Journal of Machine Learning Research 0 (2011) 0-00 Submitted 0/00; Published 00/00

Metric Learning on Manifolds

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: Lawrence Saul

Abstract

In recent years, manifold learning has become increasingly popular as a tool for performing non-linear dimensionality reduction. This has led to the development of numerous algorithms of varying degrees of complexity that aim to recover manifold geometry using either local or global features of the data.

Building on the Laplacian Eigenmap framework, we propose a new paradigm that offers a guarantee, under reasonable assumptions, that any manifold learning algorithm will preserve the geometry of a data set. Our approach is based on augmenting the output of embedding algorithms with geometric information embodied in the Riemannian metric of the manifold. The Riemannian metric allows us to compute geometric quantities (such as angle, length, or volume) for any coordinate system or embedding of the manifold. This geometric faithfulness, which is not guaranteed for most algorithms, allows us to define geometric measurements that are independent of the algorithm used, and hence move seamlessly from one algorithm to another. In this work, we provide an algorithm for estimating the Riemannian metric from data, consider its consistency, and demonstrate the advantages of our approach in a variety of examples.

1. Introduction

When working with large sets of high-dimensional data, one is regularly confronted with the problem of tractability and interpretability of the data. An appealing approach to this problem is the method of dimensionality reduction: finding a low-dimensional representation of the data that preserves all or most of the important "information". One popular idea for data 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 further statistical analysis on the lower dimensional representation of the data.

A common technique for performing dimensionality reduction is Principal Component Analysis, which assumes that the low-dimenisional manifold is an affine space. The affine space requirement is generally violated in practice and this has led to the development of more general techniques which perform non-linear dimensionality reduction. Although not all non-linear dimensionality reduction techniques are based on the manifold hypothesis, manifold learning has been a very popular approach to the problem. This is in large part due to the easy interpretability and mathematical elegance of the manifold hypothesis.

The popularity of manifold learning has led to the development of numerous algorithms that aim to recover the geometry of the low-dimensional manifold $\mathcal{M}$ using either local or global features of the data. These algorithms are of varying degrees of complexity, but all have important shortcomings that have been documented in the literature (Goldberg et al., 2008; Wittman, 2005). Two important

criticisms are that 1) the algorithms fail to recover the geometry of the manifold in many instances and 2) no coherent framework yet exists in which the multitude of existing algorithms can easily be compared and selected for a given application.

It is customary to evaluate embedding algorithms by how well they "recover the geometry", i.e. preserve the important information of the data manifold, and much effort has been devoted to finding embedding algorithms that do so. While there is no uniformly accepted definition of what it means to "recover the geometry" of the data, we give this criterion a mathematical interpretation, equating it with the concepts of *isometry* and *isometric embedding*. Not surprisingly given the criticisms noted above, we demonstrate that the majority of manifold learning algorithms output embeddings that are not isometric.

Assuming that recovering the geometry of the data is an important goal, we offer a new perspective: rather than 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.

Indeed, the information necessary for reconstructing the geometry of the manifold is embodied in its Riemannian metric. Thus, 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. The advantage of recovering g is that it then permits the computation of all geometric values of interest, such as angles, distances and volumes, on $\mathcal{M}$.

The core of our paper is the demonstration of how to obtain the Riemannian metric from the mathematical, algorithmic and statistical points of view. It is discussed in Section 5, following a brief discussion of the literature and an introduction to the Riemannian metric in Sections 2 and 3. We then explore the recovery of the pushforward of the Riemannian metric and develop and algorithm for its estimation in Sections 4 and 5. Finally, we consider the consistency of our estimator in Section 6 and offer some examples and applications in Section 8.

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.

For completeness, we first give a brief survey of manifold learning algorithms, starting with a well-known method for linear dimensionality reduction: Principal Components Analysis. PCA is a simple but very powerful technique that projects data onto a linear space of a fixed dimension that explains the highest proportion of variability in the data. It does so by performing an eigendecomposition of the data correlation 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 can only recover only affine subspaces accurately.

In contrast to linear techniques, manifold learning algorithms assume that the data lies near or along a non-linear, smooth, hyper-surface of dimension d called the *data manifold* $\mathcal{M}$, embedded in the original high-dimensional space $\mathbb{R}^r$ with $d \ll r$, and attempt to uncover this low-dimensional $\mathcal{M}$. 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, generally speaking, a *manifold learning* or *manifold embedding* algorithm is a method of non-linear dimension reduction. Its input is a set of points $\mathcal{D} = \{p_1, \dots p_n\} \subset \mathbb{R}^r$, where r is typically high. These are assumed to be sampled from a low-dimensional manifold $\mathcal{M} \subset \mathbb{R}^r$ and are mapped into vectors $\{f(p_1), \dots f(p_n)\} \subset \mathbb{R}^s$, 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.

Most existing manifold learning algorithms 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. Both k and ϵ act as smoothing/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 their results in our implementation.

2.1 Existing Algorithms

Without attempting to give a thorough overview of the existing manifold learning algorithms, we discuss two main categories of algorithms. One category uses only local information, embodied in $\mathcal{G}$, to construct the embedding. Local Linear Embedding (LLE) (Saul and Roweis, 2003), Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2002), Diffusion Maps (DM) (Coifman and Lafon, 2006), and Local Tangent Space Alignment (LTSA) (Zhang and Zha, 2004) are in this category.

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.

We now provide a short overview of each of these algorithms.

- 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.

- 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 however, 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 $\mathcal{M}$. Unfortunately, this is not true for all embedding algorithms: more often then not, the inverse map for out-of-sample points is not easy to infer.

- ISOMAP: The isomap is an example of a non-linear global algorithm. The idea is to embed $\mathcal{M}$ in $\mathbb{R}^s$ using the geodesic distances between points. The algorithm first constructs $\mathcal{G}$ by connecting the k nearest neighbors of each point and computes the distance between neighbors. Dijkstra's algorithm is then applied to the resulting local distance graph in order to find all shortest path distances between pairs of points. These are taken as approximations to the geodesic distances. The final step consists in embedding the data using Multidimensional Scaling (MDS) on the computed distances between points. 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 estimated geodesic distances.

- MDS: For the sake of completeness, and to aid in understanding the Isomap, we also provide a short description of MDS. MDS is a linear method that finds an embedding into $\mathbb{R}^s$ using dissimilarities (generally distances) between data points. Although there is more than one flavor of MDS, they all revolve around minimizing an objective function based on the difference between the dissimilarities and the distances computed from the resulting embedding.

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 $V \subset \mathbb{R}^d = \{(x^1, ..., x^d) \in \mathbb{R}^d | x^1 \geq 0\}$. 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, 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 $\mathbb{R}^3$. We will return to this observation shortly.

The second remark is that the coordinate chart(s) are not uniquely defined, and that there are infinitely many atlases for the same manifold $\mathcal{M}$ (Lee, 2003). These topological issues need to be taken into account when setting oneself the task of manifold learning. Unfortunately, they are often overlooked.

Standard manifold learning algorithms create a single coordinate chart. Thus, they can only succeed in capturing the data manifold if that manifold admits a global chart. This limitation is understood implicitly when an algorithm is applied. To overcome it, one often maps the data into $s > d$ dimensions (for instance, circles are mapped to $s = 2$ coordinates, spheres to $s = 3$ coordinates, and so on).

The mathematical grounds for mapping the data into $s > d$ dimensions are centered on the concept of *embedding*, which requires the introduction of a few important concepts. 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.

2.3 Consistency

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

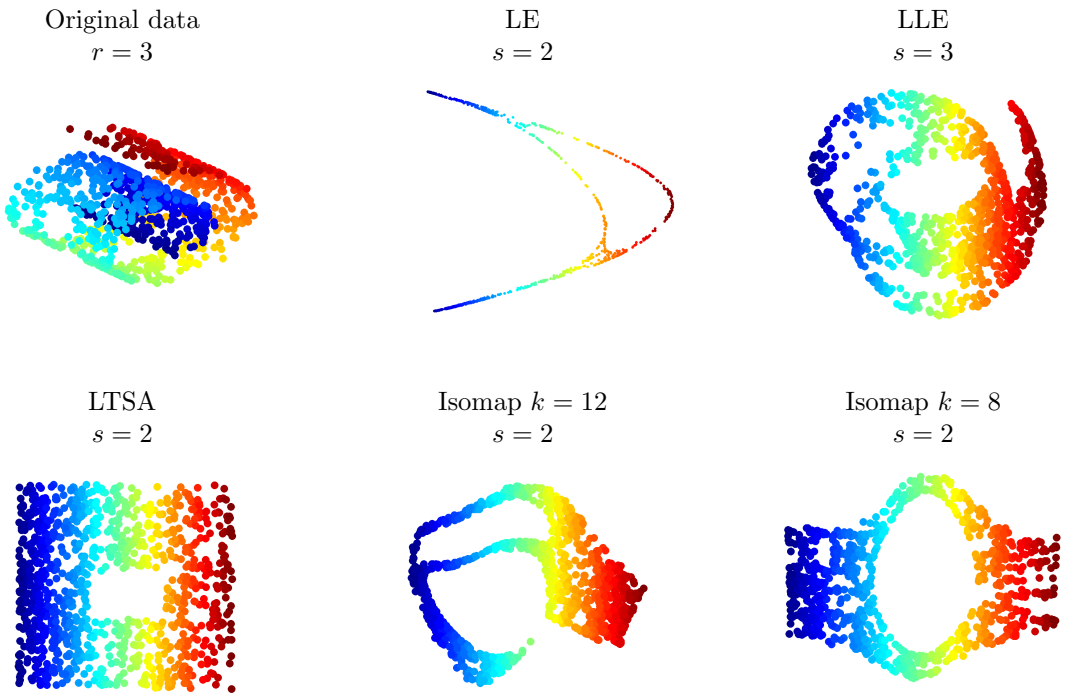


Figure 1: Manifold learning algorithms distort the geometry of the data. The classical "Swiss roll" example is shown here embedded via a variety of manifold learning algorithms. For clarity, the original data is in $r = 3$ dimensions; it is obvious that adding extra dimensions does not affect the resulting embeddings.

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.

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. 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 in general.

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 (defined in Section 4) 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.

In view of these results, we will assume when necessary that an embedding algorithm is consistent and in the limit produces a smooth embedding. Nonetheless, obtaining a consistent estimator for the Riemannian metric will require us to revisit the issue of consistency in Section 6. We now turn to the next desirable property, one for which negative results abound.

2.4 Manifold Geometry Preservation

Having a consistent smooth mapping from $f : \mathbb{R}^r \to \mathbb{R}^s$ guarantees that neighborhoods in the high dimensional ambient space will be mapped into neighborhoods in the embedding space 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 1), 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, angles between curves, and other geometric quantities. 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 1 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 $T\mathcal{M}_p$ 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 angle between two vectors $\cos(\theta) = \frac{<u,v>_g}{||u||||v||}$.

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).

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.

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 intuitive 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 by δ_r, δ_s. In matrix form, these are represented by unit matrices[2]. 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

2. 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 $Tf(\mathcal{M})$.

method involves tightly wrapping the embedding around extra dimensions, which, as observed by (Dreisigmeyer and Kirby, 2007), would not be stable numerically.

Practically, however, as it was shown in section 2.4, manifold learning algorithms do not generally define isometric embeddings. 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, 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.

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 The Laplace-Beltrami Operator and g

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 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 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, with $df_p \equiv J$, $g \equiv G$ and $h \equiv H$, (6) becomes

$$u^t J^t H J v \;=\; u^t G v \tag{7}$$

Hence,

$$H \;\equiv\; \left(J^t\right)^{\dagger} G J^{\dagger} \tag{8}$$

In particular, when $\mathcal{M} \subset \mathbb{R}^r$, with metric inherited from the ambient Euclidean space, as is often the case for manifold learning, we have that $G = \Pi^t I_r \Pi$, where I_r is the Euclidean metric in $\mathbb{R}^r$ and Π is the orthogonal projection of $v \in \mathbb{R}^r$ onto $T_p\mathcal{M}$. Hence, the embedding metric h can then be expressed as

$$H(p) = \left(J^t\right)^{\dagger} \Pi(p)^t I_r \Pi(p) J^{\dagger}. \tag{9}$$

The constraints on h mean that h is symmetric semi-positive definite (positive definite on $T_p f(\mathcal{M})$ and null on $T_p f(\mathcal{M})^{\perp}$, as one would hope), rather than symmetric positive definite like g.

One can easily verify that h satisfies the following proposition:

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

$$< df_p(u), df_p(v) >_{h(f(p))} = < df_p^\dagger \circ df_p(u), df_p^\dagger \circ df_p(v) >_{g(p)} = < u, v >_{g(p)} \tag{10}$$

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

$$< u, v >_{h(f(p))} = < df_p^\dagger(u), df_p^\dagger(v) >_{g(p)} = < 0, df_p^\dagger(v) >_{g(p)} = 0 . \tag{11}$$

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*

$$\tilde{h}^{ij} = \Delta_\mathcal{M} \frac{1}{2}\left(f^i - f^i(p)\right)\left(f^j - f^j(p)\right)|_{f^i = f^i(p), f^j = f^j(p)} \tag{12}$$

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, with $J \equiv df_p$, $G \equiv g(p)$, $H \equiv h$, $\tilde{H} \equiv \tilde{h}$, the above results take the form

$$g^{kl} \frac{\partial f^i}{\partial x^l} \frac{\partial f^j}{\partial x^k} = (JG^{-1}J^t)_{ij} = \tilde{H}_{ij} . \tag{13}$$

Hence, $\tilde{H} = JG^{-1}J^t$ and it remains to show that $H = \tilde{H}^\dagger$, i.e. that

$$\left(J^t\right)^\dagger G J^\dagger = \left(JG^{-1}J^t\right)^\dagger . \tag{14}$$

This is obviously straightforward for square invertible matrices, but if $d < s$, this might not be the case. Hence, we need an additional technical fact: guaranteeing that

$$(AB)^\dagger = B^\dagger A^\dagger \tag{15}$$

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^{-1} has full rank, J has full column rank, and J^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 (15) repeatedly to $JG^{-1}J^t$, implicitly using the fact that $\left(G^{-1}J^t\right)$ has full row rank since G^{-1} has full rank and J has full row rank, proves that h is the pseudoinverse of $\tilde{h}$. ■

Computing the pseudoinverse of $\tilde{h}$ generally means performing a Singular Value Decomposition SVD. It is interesting to note that this decomposition offers very useful insight 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 8 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 acheive 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

Another fortuitous consequence of computing the pseudoinverse is that 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 what the embedding is doing to specific regions of the manifold and it also prescribes a simple linear transformation of the embedding f that makes it locally isometric to $\mathcal{M}$ with respect to the inherited metric δ_s. This latter attribute will be explored in more detail in Section 8.

5. Recovering the Riemannian Metric: The Algorithm

The results in the previous section apply to 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 first discuss the discrete version of the problem, and then present our algorithm for the estimation procedure. Throughout, we assume that the intrinsic dimension d has been estimated, and the embedding dimension $s \geq d$ is already selected.

5.1 Discretized Problem

Prior to explaining our method for estimating h for an embedding algorithm, it is important to discuss the discrete version of the problem.

As briefly explained in Section 2, 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 a compact Riemannian manifold. These points are assumed to be an i.i.d. sample with distribution $p(x) = e^{-U(x)}$ on $\mathcal{M}$, which is absolutely continuous with respect to the Lebesgue measure on $\mathcal{M}$. From this sample, manifold learning algorithms construct a map $f_n : \mathcal{D} \to \mathbb{R}^s$, which, if the algorithm is consistent, will converge to an embedding $f : \mathcal{M} \to \mathbb{R}^s$.

Once the map is obtained, we go on to define the embedding metric h_n. Naturally, it is relevant to ask what it means to define the embedding metric h_n and how one goes about constructing it. Since f_n is defined on the set of points $\mathcal{D}$, h_n will be defined as a positive semidefinite matrix over $\mathcal{D}$. With that in mind, we can hope to construct h_n by discretizing equation (12). In practice, this is acheived by replacing f with f_n and $\Delta_{\mathcal{M}}$ with some discrete estimator $\tilde{\mathcal{L}}_{n,\epsilon}$ that is consistent for $\Delta_{\mathcal{M}}$.

In what sense the estimators f_n, h_n, and $\tilde{\mathcal{L}}_{n,\epsilon}$ can be said to be consistent will be discussed in Section 6 below. In the meantime, we still need to clarify how to obtain $\tilde{\mathcal{L}}_{n,\epsilon}$. As mentioned previously, there exist many methods for obtaining an estimator of $\Delta_{\mathcal{M}}$. The most common, and the one we favor here, is to construct a weighted neigborhood graph $\mathcal{G}_n$ on the sampled points $\mathcal{D}$

by applying the *heat kernel* to the distance between points (Belkin and Niyogi, 2002). Specifically, the weight between nodes p and p' of $\mathcal{G}_n$ is given by $k_\epsilon(p,p') = \exp(-||p-p'||^2/\epsilon)$, giving rise to the adjacency matrix $K_n = [k_\epsilon(p,p')]_{p,p'\in\mathcal{D}}$. If the sampling density $p(x)$ is uniform over $\mathcal{M}$, then one can immediately construct the random walk graph Laplacian (Hein et al., 2007):

$$\mathcal{L}_{n,\epsilon} \equiv \frac{I_n - T_n^{-1}K_n}{\epsilon}\,, \tag{16}$$

where T_n is the diagonal matrix of outdegrees, i.e. $T_n = \mathrm{diag}\{t(p), p\in\mathcal{D}\}$ with $t(p) = \sum_{p'\in\mathcal{D}} K_n(p,p')$. The random walk graph Laplacian $\tilde{\mathcal{L}}_{n,\epsilon} f$ 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 $p(x)$ is not uniform, or simply unknown (as is generally the case), it is necessary to renormalize the adjacency matrix K_n before obtaining $\tilde{\mathcal{L}}_{n,\epsilon}$. 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 $\tilde{K}_n = T_n^{-1}K_nT_n^{-1}$ for $\mathcal{G}_n$, along with a new outdegree matrix $\tilde{T}_n = \mathrm{diag}\{\tilde{t}(p), p\in\mathcal{D}\}$ with $\tilde{t}(p) = \sum_{p'\in\mathcal{D}} \tilde{K}_n(p,p')$. The associated random walk graph Laplacian is then given by

$$\tilde{\mathcal{L}}_{n,\epsilon} \equiv \frac{I_n - \tilde{T}_n^{-1}\tilde{K}_n}{\epsilon}\,. \tag{17}$$

This new random walk graph Laplacian is consistent for $\Delta_{\mathcal{M}}$ irrespective of the sampling density $p(x)$, provided $p(x)\in C^2(\mathcal{M})$ and $p(x)$ is bounded away from zero, i.e. $p(x) > l$ for all $x\in\mathcal{M}$ with $l > 0$.

We note that the heat kernel is not the only option available for defining the adjacency matrix K_n (see for example (Ting et al., 2010)). An important variation to the heat kernel is the truncated heat kernel, where $k_\epsilon(p,p') = 0$ if $||p-p'||^2 > \epsilon$ or some other multiple of ϵ. This procedure really defines two graphs, the $0,1$ neighborhood graph indexed by $||p-p'||^2 > \epsilon$ and the similarity graph with weights $\exp(-||p-p'||^2/\epsilon)$ when two points are neighbors.

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

With the discrete analogue to (12) clarified, we are now ready to introduce the central algorithm of this paper.

5.2 The MetricLearn Algorithm

Figure 2 contains the algorithm we use to compute the embedding metric of the manifold in the coordinates of the chosen embedding. Each step is introduced in some detail below.

As discussed above, the first step, common to many manifold learning algorithms, consists of finding a set of neighbors for each data point p, and evaluating a similarity $K_n(p,p')$ for each pair of points p,p' that are neighbors. Here, we stress the point that the bandwidth parameter ϵ determines the size of the neighborhood for each point, and, hence, the density of the graph. The adjacency/similarity matrix K_n represents the heat kernel applied to point differences $||p-p'||^2$.

Second, we use the similarity matrix K_n to obtain $\tilde{\mathcal{L}}_{n,\epsilon}$ (17), 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)). Again, we renormalize K_n in order for $\tilde{\mathcal{L}}_{n,\epsilon}$ to be a consistent estimator of $\Delta_{\mathcal{M}}$, even in the presence of non-uniformly sampled data.

Our third step calls for the selection of an embedding for the manifold. This can be done using any one of the many existing manifold learning algorithms (GenericEmbed), such as the Laplacian Eigenmaps, Isomap or Diffusion Maps, provided that it satisfies the requirements we discuss in 6. In short, the algorithm must have a well-defined h, i.e. be an embedding of $\mathcal{M}$, and h_n must be consistent for h.

At this juncture, we note 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, and LLE, use a neighborhood graph and/or similarities in order to obtain an embedding. In addition, Diffusion Maps and Eigemaps obtain an embedding for the eigendecomposition $\tilde{\mathcal{L}}_{n,\epsilon}$ 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 of the chosen embedding. 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. In practice, $h_n^\dagger$ will not have rank d and so it is important, when computing the pseudo-inverse, to set to zero the $s-d$ smallest singular values of $h_n^\dagger$. This is the key reason why s and d need to be fixed in advance. Failure to set the smallest singular values to zero will mean that h_n will dominated by noise. Although estimating d is outside the scope of this work, it is intersting to note that the singular values of $h_n^\dagger$ may offer a window into how to do this by looking for a "singular value gap".

Note also that in step 4 (a), we use the symbol $\cdot$ to refer to the elementwise product between two vectors. Specfically, 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 $\mathcal{D} \subset \mathcal{M}$.

In summary, the principal novelty in our MetricLearn algorithm is its last step: the estimation of the embedding metric h. The embedding metric establishes a direct correspondence between geometric computations performed using $(f(\mathcal{M}), h)$ and those performed directly on $(\mathcal{M}, g)$ *for any embedding*. Thus, once augmented with h, all embeddings become geometrically equivalent.

5.3 Computational Complexity

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 can be considered constant, then the time for this step is linear in n.

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 $\mathcal{O}(n \times s)$ computations. This step, or a variant thereof, is also a component of many embedding algorithms. A notable exception is Isomap, which iterates over this step, and is thus significantly slower than e.g Diffusion Map.

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 order n operations (assuming the $\tilde{\mathcal{L}}_{n,\epsilon}$ matrix is sparse) times $s \times s$ entries, for a total of $s^2 n$ operations. The n SVD and pseudoinverse calculations take order s^3 operations.

Thus, finding the Riemannian metric makes a small contribution to the computational burden of finding the embedding. The overhead is linear in the data set size n and polynomial in the embedding dimension s. For s with small values, we can consider the overhead as linear.

Algorithm MetricLearn

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
   (a) For each point $p \in \mathcal{D}$ compute $t_n(p) = \sum_{p'\in\mathcal{D}} K_n(p,p')$; and form the diagonal matrix $T_n = \text{diag}\{\, t_n(p),\, p \in \mathcal{D}\}$
   (b) Let $\tilde{K}_n = T_n^{-1} S_n T_n^{-1}$
   (c) Let $\tilde{t}_n(p) = \sum_{p'\in\mathcal{D}} \tilde{K}_n(p,p')$, $\tilde{T} = \text{diag}\{\, \tilde{t}_p, p \in \mathcal{D}\}$
   (d) Define $\tilde{\mathcal{L}}_{n,\epsilon}; = \left(I_n - \tilde{T}_n^{-1}\tilde{K}_n\right)/\epsilon$.
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
$$[f_n(p)]_{p\in\mathcal{D}} = \text{GenericEmbed}(\mathcal{D},\, s, \epsilon_n)$$
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[a]
$$\tilde{h}_n^{ij} = \frac{1}{2}\left[\tilde{\mathcal{L}}_{n,\epsilon}(f_n^i \cdot f_n^j) - f_n^i \cdot (\tilde{\mathcal{L}}_{n,\epsilon} f_n^j) - f_n^j \cdot (\tilde{\mathcal{L}}_{n,\epsilon} f_n^i)\right] \tag{18}$$
   (b) For each data point $p \in \mathcal{D}$, form the matrix $\tilde{h}_n(p) = [\tilde{h}^{ij}(p)]_{i,j\in 1,\ldots,s}$. The embedding metric at p is then given by $h_n(p) = \tilde{h}_n^\dagger(p)$.

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

a. Equation (18) is trivially equivalent to applying equation (12) to all the points of $p \in \mathcal{D}$ at once.

Figure 2: The MetricLearn Algorithm.

6. Consistency of the Riemannian Metric

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). Thus, obtaining a consistent estimate $\tilde{\mathcal{L}}_{n,\epsilon}$ of $\Delta_{\mathcal{M}}$ is straightforward; in fact, we relied on this in Section 5 above. However, a consistent estimate $\tilde{\mathcal{L}}_{n,\epsilon}$ is not sufficient to guarantee a consistent estimate of the embedding h. Three conditions are required to guarantee the consistency of the present method.

The first two conditions are obvious and have already been alluded to in Section 2: 1) the manifold learning algorithm must construct a proper embedding $f : \mathcal{M} \to \mathbb{R}^s$ of $\mathcal{M}$ and 2) the estimator f_n of f must be consistent. To our knowledge, very few algorithms carry explicit theoretical guarantees that these two conditions will be satisfied. In fact, the most significant work on the subject (Bernstein et al., 2000; Belkin and Niyogi, 2007) focuses on proving that $f_n \to f$, but omits the important question of whether f is indeed an embedding. It is not the aim of this paper to address this problem for all algorithms; we simply highlight the problem to underline the fact that obtaining the embedding h in manifold learning algorithms involves two important assumptions that are not always verified. Nevertheless, for illustrative purposes, we will consider the special case of Laplacian Eigenmaps and show that it can be constructed so as to satisfy condition 1, while condition 2 is already established (Giné and Koltchinskii, 2006; Belkin and Niyogi, 2007).

The third condition arises because the consistency of the embedding $f_n \to f$ is not sufficient to prove that the estimate of the embedding itself is consistent. Using the plug-in estimator for $\tilde{h}^{ij}(p) = 1/2\Delta_{\mathcal{M}}(f^i(x) - f^i(p))(f^j(x) - f^j(p))|_{x=p}$, as is done in MetricLearn , means that to establish consistency, it must also be true that $\forall i, j \in \{1, \ldots, s\}$

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

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 $\tilde{\mathcal{L}}_{n,\epsilon}$ is constructed and on the type of convergence. For example, if $\tilde{\mathcal{L}}_{n,\epsilon_n}$ is as defined in equation (17), 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).

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 (19).

The key to establishing (19) is to determine under what conditions

$$\left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f_n^i\right|\right|_\infty \overset{p}{\to} 0\,, \tag{20}$$

where the L_∞-norm is taken over the sampled points $p_l \in \mathcal{D}$ with, $l \in \{1, \ldots, n\}$. Indeed, from 20 it follows that $\left|\left|\Delta_{\mathcal{M}} f^i f^j - \tilde{\mathcal{L}}_{n,\epsilon_n} f_n^i f_n^j\right|\right|_\infty \overset{p}{\to} 0$ (see lemma 15) and hence equation 19.

Clearly, since $\tilde{h}_n$ is $s \times s$, i.e. finite dimensional, then element-wise convergence implies convergence of $\tilde{h}_n$. A minor point must be made here, however: convergence of $\tilde{h}_n$ does not guarantee convergence of its pseudoinverse h_n, since the pseudoinverse † is not a continous 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.

Although one can work directly with (20), as we will do below, we also that

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

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

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

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

We now turn we to the special case of the Laplacian Eigenmaps, where (20) 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.

6.1 Consistency of Laplacian Eigenmaps

As discussed in the previous section, three conditions are sufficient to obtain a consistent estimator of the embedding of g on the embedding $f(\mathcal{M})$:

1. f is an embedding of $\mathcal{M}$ into $\mathbb{R}^K$ and for each $i \in \{1, \ldots, K\}$, the coordinate $f^i \in C^3(\mathcal{M})$.
2. The empirical embedding is consistent, $\left|\left|f^i - f^i_n\right|\right|_\infty \overset{p}{\to} 0$
3. The empirical embedding satisfies $\left|\left|\tilde{\mathcal{L}}_{n,\epsilon_n} f^i - \tilde{\mathcal{L}}_{n,\epsilon_n} f^i_n\right|\right|_\infty \overset{p}{\to} 0 \ \forall i \in \{1, \ldots, m\}$.

The goal of this section is to show that these conditions can indeed be statisified and, by doing so, provide some hints as to how they can be shown to be satisfied in other algorithms. The special case considered here is that of the eigenmaps on a compact manifold with uniform sampling density and $\mathrm{Vol}(\mathcal{M}) = 1$. These conditions are needed to make use of the key results in (Giné and Koltchinskii, 2006; Belkin and Niyogi, 2007), which immediately establish condition 2. 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 1, i.e. that the eigenfunctions of the Laplace-Beltrami operator define an embedding and cannot be removed. To prove condition 1, 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 14 *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, $i = 1, \ldots, s$, 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 span $\big((\psi^i_{x^1}, \dots, \psi^i_{x^d}), i \in \{1, \dots, k\}\big) \subseteq$ span $\big((\phi_{l,x^1}, \dots, \phi_{l,x^d}), l \in \{0, 1, \dots\}\big)$ [3]. 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 span $\big((\phi_{l,x^1}, \dots, \phi_{l,x^d}), l \in L(p_j)\big) \cong \mathbb{R}^d$.

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 $h_n(p)$ 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 $h_n(p)$.

As mentioned before, condition 2 follows directly from (Giné and Koltchinskii, 2006; Belkin and Niyogi, 2007). We now need to prove condition 3:

$$\left|\left|\tilde{\mathcal{L}}_{n,\epsilon_n}\phi_i - \tilde{\mathcal{L}}_{n,\epsilon_n}\phi_{n,i}\right|\right|_{\infty} \xrightarrow{p} 0, \tag{21}$$

Although it is easy to establish 21 for the Eigenmaps, it is even simpler to work directly from 20 in this case.

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

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 $\mathcal{M}$.

7. Previous work

Besides the differential geometry background, for which we refer the reader to (Lee, 2003), this work draws upon two previous directions of research.

One is Laplacian estimation, and its consistency, which has already been discussed in detail in Sections 2.3 and 6.

The second and related direction is the theoretical work on analyzing the properties of manifold learning algorithms. Not surprisingly, most of the work has concentrated on the recovery of *flat manifolds*, i.e. of manifolds that are locally 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. (Donoho and Grimes, 2005) make a similar

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

argument w.r.t the continuous variant of the Isomap algorithm (called therein Continous Isomap). The domain of interest in this paper are Image Articulation Manifolds, however many of the results presented are more general. In particular, the paper develops a method to endow manifolds corresponding to simple articulations with a Euclidean metric. 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) who prove a more general theorem about the output of Continuous Isomap. This result allows to both prove isometric recovery (under given conditions) and to predict the distortions in the classical example of the swiss roll (rectangular density mapped non-linearly to higher dimensions).

Other methods focused on obtainig locally isometric maps. Among these, the already cited Hessian Eigenmaps method of (Donoho and Grimes, 2005) finds embeddings that are locally isometric to flat, connected neighborhoods . The method of (Brand, 2003) forms the embedding by aligning locally linear patches. The Local Tangent Space Alignment of (Zhang and Zha, 2004) also uses locally linear patches (estimates of tangent planes) which are merged according to a least squares cost. The method of (Weinberger and Saul, 2004) enforce local isometry in a Semidefinite Programming (SDP) framework.

Another SDP formulation for manifold embedding was introduced by (Sha and Saul, 2005); their Conformal Eigenmaps conformally maps triangles in each neighborhood, thus succeeding in preserving angles.

Departing with this work, (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.

Our work is outside this paradigm, but completes it. The Metric Learning method takes the non-isometric embedding produced by an algorithm such as the above, and augments it with the explicit Riemannian metric estimate in the coordinate system of the algorithm. It is also evident that, were an algorithm to output an atlas instead of a single chart, the Metric Learning method would apply seamlessly to this type of output.

Yet other work has brought to light the limitations of existing ML algorithms in recovering the geometry. A variety of counterexample are scattered in the literature; for a convenient synopsis we recommend the software (Wittman, 2005) and the companion paper.

A more theoretical approach comes from (Goldberg et al., 2008). They studied the family of algorithms including LLE, LEM, DFM, HEM, LTSA which compute the coordinates as solutions of a quadratic optimization subject to "normalization" constraints. They remarked that the algorithms generally do not produce an isometric mapping, and went on to define "successful recovery" as recovery up to an affine transformation. They further proved that for $d = 2$ successful recovery" is not possible if the manifold aspect ratio is too large. This definition of recovery (Goldberg et al., 2008) poses thus a global constraint on the coordinates recovered, equivalent to having a mapping f that is isometric up to an affine transformation. Thus, our results do not contradict the findings of this paper, but circumvent them. Note that f can be an embedding rather than a coordinate map, even though (Goldberg et al., 2008) assumes a coordinate map, and most of the results of (Goldberg et al., 2008) will still hold.

8. Experiments

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

8.1 g as a Measure of Local Distortion

Our first set of experiments is intended to illustrate the output of the MetricLearn algorithm. Figure 3 shows the embedding of a 2D hourglass-shaped manifold. Laplacian Eigenmaps, the embedding algorithm we used (with $s = 3$) distorts the shape by excessively flattening the top and bottom. MetricLearn outputs a $s \times s$ quadratic form for each point $p \in \mathcal{D}$, represented as ellipsoids centered at each p. Because $d = 2$, h has rank 2. Practically, this means that the ellipsoids are flat along one direction $T_{f(p)}f(\mathcal{M})^{\perp}$. 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) distortion introduced.

It is important to mention that this distortion is somewhat counterintuitive. The more the space is compressed, the larger the embedding metric 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.

We constructed the next example to demonstrate how our method applies to the popular Sculpture Faces dataset. This dataset was introduced by (Tenenbaum et al., 2000) along with Isomap as a prototypical example of how to recover a simple low dimensional manifold embedded in a high dimensional space. Specifically, the dataset consists of $n = 698$ 64×64 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 $[0, 1]^{4096}$.

As expected given its focus on preserving the geodesics distances, the Isomap seems to recover the simple rectangular geometry of the dataset. LTSA, however, does not perform so well and significant distortion is present in the corners, where the space is overstrectched (embedding is compressed). Meanwhile, the Laplacian Eigenmaps peforms the worst in terms of preserving the original geometry. It is interesting that it is the embedding for which we have theoretical guarantees of consistency that distorts the space the most. That being said, the theoretical guarantees say nothing about how distorted the embedding is likely to be.

It is important to stress that, *a priori*, it is not obvious which one of the embeddings best recovers the geometry, especially between Isomap and LTSA, since the fact that the parameter space[4] is a rectangular prism does not guarantee that the resulting manifold will have the exact same shape[5]. Therefore, the Isomap and LTSA could easily be considered equally valid representations of $\mathcal{M}$. In fact, when LTSA was introduced in (Zhang and Zha, 2004), the authors considered this particular dataset, and at no point did they suggest that LTSA's performance was inadequate in comparison to Isomap. It is truly only by computing h that we can appreciate the distortion induced by LTSA on this dataset.

Our next example, Figure 5, shows a mostly successful reconstruction of a common example, the swissroll with a square hole in the middle. This is a popular test dataset because many algorithms have trouble dealing with its unusual topology. In this case, the LTSA recovers the geometry of the manifold up to an affine transformation. This is evident from the deformation of the embedding metric, which is parallel for all points in Figure 5 (b).

One would hope that such an affine transformation of the correct geometry would be easy to correct; not surprisingly, it is. In fact, we can do more than correct it. For any embedding, there is a simple transformation that turns the embedding into a local isometry. Obviously, in the case of an affine transformation, locally isometric implies globally isometric. We describe these transformations

4. Face and light source orientation.
5. Specifically, one would expect $\mathcal{M}$ to be a diffeomorphism of the parameter space.

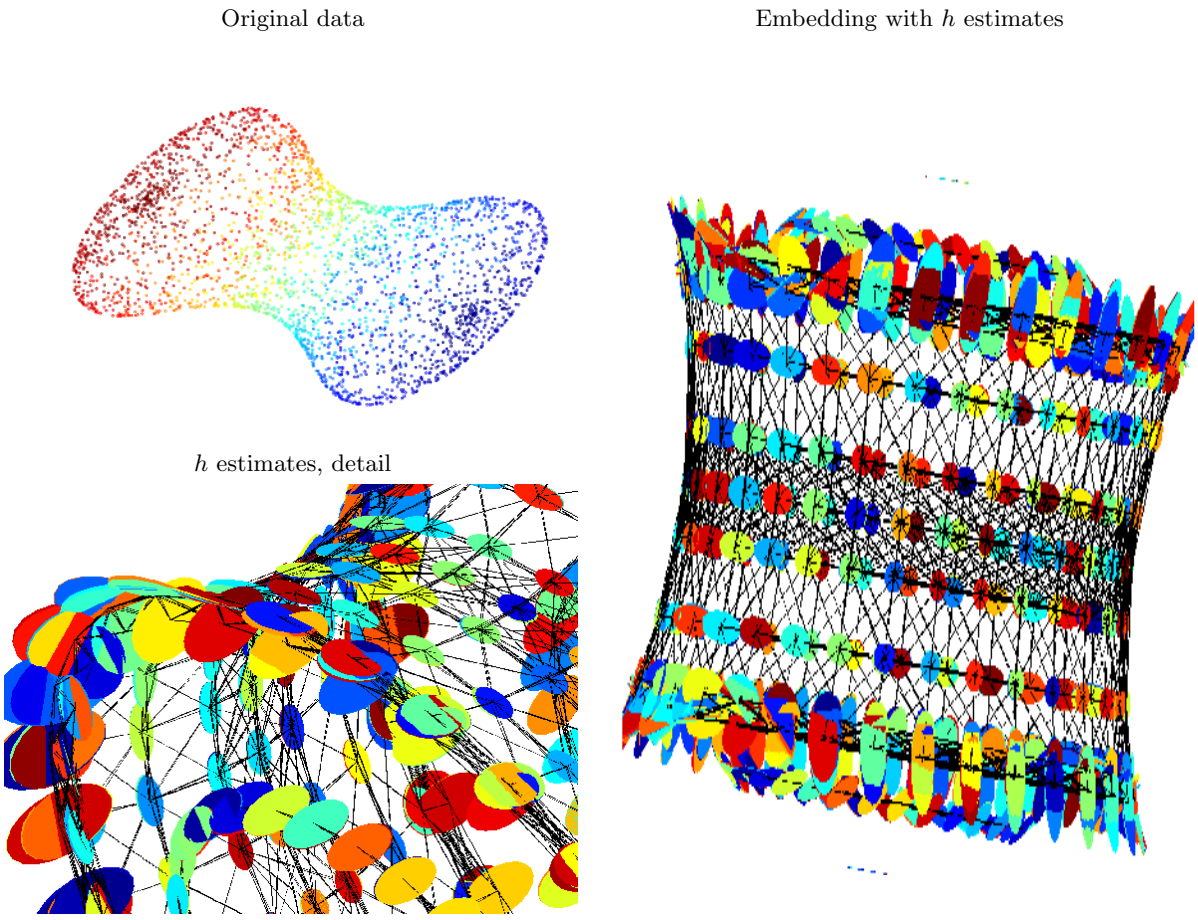


Figure 3: Estimation of h for a 2D hourglass-shaped manifold in 3D space. The embedding is obtained by Laplacian Eigenmaps. The ellipses attached to each point represent the embedding metric h estimate for this embedding. At each data point, h_p is a 3×3 symmetric semi-positive definite matrix of rank 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.

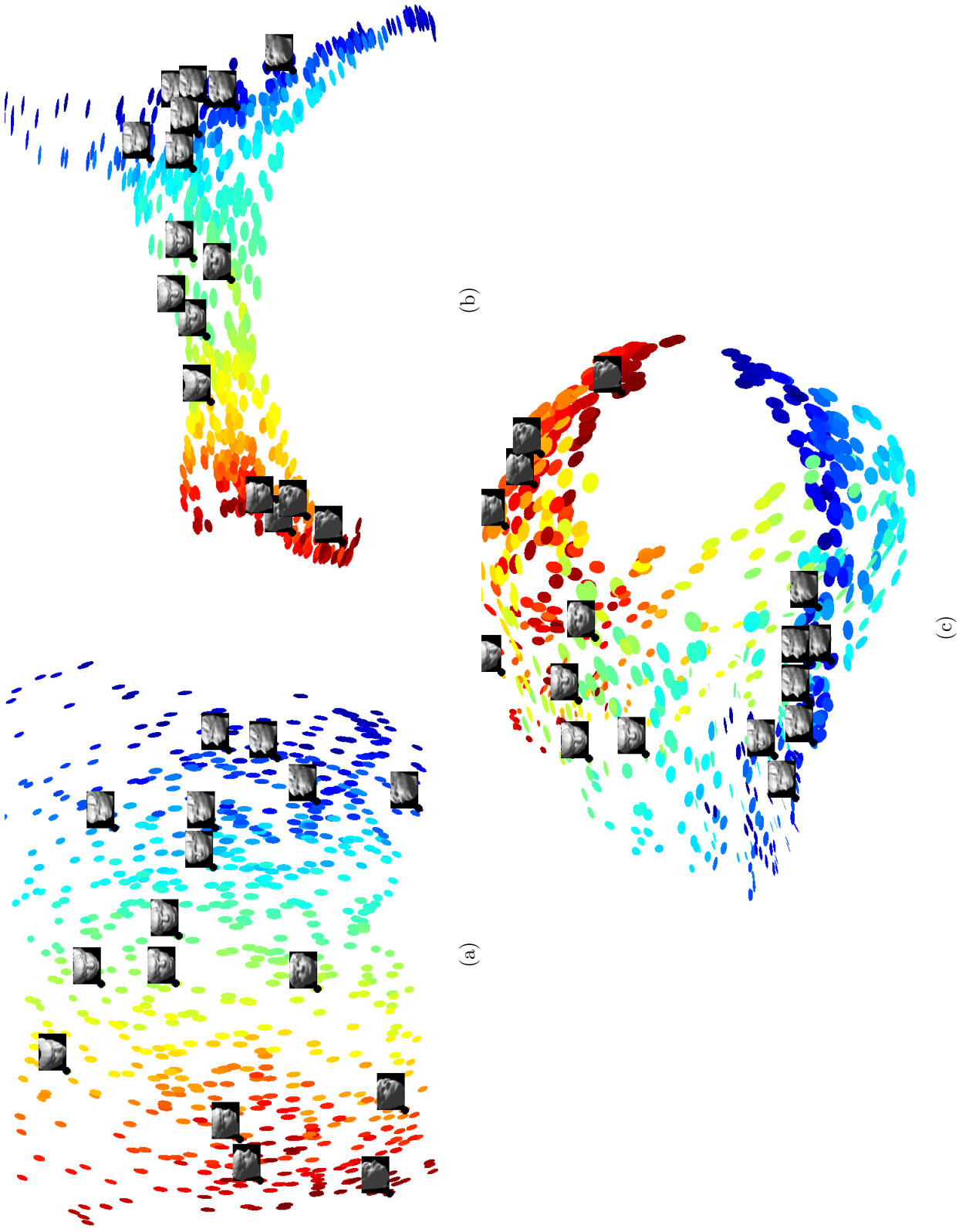
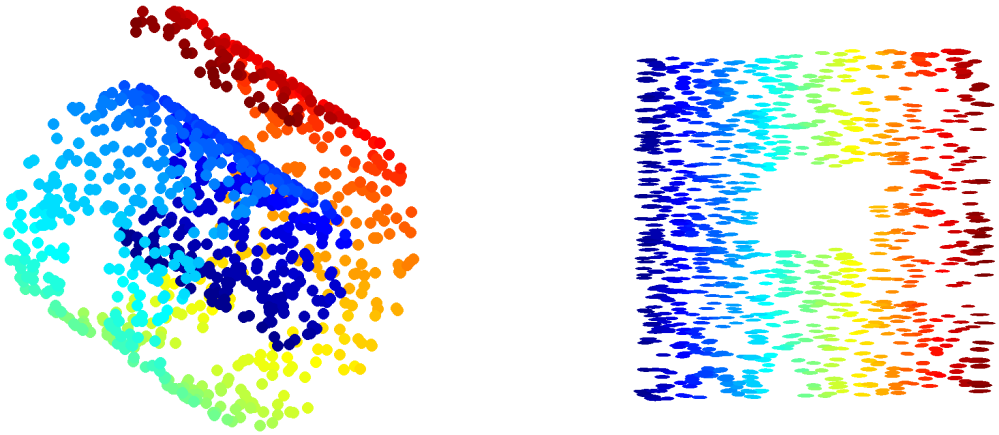


Figure 4: Two-dimensional visualization of the faces manifold, along with embedding. The color scheme corresponds to the left-right motion of the faces. The embeddings shown are: (a) Isomap, (b) LTSA, and (c) Laplacian Eigenmaps.



(a) (b)

Figure 5: (a) Swissroll with a hole in $\mathbb{R}^3$. (b) LTSA embedding of the manifold in $\mathbb{R}^2$ along with metric.

along with a few two-dimensional examples in the context of data visualization in the following section.

8.2 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 carthography: 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).

The MetricLearn algorithm 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.

Given $(f(\mathcal{D}), h(\mathcal{D}))$ Metric Embedding of $\mathcal{D}$

1. Select a point $p \in \mathcal{D}$ on the manifold
2. Transform coordinates $\tilde{f}(p') \leftarrow h_p^{-1/2} f(p')$ for $p' \in \mathcal{D}$

Display $\mathcal{D}$ in coordinates $\tilde{f}$

As mentioned above, the transformation $\tilde{f}$ ensures that the embedding of the new embedding $\tilde{f}$ is given by $\tilde{h}_p = I_s$, i.e. the unit matrix at p[6]. As h varies smoothly on the manifold, $\tilde{h}$ will also

6. Again, to be accurate, $\tilde{h}_p$ is the restriction of I_s to $T_{\tilde{f}(p)}\tilde{f}(\mathcal{M})$.

be close to I_s at points near p, and therefore the embedding will be approximately isometric in a neighborhood of p.

Figures 6, 7 and 8 exemplify this procedure for the Swiss roll with a rectangular hole of Figure 5.

8.3 Estimation 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 $\mathcal{M}$, which in our example is a half sphere of radius 1. The geodesic distance d being an intrinsic quantity, it should evidently not change with the parametrization.

We performed the following numerical experiment. First, we sampled $n = 1000$ points uniformly on a half sphere. Second, we selected two reference points p, p' on the half sphere so that their geodesic distance would be $\pi/2$. We then proceeded to run three manifold learning algorithms on $\mathcal{D}$, obtaining the Isomap, LTSA and LE embeddings. All the embeddings used the same 10-nearest neighborhood graph G.

For each embedding, and for the original data, we calculated the naive distance $||f(p) - f(p')||$. In the case of the original data, this represents the straight line that connects p and p' and crosses through the ambient space. For Isomap, $||f(p) - f(p')||$ should be the best estimator of $d_{\mathcal{M}}(p, p')$, since Isomap embeds the data by preserving geodesic distances using MDS. As for LTSA and LE, this estimator has no particular meaning, since these algorithms are derived from eigenvectors, which are defined up to a scale factor.

We also considered the *graph distance*, by which we mean the shortest path between the points in G, where the distance is given by $||f(q_i) - f(q'_{i-1})||$:

$$d_{\mathcal{G}}(f(p), f(p')) \;=\; \min\{\sum_{i=1}^{l} ||f(q_i) - f(q_{i-1})||,\, l \geq 1\,(q_0 = p,\, q_1,\, q_2, \ldots\, q_l = p')\,\text{path in}\,\mathcal{G}\}. \qquad (22)$$

Note that although we used the same graph $\mathcal{G}$ to generate all the embeddings, the shortest path between points may be different in each embedding since the distances between nodes will generally not be preserved.

The graph distance $d_{\mathcal{G}}$ is a good approximation for the geodesic distance d in the original data and in any isometric embedding, as it will closely follow the actual manifold rather then cross in the ambient space.

Finally, we computed the discrete minimizing geodesic as:

$$\hat{d}_{\mathcal{M}}(f(p), f(p')) \;=\; \min\{\sum_{i=1}^{l} \mathcal{H}(q_i, q_{i-1}),\, l \geq 1,\, (q_0 = p,\, q_1,\, q_2, \ldots\, q_l = p')\,\text{path in}\,\mathcal{G}\} \qquad (23)$$

where

$$\begin{aligned} \mathcal{H}(q_i, q_{i-1}) \;&=\; \frac{1}{2}\sqrt{(f(q_i) - f(q_{i-1}))^t h_{q_i}(f(q_i) - f(q_{i-1}))} \\ &\quad + \frac{1}{2}\sqrt{(f(q_i) - f(q_{i-1}))^t h_{q_{i-1}}(f(q_i) - f(q_{i-1}))} \end{aligned} \qquad (24)$$

is the discrete analog of the path-length formula (2) for the Voronoi tesselation of the space[7] Figure 9 shows the manifolds that we used in our experiments, and Table 1 displays the calculated distances.

As expected, for the original manifold, $||f(p) - f(p')||$ fares poorly. Crossing in the ambient space necessarily underestimates $d_{\mathcal{M}}$, while $d_{\mathcal{G}}$ is a very good approximation of $d_{\mathcal{M}}$, since it follows the

7. 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.

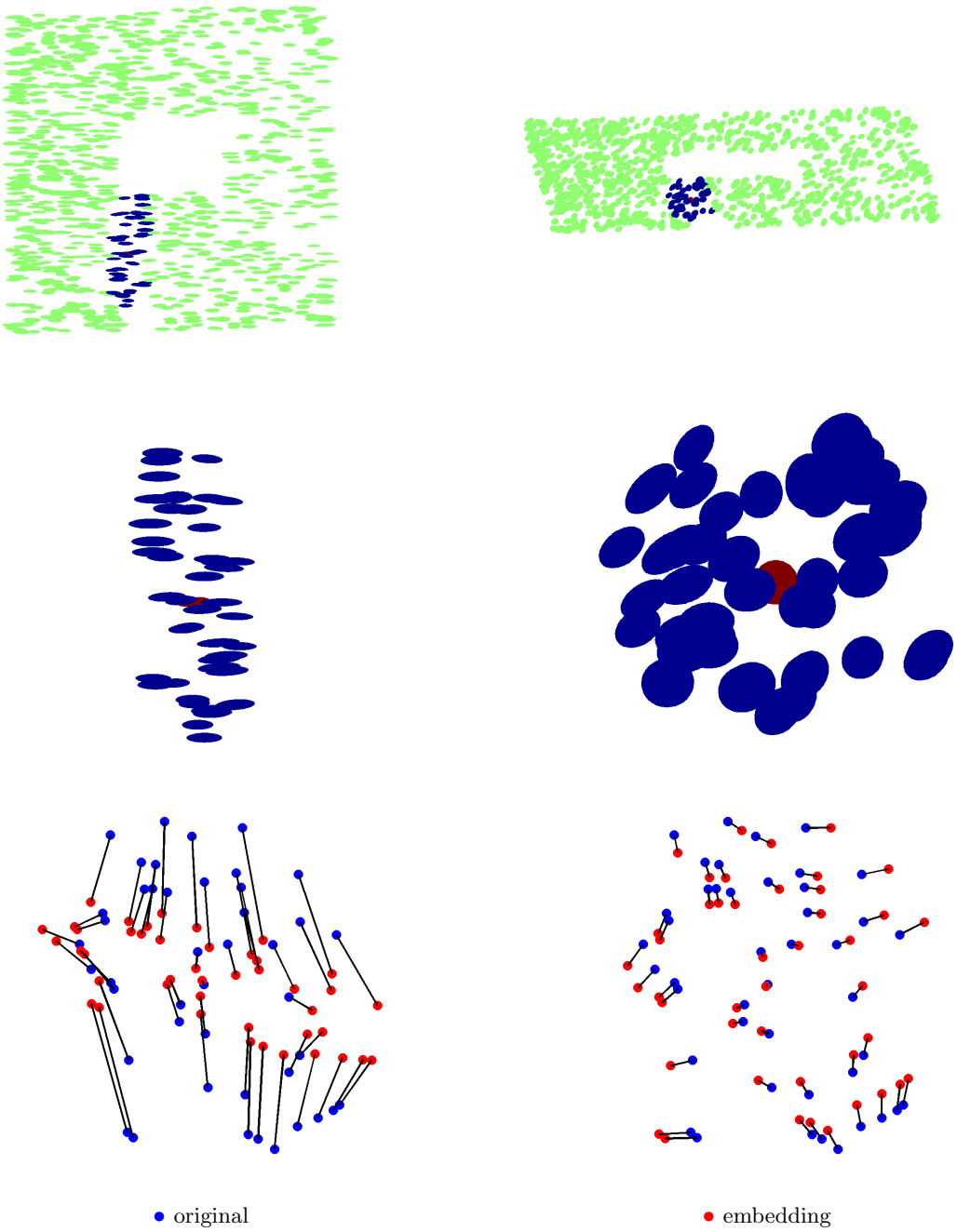


Figure 6: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by LTSA. Top, left: the LTSA embedding with selected point p (red), and its neighbors (blue). Top, right: The locally isometric embedding. Middle row, left: Neighborhood of p for the LTSA embedding. Middle row, right: Neighborhood of p for the locally isometric embedding. Bottow row, left: procrustes between the neighborhood of the p for the LTSA embedding and the original manifold projected on $T_p\mathcal{M}$. The procrustes dissimilarity measure is $D = 0.30$ (standardized sum of squares error). Bottom row, right: procrustes between the locally isometric embedding and the original manifold. The procrustes dissimilarity measure is $D = 0.02$.

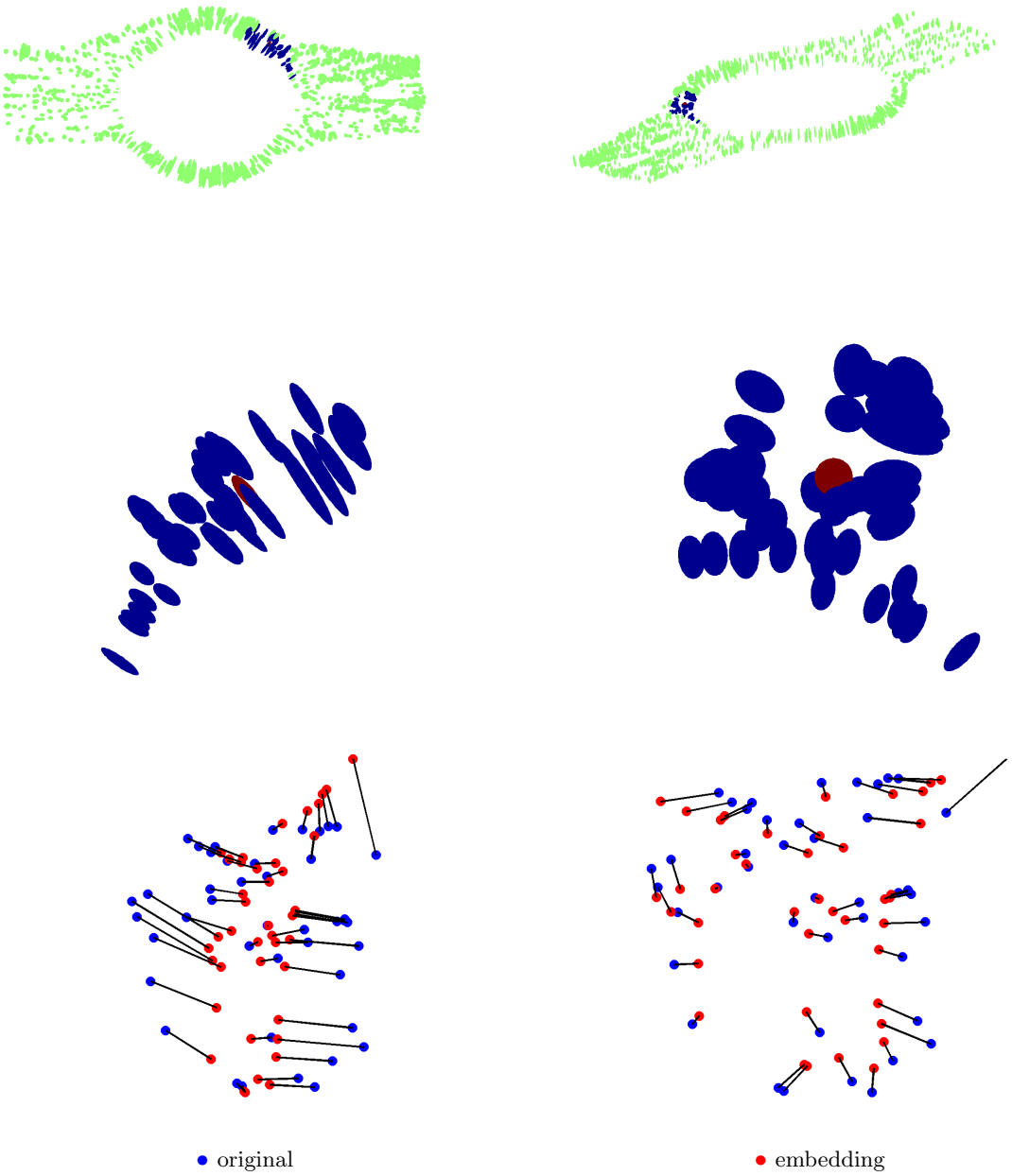


Figure 7: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by Isomap. Top, left: the Isomap embedding with selected point p (red), and its neighbors (blue). Top, right: The locally isometric embedding. Middle row, left: Neighborhood of p for the Isomap embedding. Middle row, right: Neighborhood of p for the locally isometric embedding. Bottow row, left: procrustes between the neighborhood of the p for the Isomap embedding and the original manifold projected on $T_p\mathcal{M}$. The procrustes dissimilarity measure is $D = 0.21$ (standardized sum of squares error). Bottom row, right: procrustes between the locally isometric embedding and the original manifold. The procrustes dissimilarity measure is $D = 0.06$.

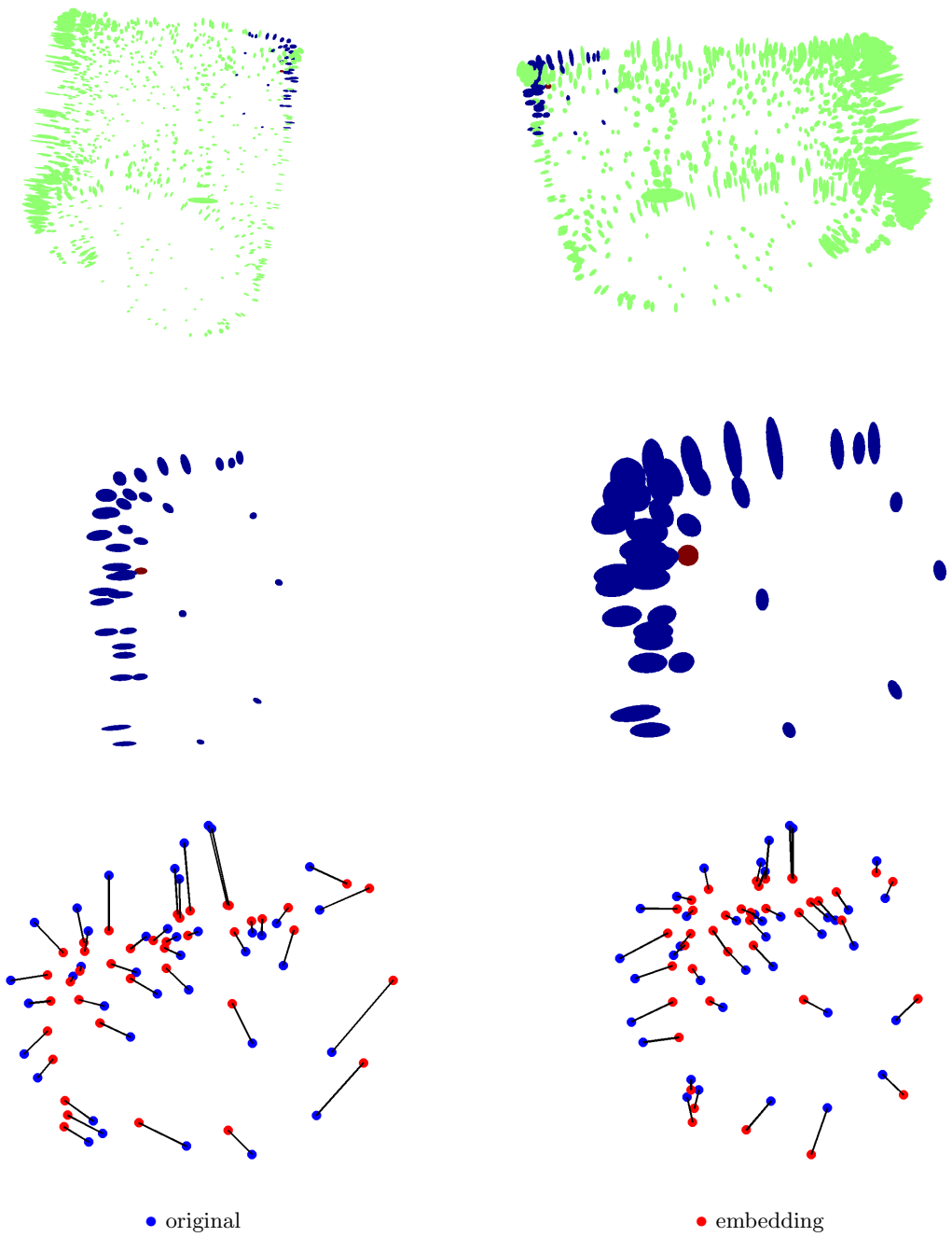
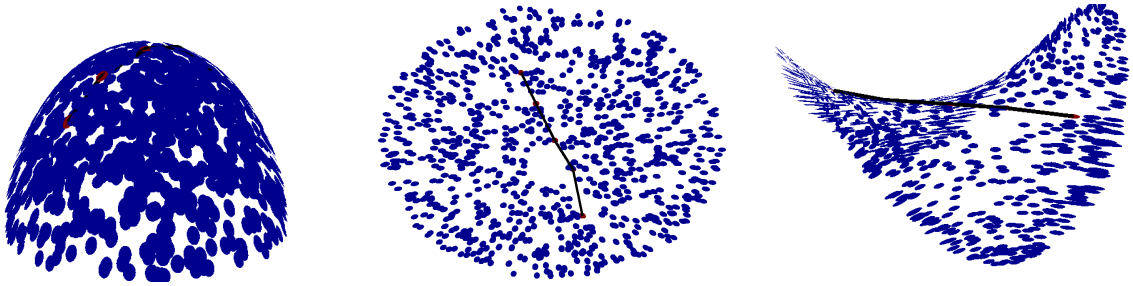


Figure 8: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by Laplacian Eigenmaps. Top, left: the Laplacian Eigenmaps embedding with selected point p (red), and its neighbors (blue). Top, right: The locally isometric embedding. Middle row, left: Neighborhood of p for the Laplacian Eigenmaps embedding. Middle row, right: Neighborhood of p for the locally isometric embedding. Bottow row, left: procrustes between the neighborhood of the p for the Laplacian Eigenmaps embedding and the original manifold projected on $T_p\mathcal{M}$. The procrustes dissimilarity measure is $D = 0.10$ (standardized sum of squares error). Bottom row, right: procrustes between the locally isometric embedding and the original manifold. The procrustes dissimilarity measure is $D = 0.07$.



(a) (b) (c)

Figure 9: Manifold and embeddings (black) 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) Laplacian Eigenmaps (c) Isomap.

Embedding	$\|\|f(p) - f(p')\|\|$	Shortest Path $d_{\mathcal{G}}$	Metric $\hat{d}$	$\hat{d}$ Relative Error
Original data	1.41	1.57	1.62	3.0%
Isomap $s = 2$	1.66	1.75	1.63	3.7%
LTSA $s = 2$	0.07	0.08	1.65	4.8%
LE $s = 3$	0.08	0.08	1.62	3.1%

Table 1: The errors in the last column are with respect to the true distance $d = \pi/2 \simeq 1.5708$.

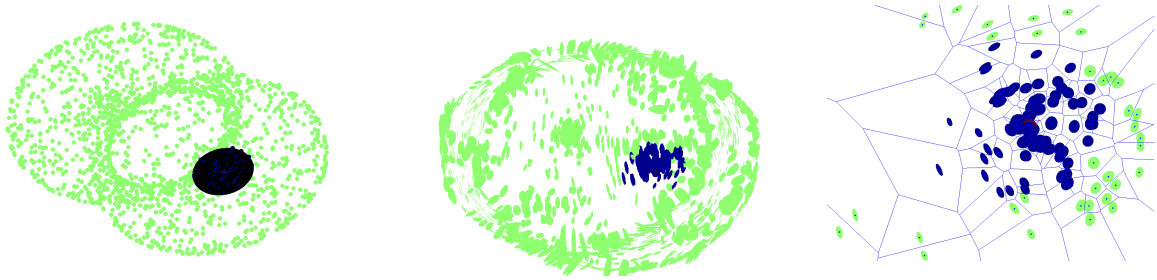
manifold more closely. Meanwhile, the opposite is true for Isomap. The naive distance $||f(p)-f(p')||$ is close to the geodesic by construction, while $d_{\mathcal{G}}$ overestimates $d_{\mathcal{M}}$ since $d_{\mathcal{G}} \geq ||f(p)-f(p')||$ by the triangle inequality. Not surprisingly, for LTSA and LE, both these estimates are wrong by nearly two orders of magnitude. This simply stresses the fact that the estimates $||f(p) - f(p')||$ and $d_{\mathcal{G}}$ have no absolute meaning insofar as those algorithms are concerned.

However, the 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.

8.4 Volume Estimation

The last set of our experiments demonstrate the use of the Riemannian metric in estimating 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).

One extra step is required when computing areas that was optional when computing distances: we need to construct coordinate chart(s) to represent the area of interest. Indeed, to make sense of the Euclidean volume element $dx^1 \dots dx^d$, we need to work in $\mathbb{R}^d$. Specifically, we resort to the idea expressed at the end of Section 4.2, which is to project the embedding on its tangent at the point p around which we wish to compute $dx^1 \dots dx^d$. This tangent plane $T_{f(p)}f(\mathcal{M})$ is easily identified from the SVD of h_p and its singular vectors with non-zero singular values. It is then straightforward to



(a) (b) (c)

Figure 10: (a) Manifold along with W, the computed area. (b) Laplacian Eigenmaps embedding with embedding metric h. (c) A locally isometric coordinate chart constructed from the Laplacian Eigenmaps along with the Voronoi tessellation. For Figures (b) and (c), the point at the center of W is in red, the other points in W are in blue and the points not in W are in green.The sample size is $n = 1000$.

Embedding	Naive Area of W	$\hat{\text{Vol}}(W)$	$\hat{\text{Vol}}(W)$ Relative Error
Original data	0.74†	0.86	2.5%
Isomap	1.74†	0.83	1.3%
LTSA	8.56e-04	0.86	2.8%
LE	6.37e-04†	0.83	0.95%

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 $\simeq 0.8409$

use the projection Π of an open neighborhood $f(U)$ of $f(p)$ onto $T_{f(p)}f(\mathcal{M})$ to define the coordinate chart $(U, x = \Pi \circ f)$ around p. Since this is a new chart, we need to recompute the embedding $\tilde{h}$ for it.

By performing a tessellation of $(U, x = \Pi \circ f)$ (we use the Voronoi tesselation for simplicity), we are now in position to compute $\triangle x^1 \dots \triangle x^d$ around p and multiply it by $\sqrt{\det(\tilde{h})}$ to obtain $\triangle\text{Vol} \simeq d\text{Vol}$. Summing over all points of the desired set gives the appropriate estimator:

$$\hat{\text{Vol}}(W) = \sum_{p \in W} \sqrt{\det(\tilde{h_p})} \triangle x^1(p) \dots \triangle x^d(p)\,. \tag{25}$$

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, $\hat{\text{Vol}}(W)$ seem 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 only through the embedding metric can this distortion be accounted for.

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 augment any embedding with the appropriate metric that ensures its faithfulness to the original geometry.

Our method essentially frees users to select their preferred embedding algorithm based on considerations unrelated to the geometric recovery; the metric learning algorithm then obtains the Riemannian metric corresponding to these coordinates. Once these are obtained, the distances, angles, and other geometric quantities can be estimated in the embedded manifold by standard manifold calculus. These quantities will preserve their values from the original data[8] and are thus embedding-invariant in the limit of $n \to \infty$.

In essence, 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 information stored under the Metric Learning algorithm is of order s^2 per point, while the coordinates only require s values per point.

Metric Learning also has other implications for the field of manifold learning. As our experiments with geodesic distances show, one can now obtain comparable distance estimates from methods designed to preserve distances, like Isomap, as well as from other methods, like the Laplacian Eigenmap. Hence, one could use the faster LE 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.

Each manifold learning algorithm produces its own coordinate system. Hence, it is not easy to compare (or align) outputs of different algorithms 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.

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.

Extensions In the next few paragraphs, we discuss the possibility of extending Metric Learning to multiple dimension choices, multiple charts, and noisy data.

In practice, the embedding dimension s is often not given, and is larger than d. In such situations, one tests a range of values of s, and proceeds to choose an optimal one. This is typical of renormalized Laplacian embedding algorithms like LE and DM, which do not assume that $s = d$. For these algorithms, the embedding coordinates correspond to eigenvectors of a given matrix, so 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 $h^\dagger$ pseudoinverse metric can also be obtained incrementally, by simply applying (12) to the new coordinate

8. This is true in the absence of noise. See the following paragraphs for a discussion of noisy manifold data.

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).

Another potential extension relates to noise in the data. Indeed, 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 among 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.

Until now, we have implicitly worked under the assumption that original geometry of data is important for the analysis. However, not everyone agrees that the original geometry is interesting in of itself and that it should be discarded in favor of a new geometry that better highlights the important features of the data. For example, clustering algorithms stress the importance increasing the dissimilarity (distance) between different classes regardless of what the original geometry dictates. This is in fact one of the arguments advanced by (Nadler et al., 2006) in support of spectral clustering which pulls points towards regions of high density.

Even in situations where the new geometry is considered more important, however, understanding the relationship between the original and the new geometry using Metric Learning - and, in particular, the pullback metric - could be of value and offer further insight. Indeed, while we explained in Section 8 how the embedding metric h can be used to infer how the original geometry was affected by the map f, we note at this juncture that the pullback metric, i.e. the geometry of $(f(\mathcal{M}), \delta_s)$ pulled back to $\mathcal{M}$ by the map f, can offer interesting insight into the effect of the transformation/embedding.[9] In fact, this idea has already been considered by (Burges, 1999) in the case of kernel methods where one can compute the pullback metric directly from the definition of the kernel used. In the framework of Metric Learning, this can be extended to any transformation of the data that defines an embedding.

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. This research was partly supported by NSF awards IIS-0313339 and IIS-0535100.

References

Raich R. Behmardi, B. 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.

9. One caveat to this idea is that, in the case where $r >> 1$, computing the pullback will not be practical and the pushforward will remain the best approach to study the effect of the map f. It is for the case where r is not too large and $r \sim s$ that the pullback may be a useful tool.

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.

Matthew Brand. Charting a manifold. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 961–968, 2003.

C. J. C. Burges. Geometry and invariance in kernel based methods. *Advances in Kernel Methods - Support Vector Learning*, 1999.

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

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.

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

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

B. Nadler, S. Lafon, and R. R. Coifman. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Neural Information Processing Systems Conference*, 2006.

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.

L. Saul and S. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifold. *Journal of Machine Learning Research*, 4:119–155, 2003.

F. Sha and L.K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *International Conference of Machine Learning (ICML)*, pages 785–792, 2005.

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.

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

K. Weinberger and L. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE International Conference Computer Vision and Pattern Recognition*, volume 2, pages 988–995. IEEE, 2004.

T. Wittman. MANIfold learning matla 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.

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM J. Scientific Computing*, 26(1):313–338, 2004.

10. Appendix

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. ■