20633. 2, 3, 8

[BK98] BREU H., KIRKPATRICK D. G.: Unit disk graph recognition is np-hard. *Computational Geometry 9*, 1 (1998), 3–24. Special Issue on Geometric Representations of Graphs. URL: https://www.sciencedirect.com/science/article/pii/S092577219700014X, doi:https://doi.org/10.1016/S0925-7721(97)00014-X. 3, 11

[BMAS14] BOUMAL N., MISHRA B., ABSIL P.-A., SEPULCHRE R.: Manopt, a Matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research 15*, 42 (2014), 1455–1459. URL: https://www.manopt.org. 3, 5

[Bou23] BOUMAL N.: *An Introduction to Optimization on Smooth Manifolds*. Cambridge University Press, 2023. doi:DOI:10.1017/9781009166164. 4, 11

[CLS12] CUCURINGU M., LIPMAN Y., SINGER A.: Sensor network localization by eigenvector synchronization over the euclidean group. *ACM transactions on sensor networks 8* (07 2012). doi:10.1145/2240092.2240093. 3, 11

[DdC22] DUCHEMIN Q., DE CASTRO Y.: Random geometric graph: Some recent developments and perspectives, 2022. arXiv:2203.15351. 2, 3

[DDHM22] DANI V., DÍAZ J., HAYES T. P., MOORE C.: Improved Reconstruction of Random Geometric Graphs. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)* (Dagstuhl, Germany, 2022), Bojańczyk M., Merelli E., Woodruff D. P., (Eds.), vol. 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, pp. 48:1–48:17. URL: https://drops.dagstuhl.de/opus/volltexte/2022/16389, doi:10.4230/LIPIcs.ICALP.2022.48. 1, 2, 3, 4

[DGLU11] DEVROYE L., GYÖRGY A., LUGOSI G., UDINA F.: High-Dimensional Random Geometric Graphs and their Clique Number. *Electronic Journal of Probability 16* (2011), 2481 – 2508. URL: https://doi.org/10.1214/EJP.v16-967. 2, 3

[DMM20] DÍAZ J., MCDIARMID C., MITSCHE D.: Learning random points from geometric graphs or orderings. *Random Structures & Algorithms 57* (09 2020). doi:10.1002/rsa.20922. 4

[ES16] ESTRADA E., SHEERIN M.: Consensus dynamics on random rectangular graphs. *Physica D: Nonlinear Phenomena 323-324* (2016), 20–26. Nonlinear Dynamics on Interconnected Networks. URL: https://www.sciencedirect.com/science/article/pii/S0167278915002171, doi:https://doi.org/10.1016/j.physd.2015.10.021. 2

[FGKS23a] FRIEDRICH T., GÖBEL A., KATZMANN M., SCHILLER L.: Cliques in high-dimensional geometric inhomogeneous random graphs, 2023. arXiv:2302.04113. 3

[FGKS23b] FRIEDRICH T., GÖBEL A., KATZMANN M., SCHILLER L.: A simple statistic for determining the dimensionality of complex networks, 2023. arXiv:2302.06357. 2, 3

[FX06] FERREIRA R., XAVIER J. M. F.: Hessian of the riemannian squared distance function on connected locally symmetric spaces with applications. 5

[Gam19] GAMARNIK D.: Explicit construction of rip matrices is ramsey-hard. *Communications on Pure and Applied Mathematics 73* (11 2019). doi:10.1002/cpa.21873. 2

[GK04] GOTSMAN C., KOREN Y.: Distributed graph layout for sensor networks. In *J. Graph Algorithms Appl.* (2004). 3, 11

[HAB*09] HAENGGI M., ANDREWS J. G., BACCELLI F., DOUSSE O., FRANCESCHETTI M.: Stochastic geometry and random graphs for the analysis and design of wireless networks. *IEEE Journal on Selected Areas in Communications 27*, 7 (2009), 1029–1046. doi:10.1109/JSAC.2009.090902. 2

[HRP08] HIGHAM D. J., RAŠAJSKI M., PRŽULJ N.: Fitting a geometric graph to a protein–protein interaction network. *Bioinformatics 24*, 8 (03 2008), 1093–1099. URL: https://doi.org/10.1093/bioinformatics/btn079, arXiv:https://academic.oup.com/bioinformatics/article-pdf/24/8/1093/49046772/bioinformatics_24_8_1093.pdf, doi:10.1093/bioinformatics/btn079. 2

[NK18] NICKEL M., KIELA D.: Learning continuous hierarchies in the lorentz model of hyperbolic geometry. *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.* (2018). 3

[OMF20] O'CONNOR L., MÉDARD M., FEIZI S.: Maximum likelihood embedding of logistic random dot product graphs. *Proceedings of the AAAI Conference on Artificial Intelligence 34* (04 2020), 5289–5297. doi:10.1609/aaai.v34i04.5975. 1, 2, 4

[PJ09] PRECIADO V. M., JADBABAIE A.: Spectral analysis of virus spreading in random geometric networks. *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference* (2009), 4802–4807. 2

[Sax79] SAXE J.: Embeddability of weighted graphs in k-space is strongly np-hard. URL: https://users.cs.duke.edu/~brd/Teaching/Bio/asmb/current/Readings3/saxe-embeddability.pdf. 3

[SSH18] SOLOVEY K., SALZMAN O., HALPERIN D.: New perspective on sampling-based motion planning via random geometric graphs. *The International Journal of Robotics Research 37*, 10 (2018), 1117–1133. URL: https://doi.org/10.1177/0278364918802957, arXiv:https://doi.org/10.1177/0278364918802957, doi:10.1177/0278364918802957. 2

[WL18] WILSON B., LEIMEISTER M.: Gradient descent in hyperbolic space, 2018. arXiv:1805.08207v2. 5

[XOLL16] XIE Z., OUYANG Z., LIU Q., LI J.: A geometric graph model for citation networks of exponentially growing scientific papers. *Physica A: Statistical Mechanics and its Applications 456* (2016), 167–175. URL: https://www.sciencedirect.com/science/article/pii/S0378437116300164, doi:https://doi.org/10.1016/j.physa.2016.03.018. 2

[Yem79] YEMINI Y.: Some theoretical aspects of position-location problems. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)* (1979), pp. 1–8. doi:10.1109/SFCS.1979.39. 3

Appendix A: Numerical Verification of Derivatives

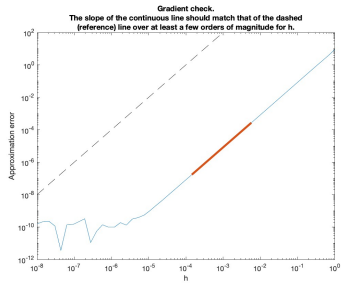

Figure 19: Numerical gradient check in spherical case.

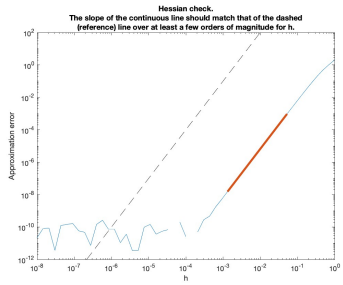

Figure 20: Numerical Hessian check in spherical case.

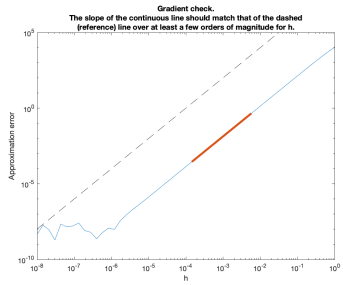

Figure 21: Numerical gradient check in hyperbolic case.

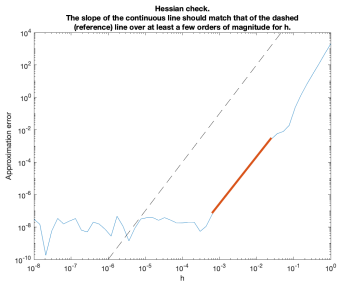

Figure 22: Numerical Hessian check in hyperbolic case.

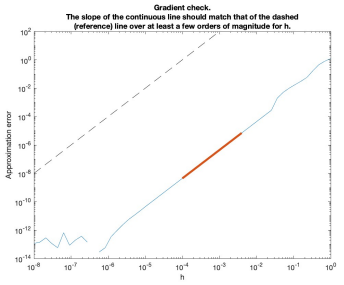

Figure 23: Numerical gradient check in toric case.

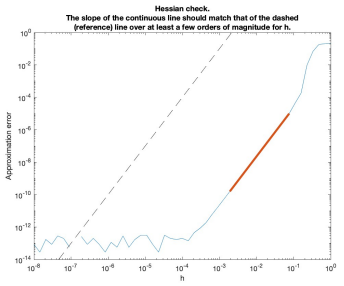

Figure 24: Numerical Hessian check in toric case.

Appendix B: Experiment With City Data

To provide more intuition and visualize the performance of our algorithm, we generated a dataset with random cities around the world. The idea was to produce a set of cities which approximately uniformly spread on land. Our sampling process was the following (code on GitHub):

1. We started with the free city data base https://simplemaps.com/data/world-cities.
2. Sampled 1000 random points $\mathbf{x}^1, \dots, \mathbf{x}^{1000}$ around the globe and mapped each $\mathbf{x}^i$ to its closest (radially) city $\mathbf{y}^i$ in the database.
3. Removed samples i for which the radial distance was more than $0.14rad$ (corresponding to $\langle \mathbf{x}, \mathbf{y} \rangle \leq 0.99$). This step was performed to remove the "coastal bias." Without it, we are more likely to sample from coastal cities as they are more likely to be closer to points $\mathbf{x}^i$ sampled in the ocean. This reduced the dataset to 611 datapoints. We kept the first 320.

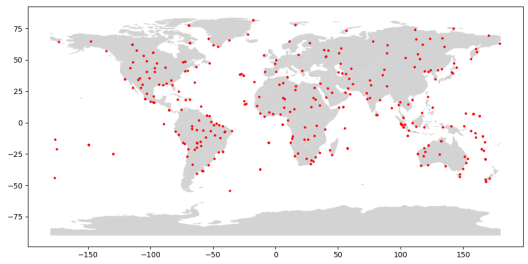

Figure 25: 320 Cities with removed coastal bias.

To test our methods we also performed the analogous experiment with other city distributions.

320 Cities Without Coastal Bias Removed. When we do not debias, a lot more of the cities are on the coast.

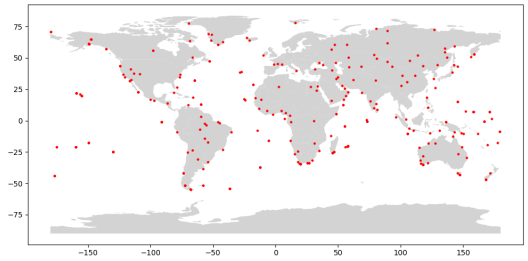

Figure 26: 320 Cities without removed coastal bias.

320 Uniformly Random Cities From DataSet. We also sample 320 random points from the dataset. They are much more concentrated around Europe, the US coasts and South-East Asia.

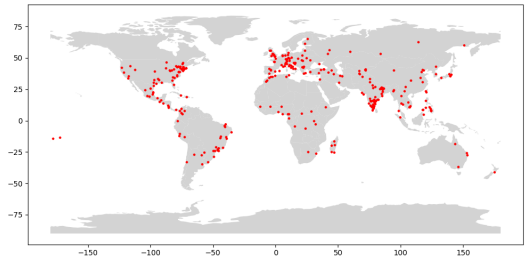

Figure 27: 320 Uniformly sampled cities from dataset.