Metric Learning and Manifolds: Preserving the Intrinsic Geometry

Dominique Perrault-Joncas DCPJ@STAT.WASHINGTON.EDU
Department of Statistics
University of Washington
Seattle, WA 98195-4322, USA

Marina Meilă MMP@STAT.WASHINGTON.EDU
Department of Statistics
University of Washington
Seattle, WA 98195-4322, USA

Editor:

Abstract

A variety of algorithms exist for performing non-linear dimension reduction, but these algorithms do not preserve the original geometry of the data except in special cases. In general, in the low-dimensional representations obtained, distances are distorted, as well as angles, areas, etc. This paper proposes a generic method to estimate the distortion incurred at each point of an embedding, and subsequently to "correct" distances and other intrinsic geometric quantities back to their original values (up to sampling noise).

Our approach is based on augmenting the output of an embedding algorithm with geometric information embodied in the Riemannian metric of the manifold. The Riemannian metric allows one to compute geometric quantities (such as angle, length, or volume) for any coordinate system or embedding of the manifold. In this work, we provide an algorithm for estimating the Riemannian metric from data, consider its consistency, and demonstrate the uses of our approach in a variety of examples.

1. Introduction

When working with high-dimensional data, one is regularly confronted with the problem of tractability and interpretability of the data. An appealing approach to this problem is dimension reduction: finding a low-dimensional representation of the data that preserves all or most of the important "information". One popular idea for data consisting of vectors in $\mathbb{R}^r$ is to appeal to the manifold hypothesis, whereby the data is assumed to lie on a low-dimensional smooth manifold embedded in the high dimensional space. The task then becomes to recover the low-dimensional manifold so as to perform any statistical analysis on the lower dimensional representation of the data. This task is known as *manifold learning* and its popularity is due to its ability to produce non-linear mappings to low dimensions by algorithms that are interpretable and mathematically elegant.

Some of these algorithms, aiming to recover the geometry of the low-dimensional manifold $\mathcal{M}$ using either local or global features of the data, will be described in the next section. Notwithstanding their widespread use, they have important shortcomings that have been documented in the literature (Goldberg et al., 2008; Wittman, 2005). The most important criticism is that the algorithms fail to recover the geometry of the manifold in many instances. A second is that no coherent framework yet exists in which the multitude of existing algorithms can easily be compared, as each algorithm's output lies in a different coordinate system. This paper directly addresses the first of these shortcomings; in the Experiments and Discussion section we show how it implicitly solves the latter as well.

It is customary to evaluate *embedding algorithms* (a.k.a manifold learning algorithms) by how well they "preserve the geometry", and much effort has been devoted to finding embedding algorithms that do so. We give this criterion a mathematical interpretation, using the concepts of *Riemannian metric* and *isometry*. The criticisms noted above reflect the fact that the majority of manifold learning algorithms output embeddings that are not isometric to the original data except in special cases.

Assuming that preserving the geometry of the data is an important goal, we offer a new perspective: rather than contributing yet another embedding algorithm that strives to achieve isometry, we provide a way to augment *any* reasonable embedding so as to allow for the correct computation of geometric values of interest in the embedding's own coordinates.

The information necessary for reconstructing the geometry of the manifold is embodied in its Riemannian metric, defined in Section 4. We propose to recover a *Riemannian manifold* $(\mathcal{M}, g)$ from the data, that is, a manifold and its Riemannian metric g, and express g in any desired coordinate system. Practically, for any given mapping produced by an existing manifold learning algorithm, we will add an estimation of the Riemannian metric g in the new data coordinates, that makes the geometrical quantities like distances and angles of the mapped data (approximately) equal to their original values, in the raw data. Our paradigm is illustrated schematically by Figure 1.

We start with a brief discussion of the literature and an introduction to the Riemannian metric in Sections 2 and 3. The core of our paper is the demonstration of how to obtain the Riemannian metric from the mathematical, algorithmic and statistical points of view. These are presented in Sections 4, 5 and 6 respectively. Finally, we offer some examples and applications in Section 7 and conclude with a discussion in Section ??. *MMP: statistical–tone that down; Metric ML – introduce the name*

2. The Task of Manifold Learning

In this section, we present the problem of manifold learning. While this represents background material, we intentionally proceed slowly in order to formulate coherently and explicitly a number of properties that cause a manifold learning algorithm to "work well", or have intuitively desirable properties.

The first desirable property is that the algorithm produces a *smooth* map, and Section 3 defines this concept in differential geometry terms. This property is common to a large number of algorithms, so it will be treated as an assumption in later sections.

The second desirable property is *consistency*. Existing consistency results will be discussed briefly in Section 2.3 below, and then a more detailed framework will be presented in Section 6.

The third property is the preservation of the *intrinsic geometry* of the manifold. This property is of central interest to our paper.

We begin our survey of manifold learning algorithms by discussing a well-known method for linear dimensionality reduction: Principal Components Analysis (PCA). PCA projects data onto the linear subspace of fixed dimension that captures most of the variability in the data. It does so by performing an eigendecomposition of the data covariance matrix and selecting the eigenvectors with the largest eigenvalues, i.e. those that explain the most variation. Since the projection of the data is linear by construction, PCA cannot recover any curvature present in the data.

In contrast to linear techniques, manifold learning algorithms assume that the data lies near or on a non-linear, smooth, surface of dimension d called the *data manifold* $\mathcal{M}$, contained in the original high-dimensional space $\mathbb{R}^r$ with $d \ll r$, and attempt to uncover $\mathcal{M}$. If they succeed in doing so, then each high-dimensional observation can accurately be described by a small number of parameters, its *embedding coordinates* $f(p)$ for all $p \in \mathcal{M}$.

Thus, a *manifold learning* or *manifold embedding* algorithm takes as input a set of points $\mathcal{D} = \{p_1, \dots p_n\} \subset \mathbb{R}^r$, where r is typically high. The algorithms maps these into vectors $\{f(p_1), \dots f(p_n)\} \subset$

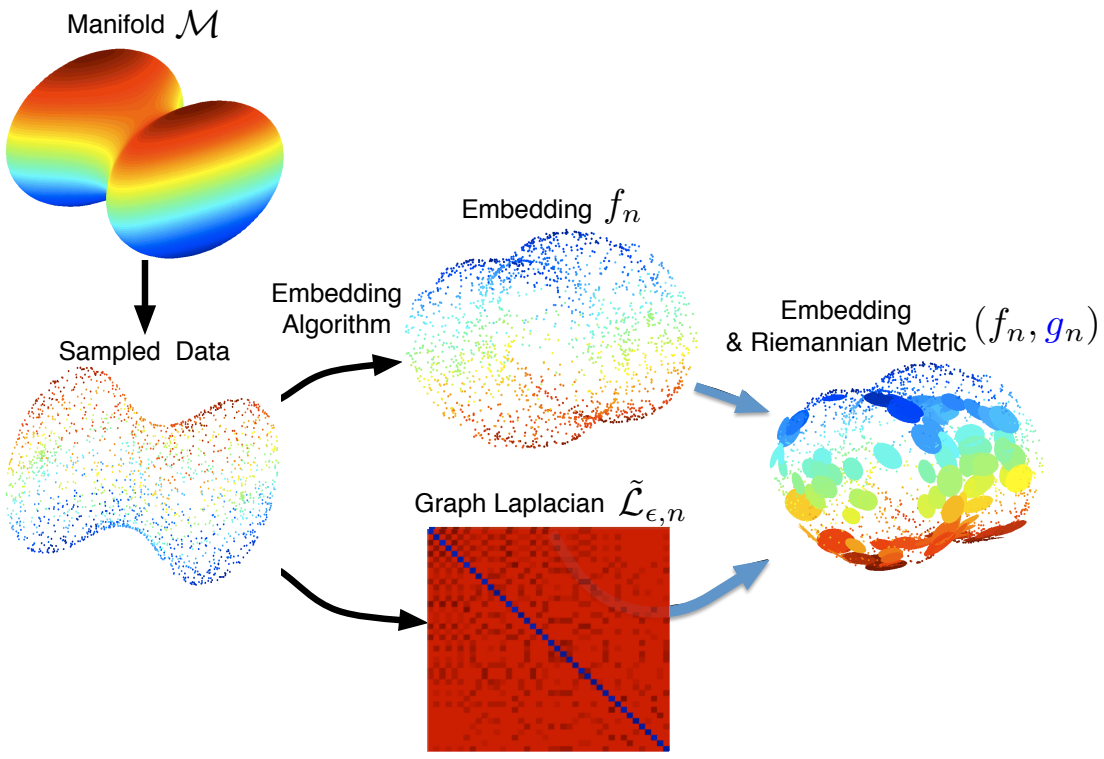


Figure 1: Schematic of our paradigm for Metric Manifold Learning. The black arrows represent existing algorithms. The blue represent the contribution of this paper. We assume the observed data is sampled i.i.d. from an unknown manifold $\mathcal{M}$; one can recover the manifold using one of the existing manifold learning algorithms (in the present case, Isomap was used). The recovery is correct topologically, but distorts distances and angles (here, the hourglass is compressed lengthwise) w.r.t the original data. We use the fact, well-known in mathematics but not exploited before in machine learning, that the manifold Laplacian (here estimated by the graph Laplacian $\tilde{L}_{n,\epsilon}$) captures the intrisic data geometry, and allows us to represent it in any given coordinates. This is what Algorithm MetricML in Section **??** does, using both the embedding f_n and the Laplacian $\tilde{L}_{n,\epsilon}$. The rightmost panel shows the algorithm's output; this is the *pushforward (Riemannian) metric* (defined in Section **??**). It associates an ellipsoid to each embedded point. The radius of the ellipsoid in a given direction measures the amount of rescaling one must apply so as to preserve the distances of the original data. To note that the ellipses are rounder on the "top" of the hourglass, but elongated on the "sides"; this elongation compensates for the foreshortening of the Isomap embedding. Note that Isomap only produces very mild deformations on this data set.

$\mathbb{R}^s$, with $s \ll r$ and $d \leq s$. This terminology, as well as other differential geometry terms used in this section, will later be defined formally. *MMP: remove this?*

2.1 Existing Algorithms

We start by presenting, as examples to which we will refer later, a few of the many non-linear dimension reduction algorithms based on the manifold approach.

Many manifold learning algorithm, including the ones described in this section, preprocess the data by first constructing a *neighborhood graph* $\mathcal{G}$. That is, for each data point p, they find a neighborhood of radius ϵ, and connect p with all other points p' in its neighborhood. Other algorithms construct $\mathcal{G}$ by connecting the k nearest neighbors of each point. *MMP: connecting each point with its k* Both k and ϵ act as smoothing, or bandwidth, parameters of the embedding algorithm that significantly affect its attributes. Methods for choosing ϵ have been studied by (Singer, 2006), and we rely on them in our implementation.

The first category of algorithms uses only local information, embodied in $\mathcal{G}$, to construct the embedding.

- LE: The Laplacian Eigenmap is based on the random walk graph Laplacian, henceforth referred to as graph Laplacian, defined formally in Section 5 below. The graph Laplacian is used because its eigendecomposition can be shown to preserve local distances while maximizing the smoothness of the embedding. Thus, the LE embedding is obtained simply by keeping the first s eigenvectors of the graph Laplacian in order of ascending eigenvalues. The first eigenvector is omitted, since it is necessarily constant and hence non-informative.

- DM: The Diffusion Map is a variation of the LE that emphasizes the deep connection between the graph Laplacian and heat diffusion on manifolds. The central idea remains to embed the data using an eigendecomposition of the graph Laplacian. However, DM defines an entire family of graph Laplacians, all of which correspond to different diffusion processes on $\mathcal{M}$ in the continuous limit. Thus, the DM can be used to construct a graph Laplacian whose asymptotic limit is the Laplace-Beltrami operator, defined in (4), independently of the sampling distribution of the data. This is the most important aspect of DM for our purposes.

- LTSA: The Linear Tangent Space Alignment algorithm, as its name implies, is based on estimating the tangent planes of the manifold $\mathcal{M}$ at each point in the data set using the k-nearest neighborhood graph $\mathcal{G}$ as a window to decide which points should be used in evaluating the tangent plane. This estimation is acheived by performing a singular value decomposition of the data matrix for the neighborhoods, which offers a low-dimensional parameterization of the tangent planes. The tangent planes are then pieced together so as to minimize the reconstruction error, and this defines a global low-dimensional parametrization of the manifold provided it can be embedded in $\mathbb{R}^d$. One aspect of the LTSA is worth mentioning here even though we will not make use of it: by obtaining a parameterization of all the tangent planes, LTSA effectively obtains the Jacobian between the manifold and the embedding at each point. This provides a natural way to move between the embedding $f(\mathcal{M})$ and $\mathcal{M}$. This desirable property does not hold in general: the inverse map for out-of-sample points is not easy to infer.

- MVU: Maximum Variance Unfolding (also known as Semi-Definite Embedding) Weinberger and Saul (2006) represents the input and output data in terms of Gram matrices. The idea is to maximize the output variance, subject to exactly preserving the distances between neighbors. This objective can be expressed as a semi-definite program.

- LLE: Local Linear Embedding is one of the algorithms that constructs $\mathcal{G}$ by connecting the k nearest neighbors of each point. In addition, it assumes that the data is linear in each neighborhood $\mathcal{G}$, which means that any point p can be approximated by a weighted average

of its neighbors. The algorithm finds weights that minimize the cost of representing the point by its neighbors under the L_2-norm. Then, the lower dimensional representation of the data is achieved by a map of a fixed dimension that minimizes the cost, again under the L_2-norm, of representing the mapped points by their neighbors using the weights found in the first step.

Another approach is to use global information to construct the embedding, and the foremost example in this category is Isomap (Tenenbaum et al., 2000). Isomap estimates the shortest path in the neighborhood graph $\mathcal{G}$ between every pair of data points p, p', then uses the Euclidean Multidimensional Scaling (MDS) algorithm (Borg and Groenen, 2005) to embed the points in d dimensions with minimum distance distortion all at once. Thus, even though Isomap uses the linear MDS algorithm to embed the data, it is able to account for the non-linear nature of the manifold by applying MDS to the minimizing geodesics.

2.2 Manifolds, Coordinate Charts and Smooth Embeddings

Now that we have explained the task of manifold learning in general terms and presented the most common embedding algorithms, we focus on formally defining manifolds, coordinate charts and smooth embeddings. These formal definitions set the foundation for the methods we will introduce in Sections 3 and 4 .

We first consider the geometric problem of manifold and metric representation, and define a smooth manifold in terms of coordinate charts.

Definition 1 (Smooth Manifold (Hein et al., 2007)) *A d-dimensional manifold $\mathcal{M}$ is a topological (Hausdorff) space such that every point has a neighborhood homeomorphic to an open subset of $\mathbb{R}^d$. A chart (U, x), or coordinate chart, of manifold $\mathcal{M}$ is an open set $U \subset \mathcal{M}$ together with a homeomorphism $x : U \to V$ of U onto an open subset $V \subset \mathbb{R}^d = \{(x^1, ..., x^d) \in \mathbb{R}^d\}$. A C^∞-Atlas $\mathcal{A}$ is a collection of charts,*

$$\mathcal{A} \equiv \cup_{\alpha \in I} \{(U_\alpha, x_\alpha)\}\,,$$

where I is an index set, such that $\mathcal{M} = \cup_{\alpha \in I} U_\alpha$ and for any $\alpha, \beta \in I$ the corresponding transition map,

$$x_\beta \circ x_\alpha^{-1} : x_\alpha(U_\alpha \cap U_\beta) \to \mathbb{R}^d\,, \tag{1}$$

is continuously differentiable any number of times.

For simplicity, we assume throughout that the manifold is smooth, but for the purposes of this paper, it is sufficient to have a $\mathcal{C}^2$ manifold. Following (Lee, 2003), we will identify local coordinates of an open set $U \subset \mathcal{M}$ by the image coordinate chart homeomorphism. That is, we will identify U by $x(U)$ and the coordinates of point $p \in U$ by $x(p) = (x^1(p), ..., x^d(p))$.

This definition allows us to express the goal of manifold learning in a formal way: assuming that our (high-dimensional) data set $\mathcal{D} = \{p_1, \dots p_n\} \subset \mathbb{R}^r$ comes from a smooth manifold with low d, the goal of manifold learning is to find a corresponding collection of d-dimensional coordinate charts for these data.

The definition also hints at two other well-known facts. First, the coordinate chart(s) are not uniquely defined, and there are infinitely many atlases for the same manifold $\mathcal{M}$ (Lee, 2003). Thus, it is not obvious from coordinates alone, if two atlases represent the same manifold or not. In particular, to compare the outputs of a manifold learning algorithm with the original data, or with the result of another algorithm on the same data, one must resort to *intrinsic*, coordinate independent quantities.

The second fact is that a manifold cannot be represented in general by a *global* coordinate chart. For instance, the sphere is a 2-dimensional manifold that cannot be mapped homeomorphically to $\mathbb{R}^2$; one needs at least two coordinate charts to cover the 2-sphere. It is also evident that the sphere is naturally embedded in $\mathbb{R}^3$.

One can in general circumvent the need for multiple charts, by mapping the data into $s > d$ dimensions as in this example. Mathematically, the grounds for this important fact are centered on the concept of *embedding*, which we introduce next.

Let $\mathcal{M}$ and $\mathcal{N}$ be two manifolds, and $f : \mathcal{M} \to \mathcal{N}$ be a C^∞ (i.e *smooth*) map between them. Then, at each point $p \in \mathcal{M}$, the Jacobian $df(p)$ of f at p defines a linear mapping between the tangent plane to $\mathcal{M}$ at p, denoted $T_p(\mathcal{M})$, and the tangent plane to $\mathcal{N}$ at $f(p)$, denoted $T_{f(p)}(\mathcal{N})$.

Definition 2 (Rank of a Smooth Map) *A smooth map $f : \mathcal{M} \to \mathcal{N}$ has rank k if the Jacobian $df(p) : T_p\mathcal{M} \to T_{f(p)}\mathcal{N}$ of the map has rank k for all points $p \in \mathcal{M}$. Then we write $rank\,(f) = k$.*

Definition 3 (Embedding) *Let $\mathcal{M}$ and $\mathcal{N}$ be smooth manifolds and let $f : \mathcal{M} \to \mathcal{N}$ be a smooth injective map, that is $rank(f) = dim(\mathcal{M})$, then f is called an immersion. If $\mathcal{M}$ is homeomorphic to its image under f, then f is an embedding of $\mathcal{M}$ into $\mathcal{N}$.*

Whitney's Embedding Theorem (Lee, 1997) states that any d-dimensional smooth manifold can be embedded into $\mathbb{R}^{2d}$. It follows from this fundamental result that if the *intrinsic dimension* d of the data manifold is small compared to the observed data dimension r, then very significant dimension reductions can be achieved, namely from r to $s \leq 2d$ with a single map $f : \mathcal{M} \to \mathbb{R}^s$.

Whitney's result is tight, in the sense that some manifolds, such as real projective spaces, need all $2d$ dimensions. However, the $r = 2d$ upper bound is probably pessimistic for most datasets. Even so, the important point is that the existence of an embedding of $\mathcal{M}$ into $\mathbb{R}^d$ cannot be relied upon; at the same time, finding the optimal s for an unknown manifold might be more trouble than it is worth if the dimensionality reduction from the original data is already significant, i.e. $2d \ll r$.

In light of these arguments, for the purposes of the present paper, we set the objective of manifold learning to be the recovery of an embedding of $\mathcal{M}$ into $\mathbb{R}^s$ subject to $d \leq s \leq 2d$ and with the additional assumption that s is sufficiently large to allow a smooth embdding. That being said, the choice of s will only be discussed tangentially in this paper and even then, the constraint $s \leq 2d$ will not be enforced.

Existing algorithms fall into two categories w.r.t. the embedding dimension s. Most algorithms work under the constraint $s = d$; exception make the LE and DM algorithms, which require s as input, and implicitly assume $d \leq s$.

2.3 Statistical consistency of manifold learning algorithms

The previous section defined smoothness of the embedding in the ideal, continuous case, when the "input data" covers the whole manifold $\mathcal{M}$ and the algorithm is represented by the map $f : \mathcal{M} \to \mathbb{R}^s$. This analysis is useful in order to define what is mathematically possible in the limit.

Naturally, we would hope that a real algorithm, on a real finite data set, behaves in a way similar to its continuous counterpart. In other words, as the sample size $n = |\mathcal{D}| \to \infty$, we want the output of the algorithm $f_n(\mathcal{D})$ to converge to the output $f(\mathcal{M})$ of the continous algorithm, irrespective of the particular sample, in a probabilistic sense. This is what is generally understood as *consistency* of the algorithm.

Consistency is a very important property. It ensures that the inferences we draw from a finite sample carry over to the generating process that produced the sample. It also lets us study the idealized continuous case in order to make predictions about an algorithm's behavior for finite samples.

Proving consistency of various manifold derived quantities has received considerable attention in the literature, e.g. (Bernstein et al., 2000), (von Luxburg et al., 2008). In view of these results and others, we will assume when necessary that an embedding algorithm is consistent and in the limit produces a smooth embedding.

We now turn to the next desirable property, one for which negative results abound.

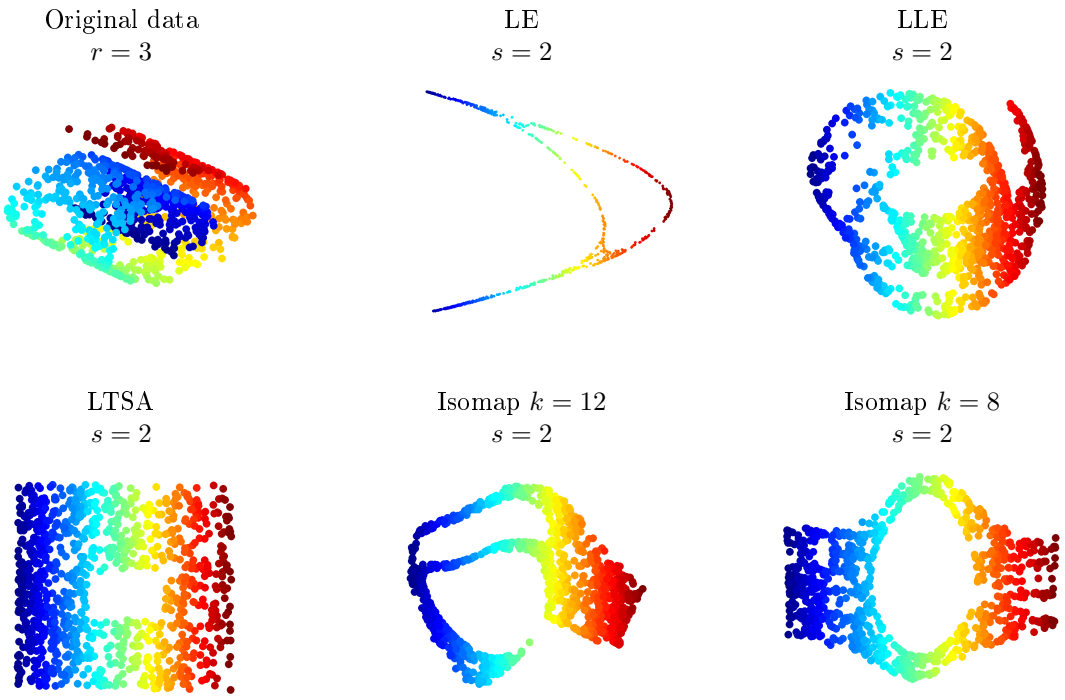


Figure 2: Manifold learning algorithms distort the geometry of the data. The classical "Swiss roll" example is shown here embedded via a variety of manifold learning algorithms. The original data is in $r = 3$ dimensions; note that for the "Swiss roll" choosing $r > 3$ ambient dimensions does not affect the resulting embeddings.

2.4 Manifold Geometry Preservation

Having a smooth mapping from $f : \mathbb{R}^r \to \mathbb{R}^s$ guarantees that neighborhoods in $\mathbb{R}^r$ will be mapped into neighborhoods in the embedding space $\mathbb{R}^s$ with some amount of "stretching", and vice versa. A reasonable question, therefore, is whether we can reduce this amount of "stretching" to a minimum, even to zero. In other words, can we preserve not only neighborhood relations, but also distances within the manifold? Or, going one step further, could we find a way to simultaneously preserve distances, areas, volumes, angles, etc. - in a word, the *intrinsic geometry* - of the manifold?

Manifold learning algorithms generally fail at preserving the geometry, even in simple cases. We illustrate this with the well-known example of the "Swiss-roll with a hole" (Figure 2), a two dimensional strip with a rectangular hole, rolled up in three dimensions, sampled uniformly. The reader will note that no matter how the original sheet is rolled without stretching, lengths of curves within the sheet will be preserved. (These concepts will be formalized in the next section.) So will be areas and angles between curves. This does not happen as manifold algorithms embed this data set. The LTSA algorithm recovers the original strip up to an affine coordinate transformation (the strip is turned into a square), for the other algorithms the "stretching" of the original manifold varies with the location on the manifold. As a consequence, distances, areas, angles between curves, that is, the intrinsic geometric quantities, are not preserved between the original manifold and the embeddings produced by these algorithms.

These shortcomings have been recognized and discussed in the literature (Goldberg et al., 2008; Zha and Zhang, 2003). More illustrative examples can easily be generated with the software in (Wittman, 2005).

The problem of geometric distortion is central to our paper, and our main contribution is to offer a constructive solution to it. The definitions of the relevant concepts and the rigorous statement of the problem we will be solving begin in the next section.

We conclude this section by stressing that the consistency of an algorithm, while being a necessary property, does not help alleviate the geometric distortion problem, because it merely guarantees that the *mapping* from a set of points in high dimensional space to a set of points in s-space induced by a manifold learning algorithm converges. It will not guarantee that the mapping recovers the correct geometry of the manifold. In other words, even with infinite data, the distortions seen in Figure 2 will persist.

3. Riemannian Geometry

In this section, we will formalize what it means for an embedding $f : \mathcal{M} \to \mathbb{R}^m$ to preserve the geometry of $\mathcal{M}$, which is the stated goal of our work.

3.1 The Riemannian Metric

The extension of Euclidean geometry to $\mathcal{M}$ is possible via the Riemannian metric, which is defined on the tangent space $T_p\mathcal{M}$ for each point $p \in \mathcal{M}$.

Definition 4 (Riemannian Metric) *The Riemannian metric g is a symmetric and positive definite tensor field which defines an inner product $<,>_g$ on the tangent space $T_p\mathcal{M}$ for every $p \in \mathcal{M}$.*

Definition 5 (Riemannian Manifold) *A Riemannian manifold $(\mathcal{M}, g)$ is a smooth manifold $\mathcal{M}$ with a Riemannian metric g defined at every point $p \in \mathcal{M}$.*

The inner product $< u, v >_g = g_{ij} u^i v^j$ (with the Einstein summation convention[1]) for $u, v \in T_p\mathcal{M}$ is used to define usual geometric quantities such as the norm of a vector $||u|| = \sqrt{< u, v >_g}$ and the

1. This convention assumes implicit summation over all indices appearing both as subscripts and superscripts in an expression. E.g in $g_{ij} u^i v^j$ the symbol $\sum_{i,j}$ is implicit.

angle between two vectors $\cos(\theta) = \frac{<u,v>_g}{||u||||v||}$. Thus, in any coordinate representation of $\mathcal{M}$, g at point p is represented as a $d \times d$ symmetric positive definte matrix.

The inner product g also defines infinitesimal quantities such as the line element $dl^2 = g_{ij}dx^i dx^j$ and the volume element $dV = \sqrt{\det(g)}dx^1 \dots dx^d$, both expressed in local coordinate charts. The length l of a curve $c : [a, b] \to \mathcal{M}$ parametrized by t then becomes

$$l(c) \;=\; \int_a^b \sqrt{g_{ij}\frac{dx^i}{dt}\frac{dx^j}{dt}}dt, \tag{2}$$

where $(x^1, ..., x^d)$ are the coordinates of chart $(U, \mathbf{x})$ with $c([a, b]) \subset U$. Similarly, the volume of $W \subset U$ is given by

$$\mathrm{Vol}(W) = \int_W \sqrt{\det(g)}dx^1 \dots dx^d\,. \tag{3}$$

Obviously, these definitions are trivially extended to overlapping charts by means of the transition map (1). For a comprehensive treatement of calculus on manifolds the reader is invited to consult (Lee, 1997).

3.2 Isometry and the Pushforward Metric

Having introduced the Riemannian metric, we can now formally discuss what it means for an embedding to preserve the geometry of $\mathcal{M}$.

Definition 6 (Isometry) *The smooth map $f : \mathcal{M} \to \mathcal{N}$ between two Riemannian manifolds $(\mathcal{M}, g)$, $(\mathcal{N}, h)$ is called an* isometry *iff for all $p \in \mathcal{M}$ and all $u, w \in T_p(\mathcal{M})$*

$$< u, w >_{g(p)} = < df(p)u, d_p f w >_{h(f(p))}$$

In the above, $df(p)$ denotes the Jacobian of f at p, i.e. the map $df(p) : T_p\mathcal{M} \to T_{f(p)}\mathcal{N}$. An embedding will be isometric if $(f(\mathcal{M}), h|_{f(\mathcal{M})})$ is isometric to $(\mathcal{M}, g)$ where $h|_{f(\mathcal{M})}$ is the restriction of h, the metric of the embedding space $\mathcal{N}$, to the tangent space $T_{f(p)}f(\mathcal{M})$. An isometric embedding obviously preserves path lengths, angles, velocities, areas and volumes. It is then natural to take isometry as the strictest notion of what it means for an algorithm to "preserve geometry".

We also formalize what it means to carry the geometry over from a Riemannian manifold $(\mathcal{M}, g)$ via an embedding f.

Definition 7 (Pushforward Metric) *Let f be an embedding from the Riemannian manifold $(\mathcal{M}, g)$ to another manifold $\mathcal{N}$. Then the pushforward $h = \varphi^* g$ of the metric g along $\varphi \equiv f^{-1}$ is given by*

$$\langle u, v\rangle_{\varphi^* g_p} \quad = \quad \left\langle df(p)^{-1}u, df(p)^{-1}v\right\rangle_{g_p}\,,$$

for $u, v \in T_{f(p)}\mathcal{N}$ and where $df(p)^{-1}$ denotes the Jacobian of f^{-1}.

This means that, by construction, $(\mathcal{N}, h)$ is isometric to $(\mathcal{M}, g)$.

As the defintion implies, the superscript -1 also refers to the fact that $df(p)^{-1}$ is the matrix inverse of the jacobian $df(p)$. This inverse is well-defined since f has full rank d. In the next section, we will extend this definition by considering the case where f is no longer full-rank.

3.3 Isometric Embedding vs. Metric Learning

Now consider a manifold embedding algorithm, like Isomap or Laplacian Eigenmaps. These algorithms take points $p \in \mathbb{R}^r$ and map them through some function f into $\mathbb{R}^s$. The geometries in the two representations are given by the induced Euclidean scalar products in $\mathbb{R}^r$ and $\mathbb{R}^s$, respectively,

which we will denote[2] by δ_r, δ_s. In matrix form, these are represented by $d \times d$ unit matrices[3]. In view of the previous definitions, the algorithm will preserve the geometry of the data only if the new manifold $(f(\mathcal{M}), \delta_s)$ is isometric to the original data manifold $(\mathcal{M}, \delta_r)$. .

The existence of an isometric embedding of a manifold into $\mathbb{R}^s$ for some s large enough is guaranteed by Nash's theorem (Nash, 1956), reproduced here for completeness.

Theorem 8 *If $\mathcal{M}$ is a given d-dimensional Riemannian manifold of class C^k, $3 \leq k \leq \infty$ then there exists a number $s \leq d(3d+11)/2$ if $\mathcal{M}$ is compact, or $s \leq d(d+1)(3d+11)/2$ if $\mathcal{M}$ is not compact, and an injective map $f : \mathcal{M} \longrightarrow \mathbb{R}^s$ of class C^k, such that*

$$< u, v > = < df(p)(v),\, df(p)(v) >$$

for all vectors u, v in $T_p\mathcal{M}$.

The method developed by Nash to prove the existence of an isometric embedding is not practical when it comes to finding an isometric embedding for a data manifold. The problem is that the method involves tightly wrapping the embedding around extra dimensions, which, as observed by (Dreisigmeyer and Kirby, 2007), may not be stable numerically.

A few existing algorithms can provably obtain isometric embeddings, and among them are several recent innovative papers. The work of (Lin and Zha, 2008) propose a direct approach to global isometry. In their framework, a reference point p on the manifold is chosen; p will represent the origin of a system of normal coordinates for the manifold (it is assumed, as usual, that the manifold is covered by a single chart). Then, shortest paths from p, taken as surrogates for geodesic distances, are mapped recursively into the normal coordinate system. In (Lin et al., 2013) the isometry is constructed by recursively building a set of mutually orthogonal *parallel vector fields* representing a system of normal coordinates. These algorithm do not come with theoretical guarantees. In Verma (2013) two algorithms are preented, which boldly combine random projections with ideas from the proofs of Nash's theorem. These are guaranteed to preserve distances approximately, but they currently do not have a practical implmentation, hence they remain of theoretical interest.

The remaining postive results concerning isometric embedding pertain on the recovery of *flat manifolds*, i.e. of manifolds that are isometric to an Euclidean space. (Bernstein et al., 2000) establish that Isomap's output is isometric to the original manifold only when the parameter space is Euclidean and convex. *MMP: they establish that isomap reproduces the geodesic distances iff the manifold is flat. is this sufficient? YES* (Donoho and Grimes, 2005) make a similar argument *MMP: continuous isomap plus mds plus convexity equals isometry* w.r.t the continuous variant of the Isomap algorithm (called therein Continous Isomap). In (Donoho and Grimes, 2003) the same authors propose the Hessian Eigenmaps algorithm, which is able to recover isometric coordinates up to a rigid deformation, under the relaxed assumption that the domain of the coordinates is connected (instead of convex, as in the Continuous Isomap). Further insights into the Continuous Isomap algorithm's behaviour come from (Zha and Zhang, 2003).

For the remaining cases, such as those in section 2.4, the popular approach to resolving this problem is to try to correct the the resulting embeddings as much as possible (Goldberg and Ritov, 2009; Dreisigmeyer and Kirby, 2007; Behmardi and Raich, 2010; Zha and Zhang, 2003).

We believe that there is a more elegant solution to this problem, which is to carry the geometry over along with f instead of trying to correct f itself. Thus, we will take the coordinates f produced by any reasonable embedding algorithm, and augment them with the appropriate (pushforward) metric h that makes $(f(\mathcal{M}), h)$ isometric to the original manifold $(\mathcal{M}, g)$. We call this procedure *metric learning*. The remainder of this paper will present the mathematics, the algorithm, and the statistics underlying it.

2. Following (**?**).
3. The actual metrics for $\mathcal{M}$ and $f(\mathcal{M})$ are $\delta_r|_{\mathcal{M}}$ and $\delta_s|_{f(\mathcal{M})}$, the restrictions of δ_r and δ_s to the tangent bundle $T\mathcal{M}$ and $Tf(\mathcal{M})$.

MMP: MOVE THIS AT THE END OF NEXT SECTON This appraoch has three important advantages: it is algorithm-agnostic; it immediately quantifies the distortion resulting from the emebdding; and it recovers the intrinsic geometry exactly as $n \to \infty$.

4. Recovering the Riemannian Metric: The Mathematics

We now establish the mathematical results that will allow us to estimate the Riemannian metric g from data. The key to obtaining g for any C^∞-atlas is the Laplace-Beltrami operator $\Delta_\mathcal{M}$ on $\mathcal{M}$, which we introduce below. Thereafter, we extend the solution to manifold embeddings, where the embedding dimension s is, in general, greater than the dimension of $\mathcal{M}$, d.

4.1 Recovering g with the Laplace-Beltrami Operator

Definition 9 (Laplace-Beltrami Operator) *The Laplace-Beltrami operator $\Delta_\mathcal{M}$ acting on a twice differentiable function $f : \mathcal{M} \to \mathbb{R}$ is defined as $\Delta_\mathcal{M} f \equiv div\ grad(f)$.*

In local coordinates, for chart (U, x), the Laplace-Beltrami operator$\Delta_\mathcal{M}$ is expressed by means of g as (Rosenberg, 1997)

$$\Delta_\mathcal{M} f = \frac{1}{\sqrt{\det(g)}} \frac{\partial}{\partial x^l} \left(\sqrt{\det(g)} g^{lk} \frac{\partial}{\partial x^k} f \right) . \tag{4}$$

In (4), g^{lk} denotes the l, k component of the inverse of g and Einstein summation is assumed. The expression in parentheses is (the definition of) $grad(f)$ in the coordinates x, and the outer expression defines the divergence operator div in the same set of coordinates.

The Laplace-Beltrami operator has been widely used in the context of manifold learning, and we will exploit various existing results about its properties in this paper. We will present those results when they become necessary. For more background, the reader is invited to consult (Rosenberg, 1997). In particular, methods for estimating $\Delta_\mathcal{M}$ from data exist and are well studied (Coifman and Lafon, 2006; Hein et al., 2007; Belkin et al., 2009). This makes using (4) ideally suited to recover g. The simple but powerful proposition below is the key to achieving this.

Proposition 10 *Given a coordinate chart (U, x) of a smooth Riemannian manifold $\mathcal{M}$ and $\Delta_\mathcal{M}$ defined on $\mathcal{M}$, then the $g(p)^{-1}$, the inverse of the Riemannian metric at point $p \in U$ as expressed in local coordinates x, can be derived from*

$$g^{ij} \;=\; \frac{1}{2} \Delta_\mathcal{M} \left(x^i - x^i(p) \right) \left(x^j - x^j(p) \right) |_{x^i = x^i(p), x^j = x^j(p)} \tag{5}$$

with $i, j = 1, \ldots, d$.

Proof This follows directly from (4). Applying $\Delta_\mathcal{M}$ to the coordinate products of x^i and x^j centered at $x(p)$, i.e. $\frac{1}{2} \left(x^i - x^i(p) \right) \left(x^j - x^j(p) \right)$, and evaluating this expression at $x = x(p)$ using (4) gives

$$g^{lk} \frac{\partial}{\partial x^l} \left(x^i - x^i(p) \right) \times \frac{\partial}{\partial x^k} \left(x^j - x^j(p) \right) |_{x^i = x^i(p), x^j = x^j(p)} \;=\; g^{ij} ,$$

since all the first order derivative terms vanish. The superscripts i, j in the equation above and in (5) refer to the fact that g^{ij} is the inverse of g for coordinates x^i and x^j. ■

With all the components of g^{-1}known, it is straightforward to compute its inverse and obtain $g(p)$. The power of Proposition 10 resides in the fact that the coordinate chart is arbitrary. Given a coordinate chart (or embeddding, as will be shown below) one can apply the *coordinate free* Laplace-Beltrami operator as in (5) to recover g for that coordinate chart.

4.2 Recovering a Rank-Deficient Embedding Metric

In the previous section, we have assumed that we are given a coordinate chart (U, x) for a subset of $\mathcal{M}$, and have shown how to obtain the Riemannian metric of $\mathcal{M}$ in that coordinate chart via the Laplace-Beltrami operator.

Here, we will extend the method to work with any embedding of $\mathcal{M}$. The main change will be that the embedding dimension s may be larger than the manifold dimension d. In other words, there will be $s \geq d$ embedding coordinates for each point p, while g is only defined for a vector space of dimension d. An obvious solution to this is to construct a coordinate chart around p from the embedding f. This is often unecessary, and in practice it is simpler to work directly from f until the coordinate charts representation is actually required. In fact, once we have the correct metric for $f(\mathcal{M})$, it becomes relatively easy to construct coordinate charts for $\mathcal{M}$.

Working directly with the embedding f means that at each embedded point f_p, there will be a corresponding $s \times s$ matrix h_p defining a scalar product. The matrix h_p will have rank d, and its null space will be orthogonal to the tangent space $T_{f(p)}f(\mathcal{M})$. We define h so that $(f(\mathcal{M}), h)$ is isometric with $(\mathcal{M}, g)$. Obviously, the tensor h over $T_{f(p)}f(\mathcal{M}) \bigoplus T_{f(p)}f(\mathcal{M})^{\perp} \cong \mathbb{R}^s$ that achieves this is an extension of the *pushforward* of the metric g of $\mathcal{M}$.

Definition 11 (Embedding (Pushforward) Metric) *For all $u, v \in T_{f(p)}f(\mathcal{M}) \bigoplus T_{f(p)}f(\mathcal{M})^{\perp}$, the* embedding *pushforward metric h, or shortly the* embedding metric*, of an embedding f at point $p \in \mathcal{M}$ is defined by the inner product*

$$< u, v >_{h(f(p))} \quad \equiv \quad < df(p)^{\dagger}(u), df(p)^{\dagger}(v) >_{g(p)}, \tag{6}$$

where $df(p)^{\dagger} : T_{f(p)}f(\mathcal{M}) \bigoplus T_{f(p)}f(\mathcal{M})^{\perp} \to T_p\mathcal{M}$ is the pseudoinverse of the Jacobian $df(p)$ of $f : \mathcal{M} \to \mathbb{R}^s$

In matrix notation (6) becomes

$$u^T (df(p))^T h(p) (df(p)) v \;=\; u^t g(p) v \quad \text{for all } u, v \in T_p\mathcal{M} \tag{7}$$

Hence,

$$h(p) \;\equiv\; \left((df(p))^t\right)^{\dagger} g(p) \, (df(p))^{\dagger} \tag{8}$$

where $df(p), g(p), h(p)$ are matrices of size $d \times s$, $d \times d$ and $s \times s$ respectively and the superscript ^{t} denotes matrix transposition.

When $\mathcal{M} \subset \mathbb{R}^r$, with metric inherited from the ambient Euclidean space, as is often the case for manifold learning, g is simply the Euclidean metric in $\mathbb{R}^r$ projected onto $T_p\mathcal{M}$.

From Definition **??** it follows that h is symmetric semi-positive definite (positive definite on $T_pf(\mathcal{M})$ and null on $T_pf(\mathcal{M})^{\perp}$), rather than symmetric positive definite like g.

Proposition 12 *Let f be an embedding of $\mathcal{M}$ into $\mathbb{R}^s$; then $(\mathcal{M}, g)$ and $(f(\mathcal{M}), h)$ are isometric, where h is the embedding metric h defined in 11. Furthermore, h is null over $T_{f(p)}f(\mathcal{M})^{\perp}$.*

Proof Let $u \in T_p\mathcal{M}$, then the map $df(p)^{\dagger} \circ df(p) : T_p\mathcal{M} \to T_p\mathcal{M}$ satisfies $df(p)^{\dagger} \circ df(p)(u) = u$, since f has rank $d = dim(T_p\mathcal{M})$. So $\forall u, v \in T_p\mathcal{M}$ we have

$$< df(p)(u), df(p)(v) >_{h(f(p))} = < df(p)^{\dagger} \circ df(p)(u), df(p)^{\dagger} \circ df(p)(v) >_{g(p)} = < u, v >_{g(p)} \tag{9}$$

Therefore, h ensures that the embedding is isometric. Moreover, the null space of the pseudo-inverse is $\text{Null}(df(p)^{\dagger}) = \text{Im}(df(p))^{\perp} = T_pf(\mathcal{M})^{\perp}$, hence $\forall u \in T_pf(\mathcal{M})^{\perp}$ and v arbitrary, the inner product defined by h satisfies

$$< u, v >_{h(f(p))} = < df(p)^{\dagger}(u), df(p)^{\dagger}(v) >_{g(p)} = < 0, df(p)^{\dagger}(v) >_{g(p)} = \; 0\,. \tag{10}$$

By symmetry of h, the same holds true if u and v are interchanged. ■

Having shown that h, as defined, satisfies the desired properties, the next step is to show that it can be recovered using $\Delta_{\mathcal{M}}$, just as g was in Section 4.1.

Proposition 13 *Let f be an embedding of $\mathcal{M}$ into $\mathbb{R}^s$, and df its Jacobian. Then, the embedding metric $h(p)$ is given by the pseudoinverse of $\tilde{h}$, where*

$$\tilde{h}^{ij}(p) \;=\; \Delta_{\mathcal{M}} \frac{1}{2}\left(f^i - f^i(p)\right)\left(f^j - f^j(p)\right)|_{f^i=f^i(p),f^j=f^j(p)} \tag{11}$$

Proof We express $\Delta_{\mathcal{M}}$ in a coordinate chart (U, x). $\mathcal{M}$ being a smooth manifold, such a coordinate chart always exists. Applying $\Delta_{\mathcal{M}}$ to the centered product of coordinates of the embedding, i.e. $\frac{1}{2}\left(f^i - f^i(p)\right)\left(f^j - f^j(p)\right)$, then (4) means that

$$\begin{aligned}\Delta_{\mathcal{M}} \frac{1}{2}\left(f^i - f^i(p)\right)\left(f^j - f^j(p)\right)|_{f^i=f^i(p),f^j=f^j(p)} &= g^{lk}\frac{\partial}{\partial x^l}\left(f^i - f^i(p)\right) \times \frac{\partial}{\partial x^k}\left(f^j - f^j(p)\right)|_{f^i=f^i(p),f^j=f^j(p)} \\ &= g^{kl}\frac{\partial f^i}{\partial x^l}\frac{\partial f^j}{\partial x^k}\end{aligned}$$

Using matrix notation as before, the above results take the form

$$g^{kl}\frac{\partial f^i}{\partial x^l}\frac{\partial f^j}{\partial x^k} = (df(p)g(p)^{-1}(df(p))^t)_{ij} = (\tilde{h}(p))_{ij}\,. \tag{12}$$

Hence, $\tilde{h}(p) = df(p)g(p)^{-1}(df(p))^t$ and it remains to show that $h(p) \;=\; (\tilde{h}(p))^\dagger$, i.e. that

$$\left(df(p)^t\right)^\dagger g(df(p))^\dagger = \left(df(p)g(p)^{-1}(df(p))^t\right)^\dagger. \tag{13}$$

This is straightforward for square invertible matrices, but if $d < s$, this might not be the case. Hence, we need an additional technical fact: guaranteeing that *MMP: make it a lemma?*

$$(AB)^\dagger = B^\dagger A^\dagger \tag{14}$$

requires $C = AB$ to constitute a full-rank decomposition of C, i.e. for A to have full column rank and B to have full row rank (Ben-Israel and Greville, 2003). In the present case, $g(p)^{-1}$ has full rank, $df(p)$ has full column rank, and $(df(p))^t$ has full row rank. All these ranks are equal to d by virtue of the fact that $dim(\mathcal{M}) = d$ and f is an embedding of $\mathcal{M}$. Therefore, applying (14) repeatedly to $df(p)g(p)^{-1}(df(p))^t$, implicitly using the fact that $\left(g(p)^{-1}(df(p))^t\right)$ has full row rank since $g(p)^{-1}$ has full rank and $df(p)$ has full row rank, proves that $h(p)$ is the pseudoinverse of $\tilde{h}(p)$. ■

5. Recovering the Riemannian Metric: The Algorithm

The results in the previous section hold for any embedding of $\mathcal{M}$ and can therefore be applied to the output of any embedding algorithm, leading to the estimation of the corresponding g if $d = s$ or h if $d < s$. In this section, we present our algorithm for the estimation procedure, called MetricML. Throughout, we assume that an appropriate embedding dimension $s \geq d$ is already selected.

5.1 The MetricML Algorithm

The input data for a manifold learning algorithm is a set of points $\mathcal{D} = p_1, \ldots, p_n \subset \mathcal{M}$ where $\mathcal{M}$ is an unknown Riemannian manifold. Our MetricML algorithm takes as input, along with dataset $\mathcal{D}$, a bandwidth parameter ϵ, the manifold dimension d and the embedding dimension s. It also calls an embedding algorithm, chosen by the user which we will denote by GenericEmbed. MetricML proceeds in four steps, the first three being preparations for the key fourth step.

1. construct a weighted neighborhood graph
2. calculate the graph Laplacian $\tilde{L}_{n,\epsilon}$
3. map the data $p \in \mathcal{D}$ to $f_n(p) \in \mathbb{R}^s$ by GenericEmbed
4. apply the Laplacian $\tilde{L}_{n,\epsilon}$ to the coordinates f_n to obtain the embedding metric h

Figure 3 presents the MetricML algorithm in pseudocode. The subscript n in the notation indicates that f, $\tilde{L}$, etc are discretized, "sample" quantities, (i.e. f_n is a vector of f evaluated at the n points of $\mathcal{D}$ and $\tilde{L}_{n,\epsilon}$ is a $n \times n$ matrix) as opposed to the continuous quantities (functions, operators) that we were considering in the previous sections. The $n \times n$ matrices supported on the neighborhood graph $\mathcal{G}$ are denoted by capital letters (e.g. K_n).

The first step, common to many manifold learning algorithms, consists of finding the set of neighbors within radius ϵ for each data point p, and evaluating a similarity $K_n(p, p')$ for each pair of points p, p' that are neighbors. The bandwidth parameter ϵ determines the size of the neighborhood for each point, and, hence, the density of the graph $\mathcal{G}$. The similarity matrix K_n represents the heat kernel (Belkin and Niyogi, 2002) applied to point differences $||p - p'||^2$:

$$K_n(p, p') = \exp - \left(\frac{||p - p'||^2}{\epsilon} \right) \mathbb{1}_{[||p-p'||^2 \leq \epsilon]}$$

The use of a *truncated kernel* induces sparsity on K_n and $\tilde{L}_{n,\epsilon}$, which substantially reduces the computational complexity involved in estimating h. Moreover, it allows the convergence of $\tilde{L}_{n,\epsilon}$ to be extended to non-compact manifolds (Hein et al., 2007) (provided additional conditions are imposed on the curvature of $\mathcal{M}$).

In the second step, we use the similarity matrix K_n to obtain $\tilde{L}_{n,\epsilon}$ (16), an estimator of the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$, by following the procedure described in (Hein et al., 2007) (see also (Coifman and Lafon, 2006; Giné and Koltchinskii, 2006; Belkin et al., 2009)). This procedure is implemented by the *Laplacian* algorithm in Figure 3. For completeness, we describe it below. One starts by the computation of the random walk graph Laplacian (Hein et al., 2007):

$$L_{n,\epsilon} \equiv \frac{I_n - T_n^{-1} K_n}{\epsilon}, \tag{15}$$

where T_n is the diagonal matrix of outdegrees, i.e. $T_n = \text{diag}\{t(p), p \in \mathcal{D}\}$ with $t(p) = \sum_{p' \in \mathcal{D}} K_n(p, p')$. The random walk graph Laplacian $\tilde{L}_{n,\epsilon} f$ applied to any function $f \in C^2(\mathcal{M})$ is then known to converge uniformly a.s. to $\Delta_{\mathcal{M}} f$ as $n \to \infty$ and $\epsilon \to 0$ (Ting et al., 2010), if the sampling distribution used to obtain the data $\mathcal{D}$ is uniform.

If the sampling distribution is not uniform, or simply unknown (as is generally the case), it is necessary to renormalize the adjacency matrix K_n before obtaining $\tilde{L}_{n,\epsilon}$. This ensures that the random walk graph Laplacian still converges to $\Delta_{\mathcal{M}}$ (Coifman and Lafon, 2006; Hein et al., 2007). The renormalization proceeds by defining a new adjacency matrix $\tilde{K}_n = T_n^{-1} K_n T_n^{-1}$ for $\mathcal{G}_n$, along with a new outdegree matrix $\tilde{T}_n = \text{diag}\{\tilde{t}(p), p \in \mathcal{D}\}$ with $\tilde{t}(p) = \sum_{p' \in \mathcal{D}} \tilde{K}_n(p, p')$. The associated random walk graph Laplacian is then given by

$$\tilde{L}_{n,\epsilon} \equiv \frac{I_n - \tilde{T}_n^{-1} \tilde{K}_n}{\epsilon}. \tag{16}$$

This new random walk graph Laplacian is consistent for $\Delta_{\mathcal{M}}$ irrespective of the sampling density (Hein et al., 2007), a fact that will be instrumental in proving the consistency of our new algorithm in Section 6. *MMP: shorten or remove lap estimation details?*

Note that the Laplacian estimate $\tilde{L}_{n,\epsilon}$ depends only on the data and ϵ, and not on the embedding method selected by the user.

The third step consists of embedding of the data points $p \in \mathcal{D}$ in $\mathbb{R}^s$ by the desired algorithm GenericEmbed. This can be any one of the many existing manifold learning algorithms, such as the Isomap, LTSA, or Diffusion Maps. The algorithm may take other parameters than the data $\mathcal{D}$ and s, such as the bandwidth parameter ϵ. But the algorithm also may use its own bandwidth parameter, different from ϵ.

It is worth noting at this point that there may be overlap in the computations involved in the first three steps. Indeed, a large number of the common embedding algorithms, including Laplacian Eigenmaps, Diffusion Maps, Isomap, LLE, and LTSA use a neighborhood graph and/or similarities in order to obtain an embedding. In addition, Diffusion Maps and Eigemaps obtain an embedding from the eigendecomposition of $\tilde{L}_{n,\epsilon}$ or a similar operator. While we define the steps of our algorithm in their most complete form, we encourage the reader to take advantage of any efficiencies that may result from avoiding to compute the same quantities multiple times.

The fourth and final step of our algorithm consists of computing the embedding metric of the manifold in the coordinates f_n output by the GenericEmbed algorithm. Step 4, a applies the $n \times n$ Laplacian matrix $\tilde{L}_{n,\epsilon}$ obtained in Step 2 to pairs f_n^i, f_n^j of embedding coordinates of the data obtained in Step 3. We use the symbol $\cdot$ to refer to the elementwise product between two vectors. Specifically, for two vectors $x, y \in \mathbb{R}^n$ denote by $x \cdot y$ the vector $z \in \mathbb{R}^n$ with coordinates $z = (x_1 y_1, \ldots, x_n y_n)$. This product is simply the usual function multiplication on $\mathcal{M}$ restricted to the sampled points $\mathcal{D} \subset \mathcal{M}$. Hence, equation (17) is equivalent to applying equation (11) to all the points of $p \in \mathcal{D}$ at once. The result are the vectors $\tilde{h}_n^{ij}$, each of which is an n-dimensional vector, with an entry for each $p \in \mathcal{D}$. Then, in Step 4, b, at each embedding point $f(p)$ the embedding metric $h_n(p)$ is computed as the matrix (pseudo) inverse of $[\tilde{h}_n^{ij}(p)]^{ij=1:s}$. If the embedding dimension s is larger than the manifold dimension d, we will obtain the rank d embedding metric h_n; otherwise, we will obtain the Riemannian metric g_n.

Of the input parameters to the MetricML, the intrinsic dimension d is used exclusively in Step 4, in order to obtain a stable pseudoinverse of $\tilde{h}_n$. The embedding dimension s is passed as a parameter to the GenericEmbed algorithm, which outputs coordinate functions $f_n^{1:s}$.

In summary, the principal novelty in the MetricML algorithm is its last step: the estimation of the embedding metric h_n from the estimate of $\tilde{L}_{n,\epsilon}$ and the embedding coordinates f_n . The embedding metric h_n establishes a direct correspondence between geometric computations performed using $(f(\mathcal{D}), h_n)$ and those performed directly on the original $\mathcal{D}$. In other words, once augmented with their corresponding h_n, all embeddings become geometrically equivalent to each other, and to the orginal data.

5.2 Computation Details

Obtaining the neighborhood graph involves computing n^2 distances in r dimensions. If the data is high or very high-dimensional, which is often the case, and if the sample size is large, which is often a requirement for correct manifold recovery, this step could be by far the most computationally demanding of the algorithm. However, much work has been devoted to speeding up this task, and approximate algorithms are now available, which can run in linear time in n and have very good accuracy (Ram et al. (2010)). Ultimately, this computationally intensive preprocessing step is required by all of the well known embedding algorithms, and would remain necessary even if our goal were solely to embed the data, and not to compute the Riemannian metric.

Step 2 of the algorithm operates on a sparse $n \times n$ matrix. If the neighborhood size is no larger than k, then it will be of order $\mathcal{O}(nk)$, and $\mathcal{O}(n^2)$ otherwise.

The computation of the embedding in Step 3 is algorithm-dependent. For the most common algorithms, it will involve eigenvector computations. These can be performed by Arnolidi iterations that each take $\mathcal{O}(n^2 s)$ computations, where n is the sample size, and s is the embedding dimension or, equivalently, the number of eigenvectors computed. This step, or a variant thereof, is also a

Algorithm MetricML

Input $\mathcal{D}$ as set of n data points in $\mathbb{R}^r$, s the number of the dimensions of the embedding, ϵ the bandwidth parameter, and GenericEmbed($\mathcal{D}$, s, ϵ) a manifold learning algorithm, that outputs s dimensional embedding coordinates

1. Construct the similary matrix
   For each pair of points $p, p' \in \mathcal{D}$, set $k_\epsilon(p,p') = e^{\frac{1}{\epsilon}||p-p'||^2}$ if p, p' are neighbors and 0 otherwise. Two points are neighbors if $||p-p'||^2 \leq \epsilon$; the graph with nodes in the data points and with an edge connecting every pair of neighbors is called the *neighborhood graph* of the data. Let $K_n = [k_\epsilon(p,p')]_{p,p'\in\mathcal{D}}$
2. Construct the Laplacian matrix
   $$\tilde{L}_{n,\epsilon} \;=\; \textsc{Laplacian}(K_n, \epsilon)$$
3. Obtain the *embedding coordinates* $f_n(p) = (f_n^1(p), \ldots, f_n^s(p))$ of each point $p \in \mathcal{D}$ by
   $$[f_n(p)]_{p\in\mathcal{D}} \;=\; \textsc{GenericEmbed}(\mathcal{D},\, s \text{ or } d, \epsilon)$$
4. Calculate the *embedding metric* $h_n(p)$ at each point
   (a) For i and j from 1 to s calculate the column vector $\tilde{h}_n^{ij}$ of dimension $n = |\mathcal{D}|$ by
   $$\tilde{h}_n^{ij} \;=\; \frac{1}{2}\left[\tilde{L}_{n,\epsilon}(f_n^i \cdot f_n^j) - f_n^i \cdot (\tilde{L}_{n,\epsilon} f_n^j) - f_n^j \cdot (\tilde{L}_{n,\epsilon} f_n^i)\right] \tag{17}$$
   (b) For each data point $p \in \mathcal{D}$, form the matrix $\tilde{h}_n(p) = [\tilde{h}^{ij}(p)]_{i,j\in 1,\ldots,s}$.
   (c) Calculate embedding metric at p by $h_n(p) = \tilde{h}_n^\dagger(p)$.

Output $(f_n(p), h_n(p))_{p\in\mathcal{D}}$

Algorithm Laplacian

1. For each point $p \in \mathcal{D}$ compute $t_n(p) \;=\; \sum_{p'\in\mathcal{D}} K_n(p,p')$; form the diagonal matrix $T_n \;=\; \text{diag}\{\, t_n(p),\, p \in \mathcal{D}\}$
2. Let $\tilde{K}_n \;=\; T_n^{-1} S_n T_n^{-1}$
3. Let $\tilde{t}_n(p) \;=\; \sum_{p'\in\mathcal{D}} \tilde{K}_n(p,p')$, $\tilde{T} = \text{diag}\{\, \tilde{t}_p, p \in \mathcal{D}\}$
4. Output $\tilde{L}_{n,\epsilon};= \left(I_n - \tilde{T}_n^{-1}\tilde{K}_n\right)/\epsilon$.

Figure 3: The MetricML Algorithm.

component of many embedding algorithms. Again, if sparsity is taken into account in the Arnoldi iteration, then the order decreases by n.

Finally, the newly introduced Step 4 involves obtaining an $s \times s$ matrix for each of the n points, and computing its pseudoinverse. Obtaining the $\tilde{h}_n$ matrices takes $\mathcal{O}(n^2 s^2)$ operations ($\mathcal{O}(nks^2)$ for sparse $\tilde{L}_{n,\epsilon}$ matrix).

Computing the pseudoinverse of $\tilde{h}$ generally means performing a Singular Value Decomposition (SVD). In the case $s > d$, $\tilde{h}_n$ will have a theoretical rank d, but numerically it might have rank between d and s. As such, it is important to set to zero the $s-d$ smallest singular values of $\tilde{h}_n$ when computing the pseudo-inverse. This is the key reason why s and d need to be known in advance. Failure to set the smallest singular values to zero will mean that h_n will be dominated by noise. Although estimating d is outside the scope of this work, it is interesting to note that the singular values of $\tilde{h}_n$ may offer a window into estimating d by looking for a "singular value gap".

The SVD decomposition offers other very useful insights into the embedding. Indeed, we know from 12 that h is positive definite over $T_{f(p)}f(\mathcal{M})$ and null over $T_{f(p)}f(\mathcal{M})^{\perp}$. This means that the singular vector(s) with non-zero singular value(s) of h at $f(p)$ define an orthogonal basis for $T_{f(p)}f(\mathcal{M})$, while the singular vector(s) with zero singular value(s) define an orthogonal basis for $T_{f(p)}f(\mathcal{M})^{\perp}$ (not that the latter is of particular interest). Having an orthogonal basis for $T_{f(p)}f(\mathcal{M})$ provides a natural framework for constructing a coordinate chart around p. The simplest option is to project a small neighborhood $f(U)$ of $f(p)$ onto $T_{f(p)}f(\mathcal{M})$, a technique we will use in Section 7 to compute areas/volumes. Another interesting approach would be to derive the exponential map for $f(U)$. However, computing all the geodesics of $f(U)$ is not a practical solution unless the geodesics themselves are of interest for the application. In either case, computing h allows us to achieve our set goal for manifold learning, i.e. construct a collection of coordinate charts for $\mathcal{D}$. As mentioned above, it is not always necessary, or even wise, to construct an Atlas of coordinate charts explicitly. It is really a matter of whether charts are required to performed the desired computations.

Moreover, the non-zero singular values yield a measure of the distortion induced by the embedding. Indeed, if the embedding were isometric to $\mathcal{M}$ with the metric inherited from $\mathbb{R}^s$, then the embedding metric h would have non-zero singular values equal to 1. This can be used in many ways, such as getting a global distortion for the embedding, and hence as a tool to compare various embeddings. It can also be used to define an objective function to minimize in order to get an isometric embedding, should such an embedding be of interest. From a local perspective, it gives insight into the magnitude of the distortion in specific regions of the manifold. The matrix $\tilde{h}(p)$ can prescribe a simple linear transformation of the embedding f that makes it approximate isometric to $\mathcal{M}$ in a neighborhood of p (with respect to the inherited metric δ_s). This latter attribute will be explored in more detail in Section 7.

The n SVD and pseudoinverse calculations take order $s^2 d$ operations each, which for simplicity we upper bound by s^3.

Thus, finding the Riemannian metric makes is small in comparison to the computational burden of finding the embedding. The MetricML Algorithm was implemented as part of the `megaman` open source package[4], and the benchmarks in McQueen et al. (2016) confirm empirically that for large data the overhead is under 0.5% of the total embedding time.

Obviously, proving consistency has received considerable interest. However, in the context of manifold learning, consistency can take different forms and it is easy to lose track of what is being proved, and of why it is significant.

MMP: Stuff from before 6.1. can be used in the proof Here, the sequence of bandwidths ϵ_n is such that $\epsilon_n \to 0$ as $n \to \infty$. The rate at which ϵ_n needs to go to zero depends on how $\tilde{L}_{n,\epsilon}$ is constructed and on the type of convergence. For example, if $\tilde{\mathcal{L}}_{n,\epsilon_n}$ is as defined in equation (16), then $\epsilon_n \to 0$ and $n\epsilon_n^{d+2} \to 0$ is sufficient to prove that $\tilde{\mathcal{L}}_{n,\epsilon_n}$ converges uniformly a.s. to $\Delta_{\mathcal{M}}$, i.e. $||\tilde{\mathcal{L}}_{n,\epsilon_n}(f)(x) - \Delta_{\mathcal{M}}(f)(x)||_{\infty} \overset{a.s.}{\to} 0\,,\ \forall f \in C^2(\mathcal{M})$ (Ting et al., 2010).

4. `http://mmp2.github.io/megaman/`

As this example illustrates, consistency of $\tilde{\mathcal{L}}_{n,\epsilon_n}$ is generally established for a fixed function $f \in C^3(\mathcal{M})$ rather than a random function f_n defined on the graph G_n (Hein et al., 2007; Giné and Koltchinskii, 2006; Ting et al., 2010). This means that proving the consistency of $\tilde{\mathcal{L}}_{n,\epsilon_n}$ or its eigenvectors in addition to proving the consistency of f_n does not suffice to establish (??).

The key to establishing (??) is Condition 3 of the Theorem (see Lemma 15 in the Appendix).

to determine under what conditions

$$\left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n\right|\right|_\infty \xrightarrow{p} 0\,, \tag{18}$$

where the L_∞-norm is taken over the sampled points $p_l \in \mathcal{D}$ with, $l \in \{1,\dots,n\}$. Indeed, from 18 it follows that $\left|\left|\Delta_{\mathcal{M}} f^i f^j - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n f^j_n\right|\right|_\infty \xrightarrow{p} 0$ and hence equation ??.

Although one can work directly with (18), as we will do below, we also that

$$\begin{aligned}\left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n\right|\right|_\infty &= \left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i + \tilde{\mathcal{L}}_{n,\epsilon_n} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n\right|\right|_\infty \\ &\le \left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i\right|\right|_\infty + \left|\left|\tilde{\mathcal{L}}_{n,\epsilon_n} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n\right|\right|_\infty .\end{aligned}$$

Hence, assuming that $f^i \in C^3(\mathcal{M})$ and that $\mathcal{M}$ is a compact manifold, we know there exits $\epsilon_n \to 0$ as $n \to \infty$ such that

$$\left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i\right|\right|_\infty \xrightarrow{a.s.} 0\,.$$

The problem, then, is reduced to showing $\left|\left|\tilde{\mathcal{L}}_{n,\epsilon_n} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n\right|\right|_\infty \xrightarrow{p} 0$.

6. Consistency of the Riemannian Metric estimator

As explained in Section 5, the input data for a manifold learning algorithm is a set of points $\mathcal{D} = p_1,\dots,p_n \subset \mathcal{M}$ where $\mathcal{M}$ is a compact Riemannian manifold. These points are assumed to be an i.i.d. sample from distribution $\pi(x)$ on $\mathcal{M}$. From this sample, a manifold learning algorithm, that we denote GenericEmbed constructs a map $f_n : \mathcal{D} \to \mathbb{R}^s$. If the algorithm GenericEmbed is consistent, when the sample size $n \to \infty$, f_n will converge to $f : \mathcal{M} \to \mathbb{R}^s$, the theoretical output of GenericEmbed if applied to the entire manifold $\mathcal{M}$. E.g., for the LE algorithm with renormalized kernel Coifman and Lafon (2006), f is be represented by the eigenfunctions of the Laplace-Beltrami operator on $\mathcal{M}$, while f_n are the eigenvectors of the Laplacian matrix $\tilde{L}_{n,\epsilon}$.

Once the map is obtained, one uses the MetricML algorithm to calculate the embedding metric h_n. Naturally, it is relevant to ask: Will the estimate h_n converge to the pushforward metric h in the limit $n \to \infty$? This is the consistency question to which we devote this section. To note that consistency of h_n is essentially equivalent[5] to the consistency of the pseudoinverses $\tilde{h}_n$, that is, to having $\tilde{h}_n$ as defined in (12) converge to $\tilde{h}$ of (6).*MMP: check these labels*

We summarize the first main result of this section in the Theorem below.

Theorem 14 *Let* GenericEmbed *be an embedding algorithm, $\mathcal{M}$ a compact manifold, a density π on $\mathcal{M}$ which is absolutely continuous with respect to the Lebesgue measure on $\mathcal{M}$, and $\mathcal{D}_n$ a sample of size n from $\mathcal{M}$ drawn i.i.d. according to π. If*

1. *The empirical embedding is consistent, i.e. there exists a sequence $\epsilon_n \to 0$ and a function $f : \mathcal{M} \to \mathbb{R}^s$ so that $\left|\left|f^i - f^i_n\right|\right|_\infty \xrightarrow{p} 0$, $i \in \{1,\dots,s\}$, where $f_n = \text{GenericEmbed}(\mathcal{D}_n)$, and where the L_∞-norm is taken over the sampled points $x_l \in \mathcal{D}$ with, $l \in \{1,\dots,n\}$.*
2. *The limit f is an embedding of $\mathcal{M}$ into $\mathbb{R}^s$ and for each $i \in \{1,\dots,s\}$, the coordinate $f^i \in C^3(\mathcal{M})$.*

5. See the Proof of Theorem ?? for details.

3. *The Laplacian matrix* $\tilde{L}_{n,\epsilon}$ *satisfies* $\left|\left|\tilde{\mathcal{L}}_{n,\epsilon_n} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f_n^i\right|\right|_\infty \xrightarrow{p} 0 \ \forall i \in \{1,\ldots,s\}$.

then the estimated embedding metric h_n *associated with* GenericEmbed *converges to the true embedding metric* h *of* $f(\mathcal{M})$*, i.e.* $||h_{ij} - h_{n,ij}||_\infty \xrightarrow{p} 0$*,* $i,j \in \{1,\ldots,s\}$*, where* $h_n = \text{MetricML}(\mathcal{D}_n, \epsilon_n, \text{GenericEmbed})$.

The proof of the theorem is in the Appendix *MMP: below, a subsection?*; here we provide some intution about its requirements. Conditions 1 and 2 above arise from the fact that the embedding metric h_n (or h) is dependent on the embedding f_n or f. Thus, it will make no sense to talk about convergence of h_n if f_n does not converge, or to ask if $\lim h_n$ is the true pushforward metric of f if the limit f of f_n is not an embedding.

The third condition is more technical, and ensures that the interaction between between the algorithm dependent f_n and the Laplacian estimator $\tilde{L}_{n,\epsilon}$ is not pathological, even when both f_n and $\tilde{L}_{n,\epsilon}$ are each, separately, consistent.

The consistency of graph Laplacians is well established in the literature (von Luxburg et al., 2008; Hein et al., 2007; Belkin and Niyogi, 2007; Giné and Koltchinskii, 2006; Ting et al., 2010). We relied on this in Section 5, by defining $\tilde{L}_{n,\epsilon}$, which is a consistent estimate of $\Delta_{\mathcal{M}}$. However, a consistent estimate $\tilde{L}_{n,\epsilon}$ and Condition 1 are not sufficient to guarantee a consistent estimate h_n. To establish consistency, it must also be true that $\forall i,j \in \{1,\ldots,s\}$

$$\tilde{\mathcal{L}}_{n,\epsilon_n}\left(f_n^i(x) - f_n^i(p)\right)\left(f_n^j(x) - f_n^j(p)\right)|_{x=p} \quad \rightarrow \quad \Delta_{\mathcal{M}}(f^i(x) - f^i(p))(f^j(x) - f^j(p))|_{x=p}\,, labeleq : consist (19)$$

and our Condition 3 implies (**??**), as shown in the proof below.

MMP: the theorem can be generalized slightly by letting GenericEmbed *have a different bandwidth than the* ϵ_n *used in the computation of the Laplacian matrix* $\tilde{L}_{n,\epsilon}$.

Thus, Theorem 14 proves that in conjunction with a well behaved embedding algorithm, the MetricML algorithm produces a consistent estimate h_n of the Riemannian metric associated with the coordinates f_n produced by the embedding algorithm.

6.1 Proof of Theorem 14

Proof First a few remarks. Clearly, since $\tilde{h}_n$ is $s \times s$, i.e. finite dimensional, if we prove the elementwise convergence of $\tilde{h}_n$, this will impy the convergence of $\tilde{h}_n$. In general, the pseudoinverse is not a continuous operation. This does not present a problem in our case, since the sequence $\tilde{h}_n$ has rank d for any n, so the probability of $\tilde{h}_n$ being at a discontinuity point of † is zero. *DCPJ: I need a citation about the discontinuity points of †.* Therefore, convergence of $\tilde{h}_n$ guarantees convergence of its pseudoinverse h_n w.p. 1. *MMP: do we need to explain this better? i.e refer to compactness of* $\mathcal{M}$*?...*

It follows that to prove consistency of h_n it is sufficient to prove (**??**) above. *MMP: Please D continue here.*

■

Lemma 15 *If both* f_n *and* g_n *satisfy* $\left|\left|\hat{\mathcal{L}}_{n,t_n} f - \hat{\mathcal{L}}_{n,t_n} f_n\right|\right|_\infty \xrightarrow{p} 0$ *and* $||f_n - f||_\infty \xrightarrow{p} 0$ *with* $f,g \in C^3(\mathcal{M})$ *and* $\mathcal{M}$ *compact, then* $\left|\left|\Delta_{\mathcal{M}} fg - \hat{\mathcal{L}}_{n,t_n} f_n g_n\right|\right|_\infty \xrightarrow{p} 0$ *if* $\hat{\mathcal{L}}_{n,t_n}$ *takes the form*

$$\hat{\mathcal{L}}_{n,t_n} g \quad (x) \equiv \quad \sum_{i=1}^{n} h_n(x, X_i)\left(g(x) - g(X_i)\right)\,,$$

for some positive kernel $h_n(x,y)$.

Proof

$$\begin{aligned}\left|\left|\Delta_{\mathcal{M}} fg - \hat{\mathcal{L}}_{n,t_n} f_n g_n\right|\right|_\infty &\leq \left|\left|\hat{\mathcal{L}}_{n,t_n}\left(fg - f_n g_n\right)\right|\right|_\infty + o_p(1) \\ &\leq \left|\left|\hat{\mathcal{L}}_{n,t_n} g\left(f - f_n\right)\right|\right|_\infty + \left|\left|\hat{\mathcal{L}}_{n,t_n} f_n\left(g - g_n\right)\right|\right|_\infty\end{aligned}$$

So the first term to control is

$$\begin{aligned}\left|\hat{\mathcal{L}}_{n,t_n}\left(g\left(f - f_n\right)\right)(x)\right| &\leq \left|\sum_{i=1}^n h_n(x, X_i)\left(g(x)\left((f - f_n)(x) - (f - f_n)(X_i)\right)\right)\right| \\ &\quad + \left|\sum_{i=1}^n h_n(x, X_i)\left(g(x) - g(X_i)\right)(f - f_n)(X_i)\right| \\ &\leq ||g(x)||_\infty \left|\hat{\mathcal{L}}_{n,t_n}\left(f - f_n\right)(x)\right| \\ &\quad + ||f - f_n||_\infty \left|\hat{\mathcal{L}}_{n,t_n}(g)(x)\right| .\end{aligned}$$

Taking the sup over the sampled points on both sides we get $\left|\left|\hat{\mathcal{L}}_{n,t_n} g\left(f - f_n\right)\right|\right|_\infty \xrightarrow{p} 0$. Similarly,

$$\begin{aligned}\left|\hat{\mathcal{L}}_{n,t_n}\left(f_n\left(g - g_n\right)\right)(x)\right| &\leq \left|\sum_{i=1}^n h_n(x, X_i)\left(f_n(x)\left((g - g_n)(x) - (g - g_n)(X_i)\right)\right)\right| \\ &\quad + \left|\sum_{i=1}^n h_n(x, X_i)\left(f_n(x) - f_n(X_i)\right)(g - g_n)(X_i)\right| \\ &\leq ||f_n||_\infty \left|\hat{\mathcal{L}}_{n,t_n}\left(g - g_n\right)(x)\right| \\ &\quad + ||g_n - g||_\infty \left|\hat{\mathcal{L}}_{n,t_n} f_n\right| .\end{aligned}$$

Again, taking the sup over the sampled points on both sides implies $\left|\left|\hat{\mathcal{L}}_{n,t_n} f_n\left(g - g_n\right)\right|\right|_\infty \xrightarrow{p} 0$ as required. ■

6.2 Consistency of Laplacian Eigenmaps

The goal of this section is to show that the conditions of Theorem 14 can indeed be statisified and, by doing so, provide some hints as to how they can be shown to be satisfied in other algorithms.

But first, we note that, somewhat suprisingly, very few algorithms carry explicit theoretical guarantees that even the first two conditions of Theorem 14 will be satisfied. That is, from the existent convergence results, it does not explicitly follow that in the limit the respective algoritms produce a smooth embedding.

For example, in the case of the Isomap algorithm, the convergence proof focuses on establishing that the graph that estimates the distance between two sampled points converges to the minimizing geodisic distance on the $\mathcal{M}$ (Bernstein et al., 2000). However, the proof does not address the question of whether the empirical embedding f_n is consistent for f or whether f defines a proper embedding. *MMP: except when M is flat? to check...*

Similarly, proofs of consistency for other popular algorithms do not address these two important questions, but instead focus on showing that the linear operators underpinning the algorithms converge to the appropriate differential operators (Coifman and Lafon, 2006; Hein et al., 2007; Giné and Koltchinskii, 2006; Ting et al., 2010). Although this is an important problem in itself, it still

falls short of establishing that $f_n \to f$. The exception to this are the results in (von Luxburg et al., 2008; Belkin and Niyogi, 2007) that prove the convergence of the eigendecomposition of the graph Laplacian to that of the Laplace-Beltrami operator for a uniform sampling density on $\mathcal{M}$.

Hence, the class of algorithms that use the eigenvectors of the Laplace-Beltrami operator to construct embeddings, e.g. LE and DM, can be assumed to be consistent by extension. However, the important question of whether the resulting f is indeed an embedding not adressed. *MMP: connect better* It is not the aim of this paper to address this problem for all algorithms. Nevertheless, for illustrative purposes, we will consider the special case of Laplacian Eigenmaps/Diffusion Maps and show that they can be constructed so as to satisfy condition 2 and 3, while condition 1 is already established (Giné and Koltchinskii, 2006; Belkin and Niyogi, 2007) as discussed above.

The following proves condition 2, i.e. shows that in the limit the LE algorithm produces a true embedding. To prove this, we simply need to show that there is a map Φ constructed from a finite number of eigenfunctions acting as coordinates and ordered from lowest to highest eigenvalues (omitting $\lambda_0 = 0$, for which the eigenfunction is constant), such that Φ is an embedding of $\mathcal{M}$.

Proposition 16 *If $\mathcal{M}$ is a compact manifold with $Vol(\mathcal{M}) = 1$, there exists a finite $s \in \mathbb{N}$ such that the map $\Phi : \mathcal{M} \to \mathbb{R}^s$ where the i^{th} coordinate ϕ_i, $i = 1, \dots, s$, of Φ is the i^{th} eigenfunction of $\Delta_{\mathcal{M}}$, defines a smooth embedding of $\mathcal{M}$ into $\mathbb{R}^s$.*

Proof To construct this embedding, we start from another embedding Ψ of $\mathcal{M}$ into $\mathbb{R}^k$. If $\mathcal{M} \subset \mathbb{R}^k$ for some k, which is generally the case for manifold learning algorithms, then we can use the identity embedding $\Psi(\mathcal{M}) = \mathcal{M}$. An alternative approach is to appeal to Whitney's Embedding Theorem (Lee, 1997), which guarantees that a d-dimensional manifold can be smoothly embedded in $2d$ Euclidean space. Either way, there exists a smooth function Ψ which defines a smooth embedding of $\mathcal{M}$ into $\mathbb{R}^k$.

Since the eigenfunctions of $\Delta_{\mathcal{M}}$ are complete in $L_2(\mathcal{M})$ (Rosenberg, 1997) and $\mathcal{M}$ is assumed to be compact, then each of the coordinates can be expressed in terms of the eigenfunctions of $\Delta_{\mathcal{M}}$ as

$$\psi^i = \sum_{l=0}^{\infty} a_l(i)\phi_l \,.$$

Using seperability, there exist countably many points p_j that are dense in $\mathcal{M}$. For each point p_j, we consider the map $d\Psi_{p_j} : T_{p_j}\mathcal{M} \to T_{\Psi(p_j)}\Psi(\mathcal{M})$. Because Ψ is an embedding, the rank of $d\Psi_{p_j}$ is d. Expressed in local coordinates x around p_j, this means that

$$(\psi^i_{x^1}, \dots, \psi^i_{x^d}) = \sum_{l=0}^{\infty} a_l(i)(\phi_{l,x^1}, \dots, \phi_{l,x^d}) \,, \;\; i = 1, \dots, k$$

spans a linear space of dimension d. Meanwhile, $(\psi^i_{x^1}, \dots, \psi^i_{x^d})$ is a linear combination of $(\phi_{l,x^1}, \dots, \phi_{l,x^d})$ for $l \in \{0, 1, \dots\}$, hence $\text{span}\left((\psi^i_{x^1}, \dots, \psi^i_{x^d}), i \in \{1, \dots, k\}\right) \subseteq \text{span}\left((\phi_{l,x^1}, \dots, \phi_{l,x^d}), l \in \{0, 1, \dots\}\right)$ [6]. This means that at point p_j, there exists $L_j \subset \{0, 1, \dots\}$ defined as $L_j \equiv \{l_{1,j}, \dots, l_{d,j}\}$ such that $\text{span}\left((\phi_{l,x^1}, \dots, \phi_{l,x^d}), l \in L(p_j)\right) \cong \mathbb{R}^d$.

Defining the map $\Phi|_{L_j} \equiv (\phi_1, \phi_2 \dots, \phi_{l_d})$ at p_j, we have that $d\Phi|_{L_j} : T_{p_j}\mathcal{M} \to T_{\Phi|_{L}(p_j)}\Phi|_{L_j}(\mathcal{M})$ is of rank d. By the inverse function theorem, there exists an open set $U_j \subset \mathcal{M}$ with $p_j \in U_j$ so that U_j is diffeomorphic to $\Phi|_{L_j}(U_j)$. Since the points p_j are dense in $\mathcal{M}$, $\cup_{j=1}^{\infty} U_j = \mathcal{M}$ and by compactness of $\mathcal{M}$, there exists a finite subcover $R \subset \{1, 2, \dots\}$ such that $\cup_{j \in R} U_j = \mathcal{M}$. Finally, we define $s = \max_{l \in \cup_{j \in R} L_j} l$ and $\Phi|_m = (\phi_1, \phi_2 \dots, \phi_s)$. Hence, $\Phi \equiv \Phi|_s$ is a smooth embedding of $\mathcal{M}$ into $\mathbb{R}^K$, i.e. $\mathcal{M}$ is diffeomorphic to $\Phi(\mathcal{M})$. ■

Interestingly, this proof could be used to construct an algorithm to estimate what s is appropriate for a given manifold when using the LE. Indeed, one could add eigenvectors to the map Φ until

6. In fact, the two are equal by virtue of $\dim(\mathcal{M}) = d$.

$h_n(p)$ has rank d at each point $p \in \mathcal{M}$. This would then constitute an embedding of $\mathcal{M}$. The main difficulty with such an approach would lie in estimating the rank of $h_n(p)$.

We now need to prove Condition 3. However, in the case of Laplacian Eigenmaps, it is simpler and more illuminating to work directly from (the equivalent) (18)

MMP: uniform sampling density here? Also, the bandwidth epsilon should be mentioned

Lemma 17

$$\left|\left|\Delta_{\mathcal{M}}\phi_i - \tilde{\mathcal{L}}_{n,\epsilon_n}\phi_{n,i}\right|\right|_\infty \xrightarrow{p}$$

Proof

$$\begin{aligned}\left|\left|\Delta_{\mathcal{M}}\phi_i - \tilde{\mathcal{L}}_{n,\epsilon_n}\phi_{n,i}\right|\right|_\infty &= ||\lambda_i\phi_i - \lambda_{n,i}\phi_{n,i}||_\infty \\ &\leq ||\lambda_i\phi_i - \lambda_{n,i}\phi_i||_\infty + ||\lambda_{n,i}\phi_i - \lambda_{n,i}\phi_{n,i}||_\infty \\ &= |\lambda_i - \lambda_{n,i}|\,||\phi_i||_\infty + |\lambda_{n,i}|\,||\phi_i - \phi_{n,i}||_\infty \xrightarrow{p} 0\,,\end{aligned}$$

since $|\lambda_i - \lambda_{n,i}| \xrightarrow{p} 0$ as shown in (Belkin and Niyogi, 2007) and $||\phi_i||_\infty < \infty$ since $\phi_i \in C^\infty(\mathcal{M})$ and by compactness of $\mathcal{M}$. ■

It is easy to show (and left to the reader) that Condition 3 of Theorem 14 follows from the Lemma.

With this we have finished proving that in the case of the Laplacian Eigenmaps embedding algorithm, Theorem 14 can be shown to be true. Although this proof might not easily extend to manifold algorithms other than the Diffusion Maps, we nevertheless establish that it is possible to recover h for at least one embedding. From the point of view of preserving the geometry of $(\mathcal{M}, h)$, this is really all we need. The fact that the Eigenmaps and Diffusion Maps are some of the most interesting embeddings (Belkin et al., 2006; Lu et al., 2007) is merely an added bonus.

These assumptions we made in Proposition 16 and Lemma lem:key allowed us to make use of the key results in (Giné and Koltchinskii, 2006; Belkin and Niyogi, 2007). The assumptions regarding uniform sampling density and volume of $\mathcal{M}$ can be relaxed without trouble, but doing so would introduce considerable technicalities without offering further insight. The compactness condition, on the other hand, is key in proving condition 2, i.e. that the eigenfunctions of the Laplace-Beltrami operator define an embedding and cannot be removed.

7. Experiments

The following experiments on simulated data demonstrate that the MetricML algorithm works and highlight a few of its immediate applications.

7.1 g as a Measure of Local Distortion

The first set of experiments is intended to illustrate the output of the MetricML algorithm. Figure 4 shows the embedding of a 2D hourglass-shaped manifold. Diffusion Maps, the embedding algorithm we used (with $s = 3$, $\lambda = 1$) distorts the shape by excessively flattening the top and bottom. MetricML outputs a $s \times s$ quadratic form for each point $p \in \mathcal{D}$, represented as ellipsoids centered at each p. Practically, this means that the ellipsoids are flat along one direction $T_{f(p)}f(\mathcal{M})^\perp$, and two-dimensional because $d = 2$, i.e. h has rank 2. If the embedding correctly recovered the local geometry, h_p would equal $I_3|_{f(\mathcal{M})}$, the identity matrix restricted to $T_{f(p)}f(\mathcal{M})$: it would define a circle in the tangent plane of $f(\mathcal{M})$, for each p. We see that this is the case in the girth area of the hourglass, where the ellipses are circular. Near the top and bottom, the ellipses' orientation and elongation points in the direction where the distortion took place and measures the amount of (local) correction needed.

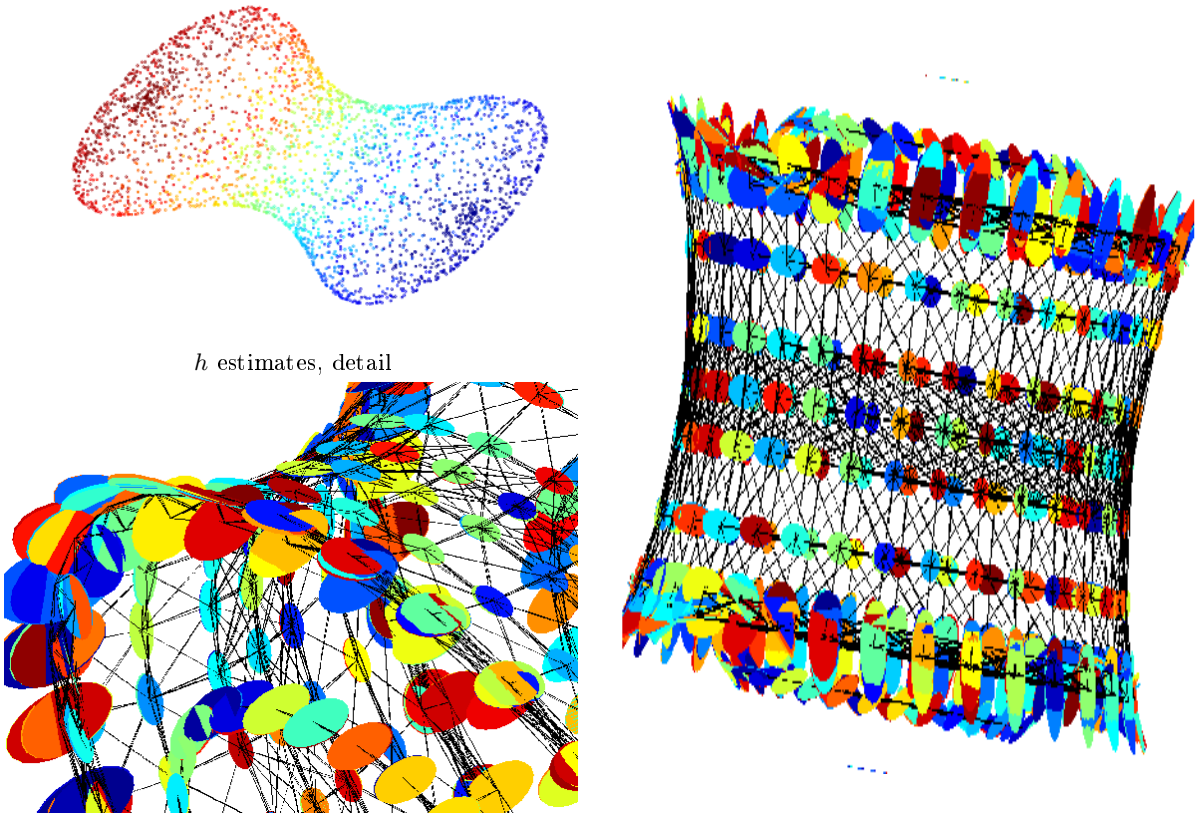
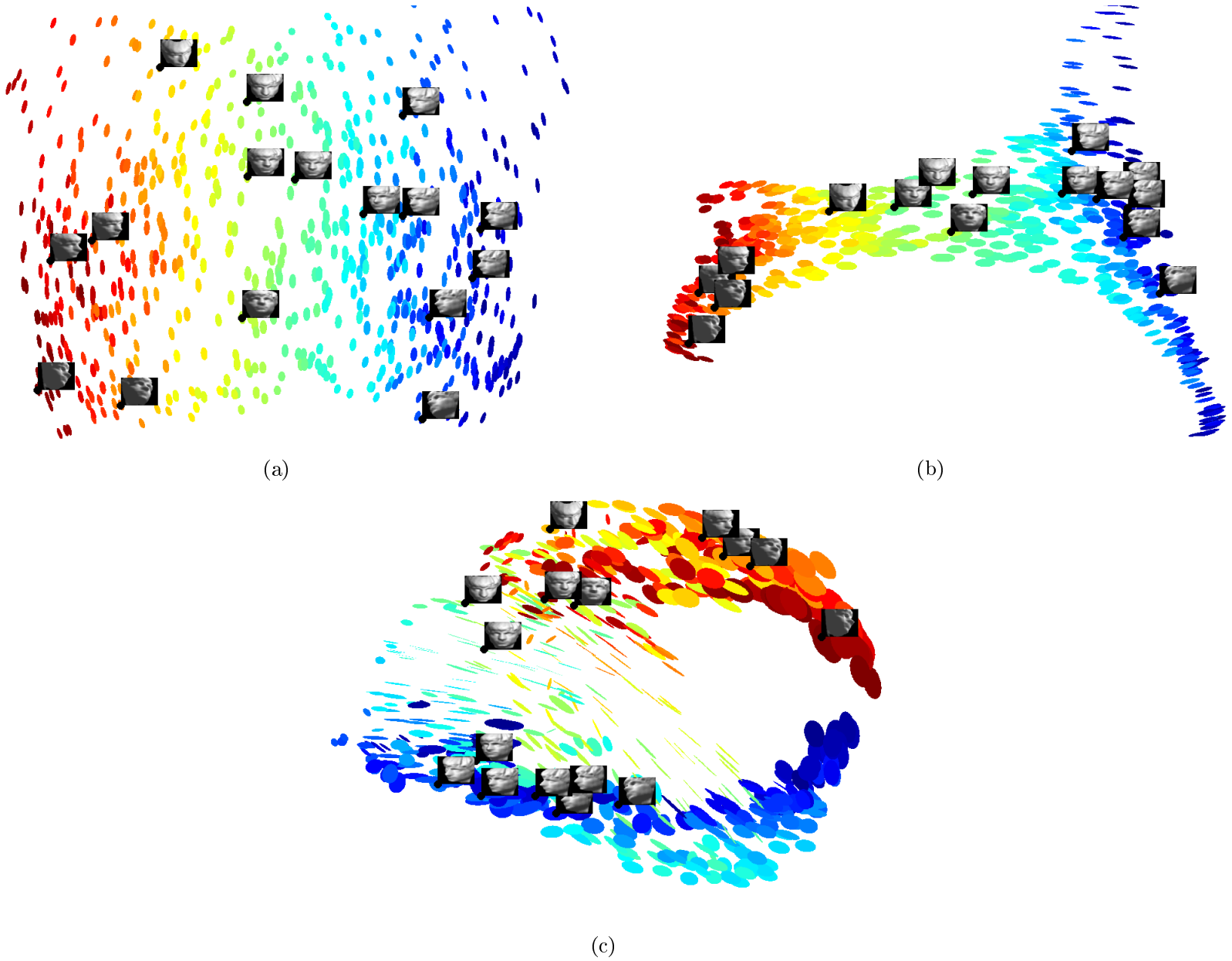


Figure 4: Estimation of h for a 2D hourglass-shaped manifold in 3D space. The embedding is obtained by Diffusion Maps. The ellipses attached to each point represent the embedding metric h estimate for this embedding. At each data point, h_p is a 3×3 symmetric semi-positive definite matrix of rank 2. Near the "girth" of the hourglass, the ellipses are round, showing that the local geometry is recovered correctly. Near the top and bottom of the hourglass, the elongation of the ellipses indicates that distances are larger along the direction of elongation than the embedding suggests. For clarity, in the embedding displayed, the manifold was sampled regularly and sparsely. The black edges show the neigborhood graph $\mathcal{G}$ that was used. For all images in this figure, the color code has no particular meaning.

The more the space is compressed in a given direction, the more elongated the embedding metric "ellipses" will be, so as to make each vector "count for more". Inversely, the more the space is stretched, the smaller the embedding metric will be. This is illustrated in Figure 4.

We constructed the next example to demonstrate how our method applies to the popular Sculpture Faces data set Tenenbaum et al. (2000). The data consist of $n = 698$ $r = 64 \times 64$ gray images of faces. The faces are allowed to vary in three ways: the head can move up and down; the head can move right to left; and finally the light source can move right to left. With only three degrees of freedom, the faces define a three-dimensional manifold in the space of all 64×64 gray images. In other words, we have a three-dimensional manifold $\mathcal{M}$ embedded in $[0, 1]^{4096}$.



(a) (b) (c)

Figure 5: Two-dimensional visualization of the faces manifold, along with embedding. The color corresponds to the left-right motion of the faces. The embeddings shown are: (a) Isomap, (b) LTSA, and Diffusion Maps ($\lambda = 1$) (c) . Note the very elongated ellipses at the top and bottom of the LTSA embedding, indicating the distortions that occurred there.

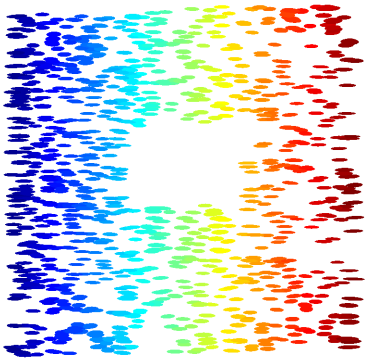


Figure 6: LTSA embedding of the manifold in $\mathbb{R}^2$ along with metric.

Figure 5 demonstrates how the embedding metric h adapts to the given embedding f, measuring at each point the local distortion induced by f.

Our next example, Figure 6, shows the same LTSA embedding of Figure ??. The embedding metric is the same for all points, showing that LTSA recovers the geometry of the manifold up to an affine transformation. Such distortions are easy to correct by applying the inverse linear transformation to the data.

Similarly, for any embedding, there is a simple transformation that turns the embedding into an isometry, locally around a point of interest; this we describe next.

7.2 Locally Isometric Visualization

Visualizing a manifold in a way that preserves the manifold geometry means obtaining an isometric embedding of the manifold in 2D or 3D. This is obviously not possible for all manifolds; in particular, only flat manifolds with intrinsic dimension below 3 can be "correctly visualized" according to this definition. This problem has been long known in cartography: a wide variety of *cartographic projections* of the Earth have been developed to map parts of the 2D sphere onto a plane, and each aims to preserve a different family of geometric quantities. For example, projections used for navigational, meteorological or topographic charts focus on maintaining angular relationships and accurate shapes over small areas; projections used for radio and seismic mapping focus on maintaining accurate distances from the center of the projection or along given lines; and projections used to compare the size of countries focus on maintaining accurate relative sizes (Snyder (1987)).

While the MetricML algorithm is a general solution to preserving intrinsic geometry for all purposes involving calculations of geometric quantities, it cannot immediately give a general solution to the visualization problem described above.

However, it offers a natural way of producing *locally* isometric embeddings, and therefore locally correct visualizations for two- or three-dimensional manifolds. The procedure is based on the transformation of the points that will guarantee that the embedding is the identity matrix.

Algorithm LocallyIsometricVisualization

Input $(f(\mathcal{D}), h(\mathcal{D}))$ Metric Embedding of $\mathcal{D}$, point $p \in \mathcal{D}$

1. Transform coordinates $f^{LI}(p') \leftarrow h(p)^{-1/2} f(p')$ for $p' \in \mathcal{D}$

Display $\mathcal{D}$ in coordinates f^{LI}

As mentioned above, the transformation f^{LI} ensures that the embedding metric at $f^{LI}(p)$ is given by $h^{LI}(p) = I_s$, i.e. the unit matrix[7]. As h varies smoothly on the manifold, h^{LI} will also be close to I_s at points near p, and therefore the embedding will be approximately isometric in a neighborhood of p.

Figures 7, 8 and 9 exemplify this procedure for the Swiss roll with a rectangular hole of Figure 6 embedded respectively by LTSA, Isomap and Diffusion Maps. In these figures, we use the Procrustes method (see e.g. Goldberg and Ritov (2009)) to align the original neighborhood of the chosen point p with the same neighborhood in an embedding. The Procrustes method minimizes the sum of squared distances between corresponding points between all possible rotations, translations and isotropic scalings. The residual sum of squared distances is what we call the *Procrustes dissimilarity* D. Its value is close to zero when the embedding is locally isometric around p.

7.3 Estimation / Recovery ?of Geodesic Distances

The *geodesic distance* $d_{\mathcal{M}}(p, p')$ between two points $p, p' \in \mathcal{M}$ is defined as the length of the shortest curve from p to p' along manifold $\mathcal{M}$. The geodesic distance d being an intrinsic quantity, it should not change with the parametrization.

We performed the following numerical experiment to verify this. We sampled $n = 1000$ points uniformly on a half sphere of radius 1 and selected two reference points p, p' on the half sphere so that their geodesic distance is $\pi/2$. We then calculated the the Isomap, LTSA and DM embeddings on this dataset. All the embedding algorithms used the same 10-nearest neighborhood graph G.

For each embedding, and for the original data, we calculated the *graph distance*, by which we mean the shortest path between the points in $\mathcal{G}$, where the distance is given by $||f(q_i) - f(q_{i-1})||$ with q_i, q_{i-1} neighbors in $\mathcal{G}$, and f the embedding. The graph distance is a surrogate for the geodesic distance, available in the absence of the embedding metric. We also calculated the (naive) Euclidean distance $||f(p) - f(p')||$ for the original data and all embeddings.

Finally, we computed the discrete minimizing geodesic as the shortest path distance in the graph $\mathcal{G}$, where distances between neighbors were corrected using h as follows: *MMP: why $\mathcal{H}$? why not something simpler like d_{ij} or l?*

$$\begin{aligned}\mathcal{H}(q_i, q_{i-1}) &= \frac{1}{2}\sqrt{(f(q_i) - f(q_{i-1}))^t h_{q_i}(f(q_i) - f(q_{i-1}))} \\ &\quad + \frac{1}{2}\sqrt{(f(q_i) - f(q_{i-1}))^t h_{q_{i-1}}(f(q_i) - f(q_{i-1}))}. \end{aligned} \tag{20}$$

This is the discrete analog of the path-length formula (2) for the Voronoi tesselation of the space[8]. Figure 10 shows the manifolds that we used in our experiments, and Table 1 displays the calculated distances.

As expected, the graph distance $d_{\mathcal{G}}$ and the Euclidean distance $||f(p) - f(p')||$ are meaningless for LTSA and DM, but perform reasonably well for Isomap. The "corrected" estimates $\hat{d}$ are quite similar for all embedding algorithms, and they provide a good approximation for the true geodesic distance. It is interesting to note that $\hat{d}$ is the best estimate of the true geodesic distance even for the Isomap, whose focus is specifically to preserve geodesic distances. In fact, the only estimate that is better than $\hat{d}$ for any embedding is the graph distance on the original manifold. In the Supplement, we present additional geodesic distance estimation experiments, with variable levels of noise normal to the manifold.

7. Again, to be accurate, $\tilde{h}_p$ is the restriction of I_s to $T_{f^{LI}(p)} f^{LI}(\mathcal{M})$.
8. By Voronoi tesselation, we mean the partition of the space into sets of points closest to the sampled data points. If the embedding is assumed to be constant on the sets of the partition, then h will change at equidistant points between sampled points.

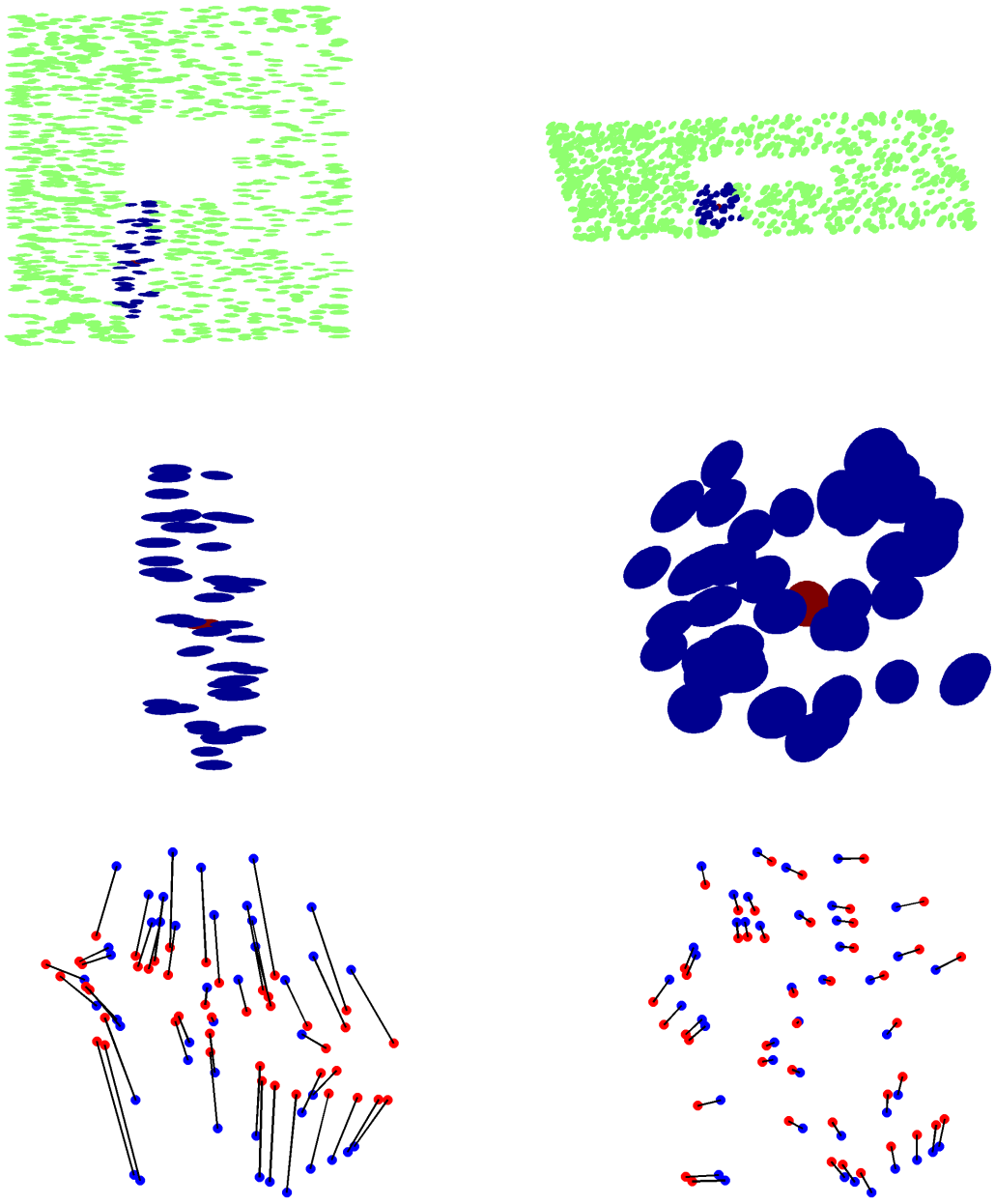


Figure 7: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by LTSA. Top left: LTSA embedding with selected point p (red) and its neighbors (blue). Top right: locally isometric embedding. Middle left: Neighborhood of p for the LTSA embedding. Middle right: Neighborhood of p for the locally isometric embedding. Bottom left: Procrustes between the neighborhood of p for the LTSA embedding and the original manifold projected on $T_p\mathcal{M}$. Bottom right: Procrustes between the locally isometric embedding and the original manifold.

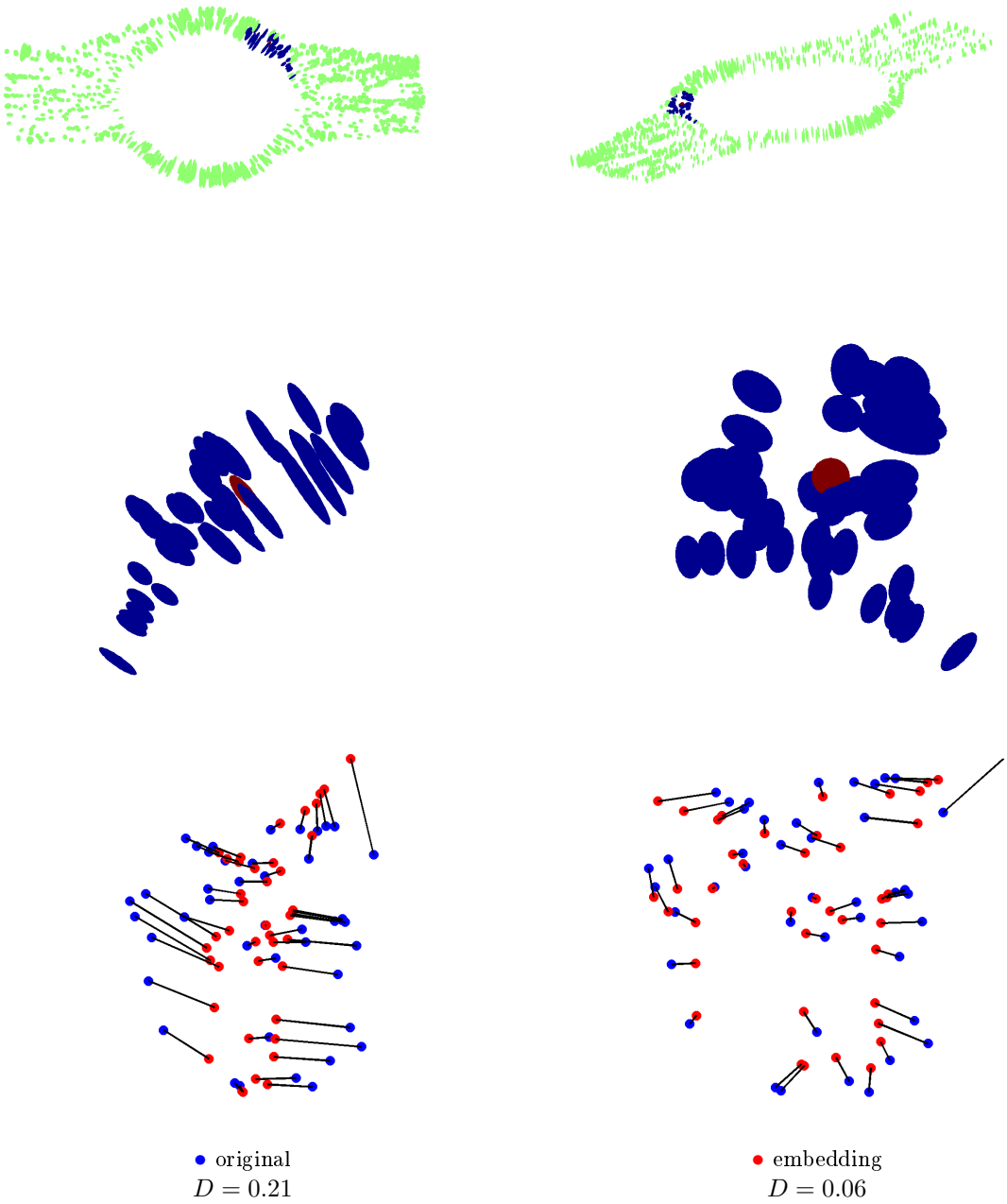


Figure 8: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by Isomap. Top left: Isomap embedding with selected point p (red), and its neighbors (blue). Top right: locally isometric embedding. Middle left: Neighborhood of p for the Isomap embedding. Middle right: Neighborhood of p for the locally isometric embedding. Bottow left: Procrustes between the neighborhood of the p for the Isomap embedding and the original manifold projected on $T_p\mathcal{M}$. Bottom right: Procrustes between the locally isometric embedding and the original manifold.

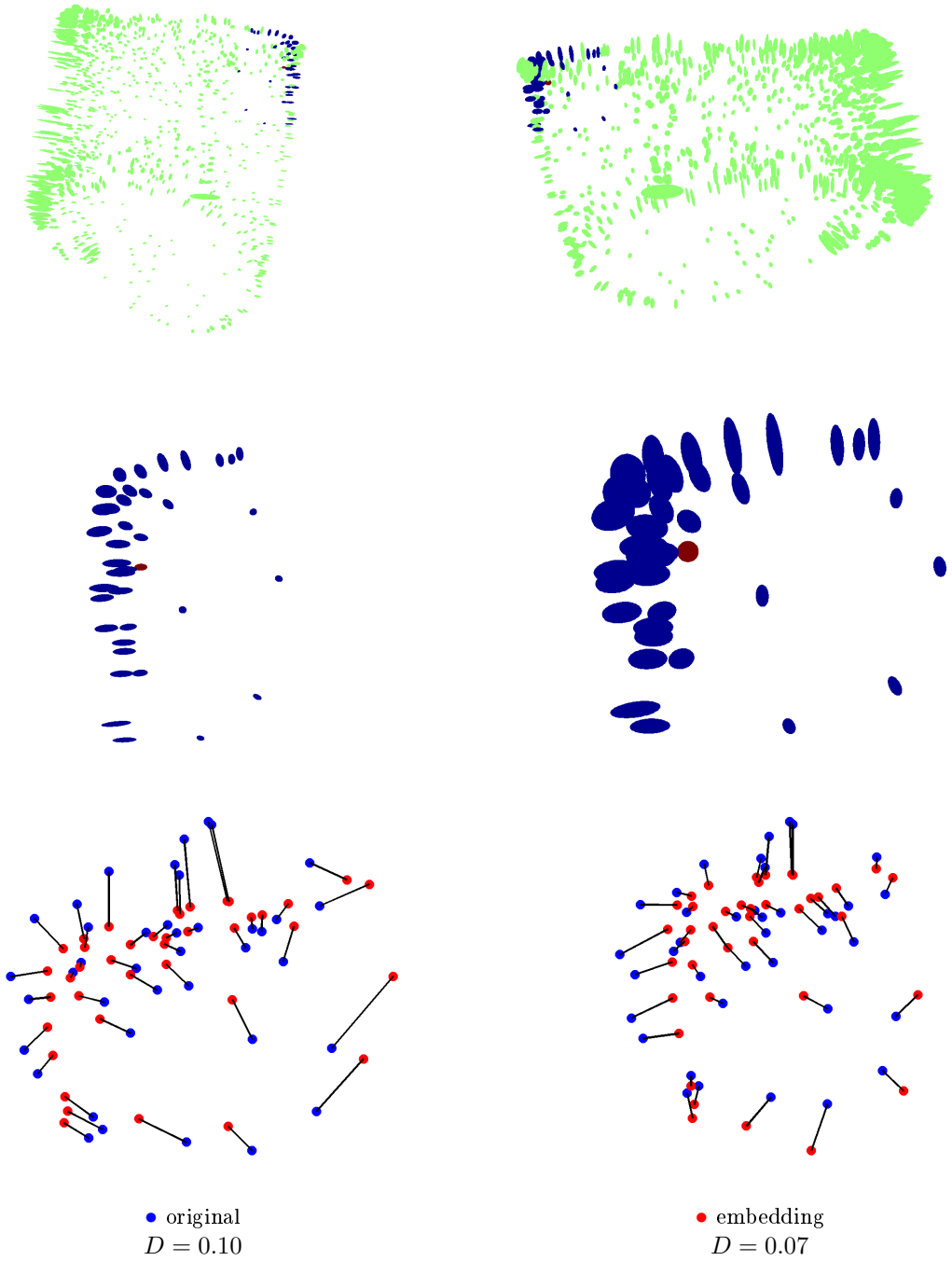
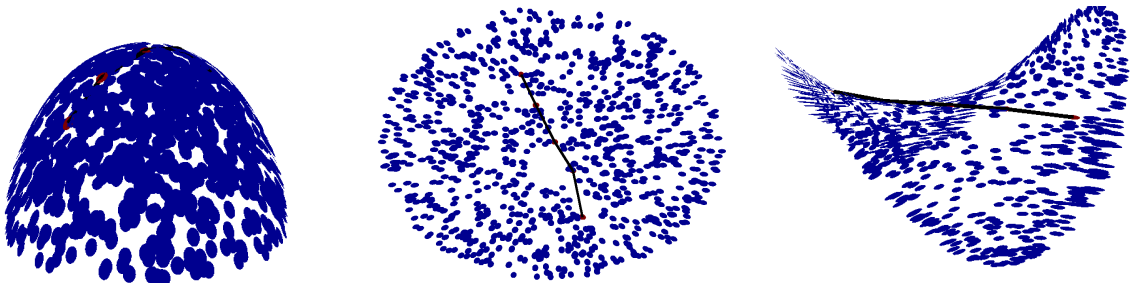


Figure 9: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by Diffusion Maps ($\lambda = 1$). Top left: DM embedding with selected point p (red), and its neighbors (blue). Top right: locally isometric embedding. Middle left: Neighborhood of p for the DM embedding. Middle right: Neighborhood of p for the locally isometric embedding. Bottow left: Procrustes between the neighborhood of the p for the DM embedding and the original manifold projected on $T_p\mathcal{M}$. Bottom right: Procrustes between the locally isometric embedding and the original manifold.



(a) (b) (c)

Figure 10: Manifold and embeddings with h displayed (in blue) used to compute the geodesic distance. Points that were part of the geodesic, including endpoints, are shown in red, while the path is shown in black. The LTSA embedding is not shown here: it is very similar to the Isomap. (a) Original manifold (b) Diffusion Maps (c) Isomap.

MMP: this is a figure we could do with if needed

Embedding	$\|\|f(p) - f(p')\|\|$	Shortest Path $d_{\mathcal{G}}$	Metric $\hat{d}$	$\hat{d}$ Relative Error
Original data	1.412	1.565 ± 0.003	1.582 ± 0.006	0.689 %
Isomap $s = 2$	1.738 ± 0.027	1.646 ± 0.016	1.646 ± 0.029	4.755%
LTSA $s = 2$	0.054 ± 0.001	0.051 ± 0.0001	1.658 ± 0.028	5.524%
DM $s = 3$	0.204 ± 0.070	0.102 ± 0.001	1.576 ± 0.012	0.728%

Table 1: A sample size of $n = 2000$ points was used for all embeddings while the standard deviations were estimated by repeating the experiment 5 times. The relative errors in the last column were computed with respect to the true distance $d = \pi/2 \simeq 1.5708$.

Embedding	Naive Area of W	$\hat{\text{Vol}}(W)$	$\hat{\text{Vol}}(W)$ Relative Error
Original data	$2.63 \pm 0.10^{\dagger}$	2.70 ± 0.10	2.90%
Isomap	$6.53 \pm 0.34^{\dagger}$	2.74 ± 0.12	3.80%
LTSA	8.52e-4 $\pm$ 2.49e-4	2.70 ± 0.10	2.90 %
DM	6.70e-4 $\pm$ 0.464e-04†	2.62 ± 0.13	4.35 %

Table 2: † The naive area/volume estimator is obtained by projecting the manifold or embedding on $T_p\mathcal{M}$ and $T_{f(p)}f(\mathcal{M})$, respectively. This requires manually specifying the correct tangent planes, except for LTSA, which already estimates $T_{f(p)}f(\mathcal{M})$. Similarly to LTSA, $\hat{\text{Vol}}(W)$ is constructed so that the embedding is automatically projected on $T_{f(p)}f(\mathcal{M})$. Here, the true area is 2.658

7.4 Volume Estimation

This experiment demonstrates the use of the Riemannian metric in recovering two-dimensional volumes: areas. We used an experimental procedure similar to the case of geodesic distances, in that we created a two-dimensional manifold, and selected a set W on it. We then estimated the area of this set by generating a sample from the manifold, embedding the sample, and computing the area in the embedding space using a discrete form of (3). The estimation results are given in the table 2.

Here, the estimator $\hat{\text{Vol}}(W)$ performs better than the naive estimator, i.e. projecting on the tangent plane and obtaining the area from the Voronoi tessellation, for all embeddings. In fact, based on the relative error, $\hat{\text{Vol}}(W)$ seems to perform better than the estimator of distances $\hat{d}$. Surprisingly, $\hat{\text{Vol}}(W)$ performs better than the naive estimator used on the original data. This is probably due to the fact that a fair amount of distortion is induced by projecting the manifold on $T_{f(p)}f(\mathcal{M})$ and it is only through the embedding metric that this distortion can be accounted for.

The relative error for LTSA is similar to that for the original data and larger than for the other methods. One possible reason for this is the error in estimating the tangent plane $\mathcal{T}_p\mathcal{M}$, which, in the case of these two methods, is done by local PCA.

8. Experiments with real data

Here we briefly show results of the MetricML algorithm on real data examples where the inverse embedding metric was used to visualize distortion. We also take the opportunity to list a few scientific domains where manifold learning is being used.

The first illustration is in Figure **??**.

In theoretical chemistry, manifold learning methods have been used routinely to summarize and understand the high-dimensional configuration spaces of molecules and materials. The low dimensional representations, often known as *order parameters*, or *collective variables* describe the long-time scale evolution of the molecular dynamics. Figure 11 illustrates an example of such collective variable. The order parameters are used in conjunction with on-line simulations, to facilitate the exploration of the configuration space. The paper **?** reviews this field, and cites more than 15 papers on manifold learning, of which 6 are applications for molecular dynamics. Manifold learning is also used construct low dimensional "maps" from large classes of molecules, with the ultimate goal of predicting their properties, in e.g (Ceriotti et al., 2011). In astronomy, manifold learning is used frequently as a tool to visualize high dimensional observations such as galaxy and star spectra (Daniel et al., 2011). Figure 12 reproduced from **?**, exemplifies such an application. A third domain of where manifold learning has been used for a long time is neuroscience (Cunningham and Yu, 2014; Brown, 2005).

Figure 11: Samples from a molecular dynamics simulation of the Aspirin $C_9H_8O_4$ molecule (Chmiela et al., 2017) in its main energy well, at $T = 500K$ were embedded by the LE algorithm into $s = 2$ dimensions. Left: embedding, colored by the *methyl-carbonyl bond torsion* τ, a quantity of interest to chemists. Middle: the same embedding displaying $\tilde{h}$ as a measure of local distortion. In this case, $\tilde{h}$ indicates that (1) distortions exist but are moderate in the "loop" part of the embedding; (2) the "loop" is narrower than it appears, strongly indicating a one-dimensional manifold parametrized by τ; (3) distortion in the bottom part of the embedding are large and wildly varying, indicating that in this region the embedding geometry bears no ressemblance to the original data.

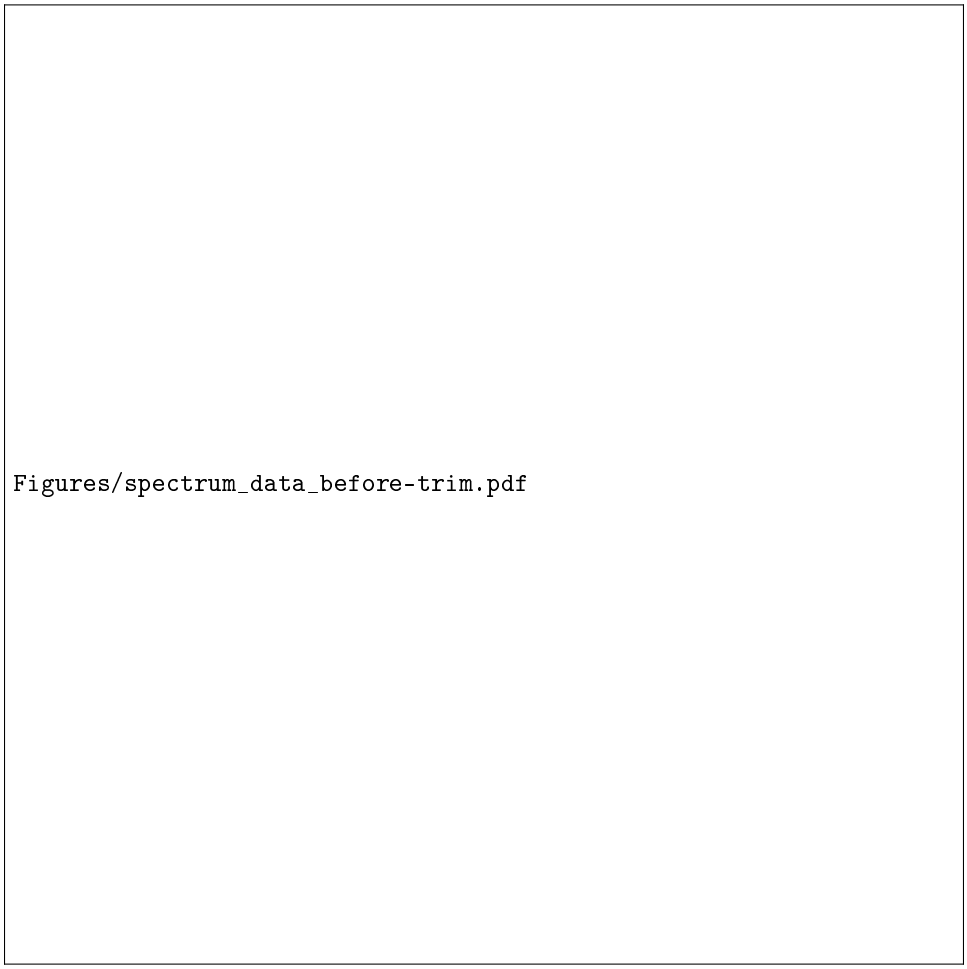


Figure 12: Embedding of main sample of galaxy spectra ($n = 675,000$, $r = 3750$ dimensions) from the Sloan Digital Sky Survey (`www.sdss.org`) into 3 dimensions with the LE algorithm. The ellipses represent the distortion measured by $\tilde{h}$; the color corresponds to the depth coordinate. For clarity, a subsample of $N' = 40,000$ is displayed, from which we show $\tilde{h}$ for about 200 points sampled on the high-density "arms". The large elongated ellipses in the yellow "arm" indicate that this "arm" is several times shorter than it appears.

9. Discussion

Significance Our work departs from the standard manifold learning paradigm. Instead of focusing on finding an embedding that preserves as much as possible of the original data geometry, we allow for any reasonable embedding and then estimate and correct the distortion incurred. Hence, while existing manifold learning algorithms, when faced with the impossibility of mapping curved manifolds to Euclidean space, choose to focus on distances, angles, or specific properties of local neighborhoods and thereby settle for trade-offs, our method allows for dimensionality reduction without sacrificing *any* of these data properties. Of course, this entails recovering and storing more information than the coordinates alone. The MetricML algorithm augments an embedding with the appropriate metric that ensures its faithfulness to the original geometry. The information stored is of order s^2 per point, while the coordinates only require s values per point.

Moreover, our method eliminates the need for users to base their choice of embedding algorithm on the algorithm's capacity to preserve the geometry of the data. Any algorithm can be made to preserve that geometry, so more emphasis can be put on other considerations, such as ease of implementation, running time, flexibility, rates of convergence, or other problem-specific properties of interest. In this way, Metric Learning has the capacity to fundamentally change the way non-linear dimension reduction is carried out in practice. For example, one could use the faster Diffusion Maps method instead of the slower Isomap and still preserve the geodesic distances. This is only an example of the many new possibilities and research questions that our method introduces. We will expand on some of them in the following paragraphs.

One can also use Metric Learning to obtain meaningful comparisons between different embedding algorithms. Each manifold learning algorithm produces its own coordinate system. The same algorithm, applied to different samples from the same distribution, produces outputs that are not easy to compare (or align) other than by visual inspection. This drawback has been highlighted in the literature by (Lin and Zha, 2008), among others. Our method addresses this drawback by granting access to intrinsic, coordinate invariant manifold properties through the estimate of the Riemannian metric, and allowing for the comparison of various algorithms based on these intrinsic quantities. For example, one can quantitatively compare geodesic distance estimates from different algorithms to find which one converges faster or is less sensitive to noise. Thus, by augmenting each algorithm with its Riemannian (pushforward) metric, we have provided a way to unify their outputs.

Multiple charts, multiple dimension choices, noise It is obvious that the Metric Learning framework extends seamlessly to multiple charts. With embedding algorithms based on the (renormalized) Laplacian eigenvectors, like LE and DM, which do not assume that $s = d$, it is practical to obtain a whole range of embeddings with dimensions between d and s^{MAX} by solving a single eigenproblem for s^{MAX} eigenvectors. Thus, embeddings of higher dimensions are obtained by adding new eigenvectors to embeddings of lower dimension. It is easy to see that the $h^\dagger$ pseudo-inverse metric can also be obtained incrementally, by simply applying (11) to the new coordinate vector pairs. The previously computed elements of $h^\dagger$ will remain the same. Recursively updating a $s \times s$ pseudoinverse is $\mathcal{O}(s^2)$ per added dimension (Harville, 1997). *MMP: I wanted to say somethign about recursive methodss for SVS's/ pseudoinverses. Didn't find the right citation -see spectral-temp.bib, but from a little back of the envelope calculation of my own, it looks like the recursive method will still take $O(s^3)$??. Maybe i didn't hit the smartest method..*

Often, the original data often lies *near* a manifold, but not exactly on it. We call this case *noisy*. We do not have theoretical results about the behavior of the Riemannian metric estimate for noisy data; however, empirically, it has been observed that manifold learning with noisy data has a smoothing effect, bringing the data near the embedding of the noiseless manifold. Indeed, our own preliminary experiments, indicate that metric learning is tolerant of noise. In fact, the estimates from noisy data, for low noise, are usually as close to the true distance as the estimates from zero-noise data. *MMP: to add to experiments: how large is the noise compared to sphere radius?*

Applications: Riemannian Relaxation() and bandwidth estimation() are two application of Metric Learning. The former, () proposes to use the estimated embedding metric $\tilde{h}(p)$, in a cost function $\sum_{p\in\mathcal{D}} ||\tilde{h}(p) - I||$, that measures the "total" distortion incurred by embedding f. Then, f is modified incrementally to minimize this cost. If the cost becomes approximatively 0, then an isometric embedding was achieved. In the second paper (), the same distortion is minimized w.r.t. ϵ the kernel bandwidth; the optimal ϵ value w.r.t this criterion is used to construct $\tilde{L}_{n,\epsilon}$.

Acknowledgements

We thank Jack Lee for many fruitful discussions on the topic of manifolds, M. Wittman for creating the software `mani.m` and the numerous authors of manifold learning algorithms who made their code available. Stefan Chmiela provided the aspirin molecular simulation data. Figure 12 was created by James McQueen from data preprocessed by Jake VanderPlas. This research was partly supported by NSF awards IIS-0313339 and IIS-0535100.

References

B. Behmardi and R. Raich. Isometric correction for manifold learning. In *AAAI symposium on manifold learning*, 2010.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15:1373–1396, 2002.

M. Belkin and P. Niyogi. Convergence of laplacians eigenmaps. *NIPS*, 19:129–136, 2007.

M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, December 2006.

M. Belkin, J. Sun, and Y. Wang. Constructing laplace operator from point clouds in rd. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 1031–1040, 2009.

A. Ben-Israel and T. N. E. Greville. *Generalized inverses: Theory and applications*. Springer, New York, 2003.

M. Bernstein, V. deSilva, J. C. Langford, and J. Tennenbaum. Graph approximations to geodesics on embedded manifolds. http://web.mit.edu/cocosci/isomap/BdSLT.pdf, 2000.

I. Borg and P. Groenen. *Modern Multidimensional Scaling: theory and applications*. Springer-Verlag, 2nd edition, 2005.

S.L. et al. Brown. Encoding a temporally structured stimulus with a temporally structured neural representation. *Nature Neuroscience*, 8:1568—-1576, 2005. uses isomap/lle.

Michele Ceriotti, Tribello G. A., and Michele Parrinello. Simplifying the representation of complex free-energy landscapes using sketch-map. *Proceedings of the National Academy of Science, USA*, 108:13023—28, 2011.

Stefan Chmiela, Alexandre Tkatchenko, Huziel Sauceda, Igor Poltavsky, Kristof T. Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science Advances*, March 2017.

R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21 (1):6–30, 2006.

John P. Cunningham and Byron M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 16:1500—-1509, 2014. survey, 3isomap/lle cited.

Scott F. Daniel, Andrew Connolly, Jeff Schneider, Jake Vanderplas, and Liang Xiong. Classification of stellar spectra with local linear embedding. *Astronomical Journal 2011*, 142(6):203,, 2011.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Science*, 100(10):5591–5596, May 2003.

D. L. Donoho and C. Grimes. Image manifolds which are isometric to euclidean space. *Journal of Mathematical Imaging and Vision*, 23(1):5–24, 2005.

D. W. Dreisigmeyer and M. Kirby. A pseudo-isometric embedding algorithm. http://www.math.colostate.edu/~thompson/whit_embed.pdf, retrived June 2010, 2007.

E. Giné and V. Koltchinskii. Empirical Graph Laplacian Approximation of Laplace-Beltrami Operators: Large Sample results. *High Dimensional Probability*, pages 238–259, 2006.

Y. Goldberg and Y. Ritov. Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine Learning*, 77(1):1–25, OCT 2009. ISSN 0885-6125. doi: {10.1007/s10994-009-5107-9}.

Y. Goldberg, A. Zakai, D. Kushnir, and Y. Ritov. Manifold Learning: The Price of Normalization. *Journal of Machine Learning Research*, 9:1909–1939, AUG 2008. ISSN 1532-4435.

David A. Harville. *Matrix algebra from a statistician's perspective*. Springer-Verlag, New York, 1997.

M. Hein, J.-Y. Audibert, and U. von Luxburg. Graph Laplacians and their Convergence on Random Neighborhood Graphs. *Journal of Machine Learning Research*, 8:1325–1368, 2007.

J. M. Lee. *Riemannian Manifolds: An Introduction to Curvature*, volume M. Springer, New York, 1997.

J. M. Lee. *Introduction to Smooth Manifolds*. Springer, New York, 2003.

Binbin Lin, Xiaofei He, Chiyuan Zhang, and Ming Ji. Parallel vector field embedding. *Journal of Machine Learning Research*, 14:2945–2977, 2013.

Tong Lin and Hongbin Zha. Riemannian manifold learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(5):796–809, 2008.

Z. Lu, M. A. Carreira-perpiñán, and C Sminchisescu. People tracking with the laplacian eigenmaps latent variable model. In *Neural Information Processing Systems*, 2007.

J. McQueen, M. Meila, J. VanderPlas, and Z. Zhang. Megaman: Scalable manifold learning in python. *Journal of Machine Learning Research*, 17, 2016.

J. Nash. The imbedding problem for Riemannian manifolds. *Ann. Math.*, 63:20–63, 1956.

Parikshit Ram, Dongryeol Lee, William March, and Alexander G. Gray. Linear-time Algorithms for Pairwise Statistical Problems. In *Advances in Neural Information Processing Systems (NIPS) 22 (Dec 2009)*. MIT Press, 2010.

S. Rosenberg. *The Laplacian on a Riemannian Manifold*. Cambridge University Press, 1997.

A. Singer. From graph to manifold laplacian: the convergence rate. *Applied and Computational Harmonic Analysis*, 21(1):128–134, 2006.

J. P. Snyder. *Map Projections: A Working Manual*. United States Government Printing, 1987.

J. Tenenbaum, V. deSilva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

D. Ting, L Huang, and M. I. Jordan. An analysis of the convergence of graph laplacians. In *ICML*, pages 1079–1086, 2010.

Nakul Verma. Distance preserving embeddings for general n-dimensional manifolds. *Journal of Machine Learning Research*, 14:2415–2448, 2013.

U. von Luxburg, M. Belkin, and O. Bousquet. Consistency of spectral clustering. *Annals of Statistics*, 36(2):555–585, 2008.

K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90, 2006. ISSN 0920-5691. 10.1007/s11263-005-4939-z.

T. Wittman. MANIfold learning matlab demo. http://www.math.umn.edu/~wittman/mani/, retrived June 2010, 2005.

H. Zha and Z. Zhang. Isometric embedding and continuum isomap. In Tom Fawcett and Nina Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003)*, pages 864–871. AAAI Press, 2003.