Learning and Manifolds: Leveraging the Intrinsic Geometry

Dominique C. Perrault-Joncas

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2012

Reading Committee:

Marina Meilă, Chair

Jon A. Wellner

John M. Lee

Vladimir Minin

Program Authorized to Offer Degree:
Department of Statistics

University of Washington

Abstract

Learning and Manifolds: Leveraging the Intrinsic Geometry

Dominique C. Perrault-Joncas

Chair of the Supervisory Committee:
Associate Professor Marina Meilă
Department of Statistics

In this work, we explore and exploit the use of differential operators on manifolds - the Laplace-Beltrami operator in particular - in learning tasks. In particular, we are interested in uncovering the geometric structure of data (unsupervised learning) and in exploiting information contained in unlabelled data for regression and classification tasks (semi-supervised learning).

First, building on the Laplacian Eigenmap and Diffusionmaps 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. We provide an algorithm for estimating the Riemannian metric from data, consider its consistency, and demonstrate possible applications of our approach in a variety of examples.

Second, we extend the idea of learning the geometry of the data to improve the performance of prediction tasks. From a statistical point of view, this means dealing with data that are locally collinear, but where the global relationship between the covariates is non-linear. We do this by combining the Matérn Gaussian process - a flexible and easily interpretable Bayesian non-parametric regression model - with the Laplace-Beltrami operator, which embodies all the intrinsic geometry of the mani-

fold. This yields a principled geometrical approach for learning tasks on the intrinsic geometry of the manifold.

Finally, we turn to the problem of setting the hyperparameters used to construct the graph Laplacian, the highly prevalent non-parametric estimator of the Laplace-Beltrami operator. Specifically, we study the problem of setting ϵ, the kernel bandwidth, to construct the graph Laplacian for Euclidean data - a parameter that, according to our results, has a material impact in both the unsupervised and semi-supervised learning context. We exploit the connection between manifold geometry and the Laplace-Beltrami operator so as to obtain the hyperparameters for which the graph Laplacian best captures the geometry of the data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to his advisor, Professor Marina Marina Meilă, for her support and guidance, Professors Lee, Wellner, and Minin for their insight and comments, and his family for their encouragement and faith.

DEDICATION

to my wonderful wife, Nevena

Chapter 1

INTRODUCTION

In this work, we explore and exploit the use of differential operators on manifolds - the Laplace-Beltrami operator in particular - for uncovering the structure of data (unsupervised learning) and for exploiting information contained in unlabelled data for regression and classification tasks (semi-supervised learning).

The need for uncovering the structure of data often arises in cases where the data is very high-dimensional. Such data is typically processed with feature selection, such as Lasso-based regularization. However, in the presence of local collinearity, this approach tends to fail (Aswani et al. [2011]). Since collinearity is a geometric concept that is equivalent to having predictors on manifolds, manifold learning is much better suited to dealing with such data.

In that framework, the data is assumed to lie on or close to a low-dimensional Riemannian manifold embedded in the high-dimensional space. The task then becomes to recover the low-dimensional manifold so as to perform further statistical analysis with respect to the low-dimensional geometry of the data.

Manifold learning has been applied extensively to this problem, yielding numerous algorithms of varying degrees of complexity for embedding the data. These algorithms share two important shortcomings: 1) they fail to recover the intrinsic geometry of the manifold in many instances (i.e. they do not preserve important elements such as the distance between points) and 2) no coherent framework yet exists in which they can easily be compared and selected for a given application (Goldberg et al. [2008], Wittman [2005, retreived 2010]).

Our work on the Laplace-Beltrami operator helps us address these shortcomings. Indeed, in Chapter 2, our key contribution is to establish a correspondence between the Laplace-Beltrami operator and the geometry of the embedding. This allows us to use the operator on any embedding of the data to estimate the local distortion of the original Riemannian metric. This local distortion, in turn, allows us to to evaluate either how the embedding modifies the geometry of the original manifold or how it differs from other embeddings in its effect on that geometry.

In Chapter 3, we exploit the fact that stochastic partial differential equations based on the Laplace-Beltrami operator can be used to define the popular Matérn Gaussian field on the manifold to introduce geometrically motivated non-parametric Bayesian regression or, equivalently, geometrically motivated kernel regularization. In so doing, we combine some of the techniques that have been developed independently in the arenas of semi-supervised and supervised learning.

In supervised learning, the Matérn covariance function and its properties are well understood, and its parameters can be ascribed clear meanings and uses (length scale and smoothness), leading to its popularity for spatial statistics and Gaussian Process Regression. Recent work by Lindgren et al. [2011] highlights that the Matérn Gaussian field is equivalent to a specific stochastic partial differential equation, which allows for defining a Matérn Gaussian field on a known manifold.

We show that work already done in semi-supervised learning - that is to say, kernel regularization with respect to the Laplace-Beltrami operator - can be understood as a special case of the Matérn stochastic partial differential equation. The natural approach, then, is to extend the Matérn Gaussian process regression to the semi-supervised setting. Indeed, because the graph Laplacian provides a non-parametric estimator of the Laplace-Beltrami operator, we possess all the necessary tools to extend the stochastic partial differential equation definition of the Matérn Gaussian field of [Lindgren et al., 2011] to input spaces of unknown geometry. Our contribution in Chapter 3 is thus to bridge the gap between knowledge in supervised and semi-supervised learning and introduce the Matérn Gaussian field to semi-supervised learning, showing that it allows for a much more elegant and useful interpretation of the parameters involved.

Finally, in Chapter 4, we exploit the equivalence between the Laplace-Beltrami operator and the geometry of the manifold to find the optimal parameter for the bandwidth of the graph Laplacian - one of the hyperparameters needed to ensure the graph Laplacian is a good approximation of the Laplace-Beltrami operator that are

not easy to learn by cross-validation or other model selection approaches. Improving on the current technique of simply "tuning" the bandwidth of the graph Laplacian, we present a new, principled approach to selecting the scale parameter that is based on the faithfulness to the geometry of the original data. Our approach consists of choosing the bandwidth parameter that yields the graph Laplacian that least distorts the Riemannian metric of the original data. The experiments we conduct show that our principled estimate of the bandwidth leads to improvements in the performance of regression and classification methods and reduces their computational complexity.

Chapter 2

LEARNING RIEMANNIAN GEOMETRY

In this chapter, we consider the unsupervised problem of learning manifolds on unlabelled data. **Unsupervised learning** involves a set of n points, $\mathcal{P} = \{x_1, ..., x_n\}$, often assumed to be drawn i.i.d. from a distribution $\pi(x)$ on some sample space $\mathcal{X}$. The objective of unsupervised learning is to find interesting *structure* in the data - often some features of $\pi(x)$, such as its support in $\mathcal{X}$, clusters, quantiles, or outliers.

One form of unsupervised learning is dimensionality reduction. A common technique for performing this task with large sets of high-dimensional data is Principal Component Analysis, which assumes that the data lies on a low-dimensional affine space. Since the affine space requirement is generally violated in practice, more general techniques for non-linear dimensionality reduction have been developed. Most of these approaches rely on the **manifold hypothesis**, which assumes that the support of $\pi(x)$ lies on or close to a manifold $\mathcal{M} \subset \mathcal{X}$. 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, retreived 2010]). 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, using the concepts of

Riemannian metric and *isometry*. The criticisms noted above reflect the fact that the majority of manifold learning algorithms output embeddings that are not isometric to the original data except in special cases.

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

The information necessary for reconstructing the geometry of the manifold is embodied in its Riemannian metric, defined in Section 2.3. We propose to use the the Laplace-Beltrami operator to recover a *Riemannian manifold* $(\mathcal{M}, g)$ from the data, that is, a manifold and its Riemannian metric g, and express g in any desired coordinate system. Practically, for any given mapping produced by an existing manifold learning algorithm, we will add an estimation of the Riemannian metric g in the new data coordinates, that makes the geometrical quantities like distances and angles of the mapped data (approximately) equal to their original values, in the raw data.The core of our contribution in this chapter is the demonstration of how to obtain the Riemannian metric from the mathematical, algorithmic and statistical points of view.

2.1 The Task of Manifold Learning

In this section, we present the problem of manifold learning. We focus on formulating 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 2.2 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 2.1.4 below, and then a more detailed framework will be presented in Section 2.5.

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

We begin our survey of manifold learning algorithms by discussing a well-known method for linear dimensionality reduction: Principal Component 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 cannot recover any curvature present in the data.

In contrast to linear techniques, manifold learning algorithms assume that the data lies near or along a non-linear, smooth, submanifold 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 $\mathcal{P} = \{p_1, \dots p_n\} \subset \mathbb{R}^r$, 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.

2.1.1 Nearest Neighbors Graph

Existing manifold learning algorithms pre-process the data by first constructing a *neighborhood graph* $\mathcal{G} = (V, E)$, where V are the vertices and E the edges of $\mathcal{G}$. While the vertices are generally taken to be the observed points $V = \{p_1, \dots, p_n\}$, there are three common approaches for constructing the edges.

The first approach is to construct the edges by connecting the k nearest neighbors for each vertex. Specifically, $(p_i, p_j) \in E_k$ if p_i is one of the k-nearest neighbors of p_j or if p_j is one of the k-nearest neighbors of p_i. $\mathcal{G}_k = (V, E_k)$ is then known as the $k-$nearest neighbors graph. While it may be relatively easy to choose the neighborhood parameter k with this method, it is not very intuitive in a geometric sense.

The second approach is to construct the edges by finding all the neighborhoods of radius $\sqrt{\epsilon}$ so that $E_\epsilon = \{\mathbf{1}[||p_i - p_j||^2 \leq \epsilon] : i, j = 1, \dots, n\}$. This is known as the ϵ-neighborhood graph $\mathcal{G}_\epsilon = (V, E_\epsilon)$. The advantage of this method is that it is geometrically motivated; however, it can be difficult to choose $\sqrt{\epsilon}$, the bandwidth parameter. Choosing a $\sqrt{\epsilon}$ that is too small may lead to disconnected components, while choosing a $\sqrt{\epsilon}$ that is too large fails to provide locality information - indeed, in the extreme limit, we obtain a complete graph. Unfortunately, this does not mean that the range of values between these two extremes necessarily constitutes an appropriate middle ground for any given learning task.

The third approach is to construct a complete weighted graph where the weights

represent the *closeness* or *similarity* between points. A popular approach for constructing the weights, and the one we will be using here, relies on kernels Ting et al. [2010]. For example, weights defined by the heat kernel are given by

$$w_\epsilon(i,j) = \exp\left(\frac{-||p_i - p_j||^2}{\epsilon}\right) \tag{2.1}$$

such that $E_{w_\epsilon} = \{w_\epsilon(i,j) : i,j = 1,\ldots,n\}$. The weighted neighborhood graph G_{w_ϵ} has the same advantage as the ϵ-neighborhood graph in that it is geometrically motivated; however, it can be difficult to work with given that any computations have to be performed on a complete graph. This computational complexity can partially be alleviated by truncating for very small values of w (or, equivalently, for a large multiple of $\sqrt{\epsilon}$), but not without reinstating the risk of generating disconnected components. However, using a truncated weighted neighborhood graph compares favorably with using an ϵ'-neighborhood graph with large values of ϵ' since the truncated weighted neighborhood graph G_{w_ϵ} - with $\epsilon < \epsilon'$ - preserves locality information through the assigned weights.

Though we merely allude to the importance of selecting the bandwidth $\sqrt{\epsilon}$ here, we will discuss it at length in Chapter 4.

In closing, we note that some authors distinguish between the step of creating the nearest neighbors graph using any one of the methods we discussed above, and the step of creating the similarity graph (Belkin and Niyogi [2002]). In practical terms, this means that one can improve on the k nearest neighbors graph by applying the heat kernel on the existing edges, generating a weighted k nearest neighbors graph.

2.1.2 Existing Algorithms

Without attempting to give a thorough overview of the existing manifold learning algorithms, we discuss two main categories. 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 s 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 2.4 below. The graph Laplacian is used because its eigendecomposition can be shown to preserve local distances while maximizing the smoothness of the embedding. Thus, the LE embedding is obtained simply by keeping the first s eigenvectors of the graph Laplacian in order of ascending eigenvalues. The first eigenvector is omitted, since it is necessarily constant and hence non-informative.

- ***DM: The Diffusion Map*** is a variation of the LE that emphasizes the deep

connection between the graph Laplacian and heat diffusion on manifolds. The central idea remains to embed the data using an eigendecomposition of the graph Laplacian. However, DM defines an entire family of graph Laplacians, all of which correspond to different diffusion processes on $\mathcal{M}$ in the continuous limit. Thus, the DM can be used to construct a graph Laplacian whose asymptotic limit is the Laplace-Beltrami operator, defined in (2.3), 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.

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

expressed as a semi-definite program.

- ***ISOMAP***: This is an example of a non-linear global algorithm. The idea is to embed $\mathcal{M}$ in $\mathbb{R}^s$ using the minimizing geodesics 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 approximate the minimizing geodesics between each pair of points. The final step consists in embedding the data using Multidimensional Scaling (MDS) on the computed geodesics 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 minimizing geodesics.

- ***MDS***: For the sake of completeness, and to aid in understanding the Isomap, we also provide a short description of MDS. MDS is a spectral 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.

2.1.3 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 2.2 and 2.3, as well as in later chapters.

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 with Boundary) *A d-dimensional* ***manifold*** $\mathcal{M}$ ***with boundary*** *is a topological (Hausdorff) space such that every point has a neighborhood homeomorphic to an open subset of* $\mathbb{H}^d \equiv \{(x^1, ..., x^d) \in \mathbb{R}^d | x^1 \geq 0\}$. *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* $V \subset \mathbb{H}^d$. *A* C^∞-***Atlas*** $\mathcal{A}$ *is a collection of charts,*

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

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

$$x_\beta \circ x_\alpha^{-1} : x_\alpha(U_\alpha \cap U_\beta) \to \mathbb{R}^d, \tag{2.2}$$

is continuously differentiable any number of times. Finally, a ***smooth manifold*** $\mathcal{M}$ ***with boundary*** *is a manifold with boundary with a* C^∞*-Atlas.*

Note that to define a manifold without boundary, it suffices to replace $\mathbb{H}^d$ with $\mathbb{R}^d$ in Definition 1 . For simplicity, we assume throughout that the manifold is smooth, but it is commonly sufficient to have a $\mathcal{C}^2$ manifold, i.e. a manifold along with a $\mathcal{C}^2$ atlas. 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 reformulate the goal of manifold learning: assuming that our (high-dimensional) data set $\mathcal{P} = \{p_1, \dots p_n\} \subset \mathbb{R}^r$ comes from a smooth manifold with low d, the goal of manifold learning is to find a corresponding collection of d-dimensional coordinate charts for these data.

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

algorithm on the same data, one must resort to *intrinsic*, coordinate-independent quantities. As we shall see later in this chapter, the framework we propose takes this observation into account.

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

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

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

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

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

The Strong Whitney Embedding Theorem (Lee [2003]) 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 $s \leq 2d$[1] with a single map $f : \mathcal{M} \to \mathbb{R}^s$.

[1] In practice, it may be more practical to consider $s \leq 2d + 1$, since any smooth map $f : \mathcal{M} \to \mathbb{R}^{2d+1}$ can be perturbed to be an embedding. See Whitney Embedding Theorem in Lee [2003] for details.

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 data sets. 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 our work, 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 embedding. That being said, the choice of s will only be discussed tangentially in this thesis and even then, the constraint $s \leq 2d$ will not be enforced.

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

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

Proving consistency of various manifold-derived quantities has received considerable attention in the literature ([Bernstein et al., 2000], [von Luxburg et al., 2008]).

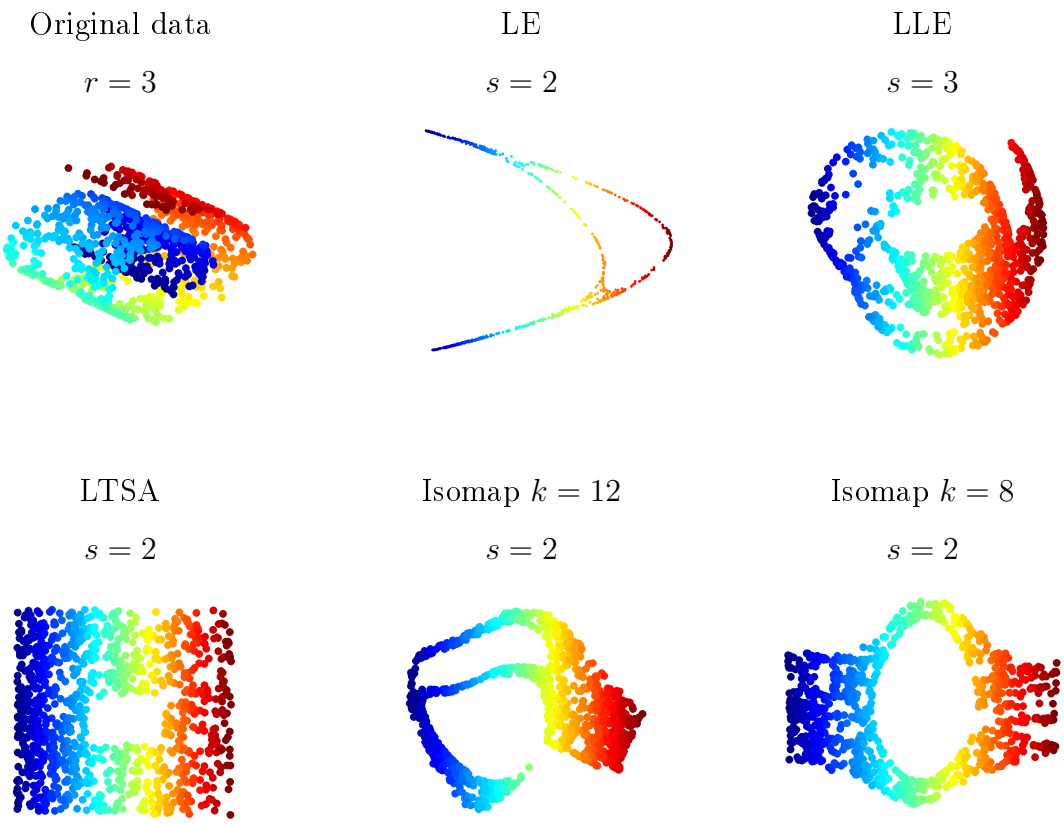


Figure 2.1: Manifold learning algorithms distort the geometry of the data. The classical "Swiss roll" example is shown here embedded via a variety of manifold learning algorithms. For clarity, the original data is in $r = 3$ dimensions; it is obvious that adding extra dimensions does not affect the resulting embeddings.

However, the meaning of consistency in the context of manifold learning remains unclear. 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 geodesic distance on the manifold $\mathcal{M}$ (Bernstein et al. [2000]). Unfortunately, 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.

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 2.3) for a uniform sampling density on $\mathcal{M}$. These results also allow us to assume, by extension, the consistency of the class of algorithms that use the eigenvectors of the Laplace-Beltrami operator to construct embeddings - Laplacian Eigenmaps and Diffusion Maps.

Though incomplete in some respects, these results allow us to 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 2.5.

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

2.1.5 Manifold Geometry Preservation

Having a consistent smooth mapping from $f : \mathcal{M} \to \mathbb{R}^s$ 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 mini-

mum, even to zero. In other words, can we preserve not only neighborhood relations, but also distances within the manifold? Or, going one step further, could we find a way to simultaneously preserve distances, areas, volumes, angles, etc. - in a word, the *intrinsic geometry* - of the manifold?

Manifold learning algorithms generally fail at preserving the geometry, even in simple cases. We illustrate this with the well-known example of the "Swiss-roll with a hole" (Figure 2.1), a two dimensional strip with a rectangular hole, rolled up in three dimensions, sampled uniformly. Of course, no matter how the original sheet is rolled without stretching, lengths of curves within the sheet will be preserved. So will areas, angles between curves, and other geometric quantities. However, when this data set is embedded using various algorithms, this does not occur. 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 - 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, retreived 2010].

The problem of geometric distortion is central to this thesis: its 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 in this Chapter 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 observed in Figure 2.1 will persist.

2.2 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 the Chapter.

2.2.1 The Riemannian Metric

The extension of Euclidean geometry to a manifold $\mathcal{M}$ is defined mathematically via the Riemannian metric.

Definition 4 (Riemannian Metric) *A 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[2]) 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 $\cos(\theta) = \frac{<u,v>_g}{||u||||v||}$. Thus, in any coordinate representation of $\mathcal{M}$, g at point p is represented as a $d \times d$ symmetric positive definite matrix.

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

$$l(c) \;=\; \int_a^b \sqrt{g_{ij}\frac{dx^i}{dt}\frac{dx^j}{dt}}dt, \tag{2.3}$$

[2]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.

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

$$\mathrm{Vol}(W) = \int_W \sqrt{\det(g)} dx^1 \dots dx^d \,. \tag{2.4}$$

Obviously, these definitions are trivially extended to overlapping charts by means of the transition map (2.2). For a comprehensive treatement of calculus on manifolds, the reader is invited to consult [Lee, 1997].

2.2.2 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) *Let $f : \mathcal{M} \to \mathcal{N}$ be a diffeomorphism 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, areas and volumes. It is then natural to take isometry as the strictest notion of what it means for an algorithm to "preserve geometry".

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

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

$$\langle u, v \rangle_{\varphi^* g_p} \;=\; \left\langle df_p^{-1} u, df_p^{-1} v \right\rangle_{g_p} ,$$

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.

2.2.3 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[3]. In view of the previous definitions, the algorithm will preserve the geometry of the data only if the new manifold $(f(\mathcal{M}), \delta_s)$ is isometric to the original data manifold $(\mathcal{M}, \delta_r)$.

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

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

$$< u, v > = < df_p(v),\, df_p(v) >$$

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

---

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

The method developed by Nash to prove the existence of an isometric embedding is not practical when it comes to finding an isometric embedding for a data manifold. The problem is that the method involves tightly wrapping the embedding around extra dimensions, which, as observed by [Dreisigmeyer and Kirby, 2007, retrieved June 2010], may not be stable numerically[4].

Practically, however, as it was shown in Section 2.1.5, 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, retrieved June 2010, Behmardi and Raich, 2010, Zha and Zhang, 2003]).

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

[4]Recently, we became aware of a yet unpublished paper, which introduces an algorithm for an isometric embedding derived from Nash's theorem. We are enthusiastic about this achievement, but we note that achieving an isometric embedding via Nash does not invalidate what we propose here, which is an alternative approach in pursuit of the desirable goal of "preserving geometry".

2.3 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.

2.3.1 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 $\Delta_\mathcal{M} f \equiv div \cdot \nabla f$.*

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

$$\Delta_\mathcal{M} f = \frac{1}{\sqrt{\det(g)}} \frac{\partial}{\partial x^l} \left(\sqrt{\det(g)} g^{lk} \frac{\partial}{\partial x^k} f \right) . \tag{2.5}$$

In (2.5), 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. 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 (2.5) 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, or dual metric, at point $p \in U$ as expressed in local coordinates x, can be derived from*

$$g^{ij} = \frac{1}{2} \Delta_\mathcal{M} \left(x^i - x^i(p) \right) \left(x^j - x^j(p) \right) |_{x^i = x^i(p), x^j = x^j(p)} \tag{2.6}$$

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

Proof This follows directly from (2.5). 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 (2.5) 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 (2.6) refer to the fact that g^{ij} is the inverse, i.e. dual metric, of g for coordinates x^i and x^j. ■

With all the components of g^{-1}known, it is straightforward to compute its inverse and obtain $g(p)$. The power of Proposition 10 resides in the fact that the coordinate chart is arbitrary. Given a coordinate chart (or embeddding, as will be shown below), one can apply the *coordinate-free* Laplace-Beltrami operator as in (2.6) to recover g for that coordinate chart.

2.3.2 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 unnecessary, and in practice it is simpler to work directly from f until the coordinate chart 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*

$$< u, v >_{h(f(p))} \quad \equiv \quad < df_p^{\dagger}(u), df_p^{\dagger}(v) >_{g(p)}, \tag{2.7}$$

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 $h \equiv H$, (2.7) becomes

$$u^t J^t H J v \;=\; u^t G v \tag{2.8}$$

Hence,

$$H \;\equiv\; \left(J^t\right)^{\dagger} G J^{\dagger} \tag{2.9}$$

In particular, when $\mathcal{M} \subset \mathbb{R}^r$, with the 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

$$H(p) = \left(J^t\right)^{\dagger} \Pi(p)^t I_r \Pi(p) J^{\dagger}. \tag{2.10}$$

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 Definition 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

$$< df_p(u), df_p(v) >_{h(f(p))} = < df_p^\dagger \circ df_p(u), df_p^\dagger \circ df_p(v) >_{g(p)} = < u, v >_{g(p)} \qquad (2.11)$$

Therefore, h ensures that the embedding is isometric. Moreover, the null space of the pseudoinverse 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

$$< u, v >_{h(f(p))} = < df_p^\dagger(u), df_p^\dagger(v) >_{g(p)} = < 0, df_p^\dagger(v) >_{g(p)} = 0\,. \qquad (2.12)$$

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 2.3.1.

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*

$$\tilde{h}^{ij} = \Delta_\mathcal{M} \frac{1}{2} \left(f^i - f^i(p)\right)\left(f^j - f^j(p)\right)|_{f^i = f^i(p), f^j = f^j(p)} \qquad (2.13)$$

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 (2.5) means that

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

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

$$g^{kl}\frac{\partial f^i}{\partial x^l}\frac{\partial f^j}{\partial x^k} = (JG^{-1}J^t)_{ij} = \tilde{H}_{ij}\,. \tag{2.14}$$

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

$$\left(J^t\right)^\dagger G J^\dagger = \left(JG^{-1}J^t\right)^\dagger. \tag{2.15}$$

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

$$(AB)^\dagger = B^\dagger A^\dagger \tag{2.16}$$

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

Discussion

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 Proposition 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 2.6 to compute areas or volumes. An interesting extension of our approach would be to derive the exponential map for $f(U)$. However, computing all the geodesics of $f(U)$ is not practical unless the geodesics themselves are of interest for the application. In either case, computing h allows us to achieve our set goal for manifold learning, i.e. construct a collection of coordinate charts for $\mathcal{P}$. We note that 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 perform 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 2.6.

2.4 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 present our algorithm for the estimation procedure, called LearnMetric. Throughout, we assume that an appropriate embedding dimension $s \geq d$ is already selected and d is known.

2.4.1 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 2.1, the input data for a manifold learning algorithm is a set of points $\mathcal{P} = \{p_1, \ldots, p_n\} \subset \mathcal{M}$ where $\mathcal{M}$ is a compact Riemannian manifold. These points are assumed to be an i.i.d. sample with distribution π 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 $f_n : \mathcal{P} \to \mathbb{R}^s$, 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 $\mathcal{P}$, h_n will be defined as a positive semidefinite matrix over $\mathcal{P}$. With that in mind, we can hope to construct h_n by discretizing equation (2.13). In practice, this is acheived by replacing f with f_n and $\Delta_{\mathcal{M}}$ with some discrete estimator $\tilde{\mathcal{L}}_{\epsilon,n}$ that is consistent for $\Delta_{\mathcal{M}}$.

In what sense the estimators f_n, h_n, and $\tilde{\mathcal{L}}_{\epsilon,n}$ can be said to be consistent will be discussed in Section 2.5 below. In the meantime, we still need to clarify how to obtain $\tilde{\mathcal{L}}_{\epsilon,n}$. The most common approach, and the one we favor here, is to start by

considering the "diffusion-like" operator $\tilde{\mathcal{D}}_{\epsilon,\lambda}$ defined via the heat kernel w_ϵ (see (2.1)):

$$\begin{aligned}
\tilde{\mathcal{D}}_{\epsilon,\lambda}(f)(x) &= \int_{\mathcal{M}} \frac{\tilde{w}_{\epsilon,\lambda}(x,y)}{\tilde{t}_{\epsilon,\lambda}}) f(y)\pi(y)dV_g(y)\,, \text{ with } x \in \mathcal{M} \text{ and where} \quad (2.17)\\
\tilde{t}_{\epsilon,\lambda}(x) &= \int_{\mathcal{M}} \tilde{w}_{\epsilon,\lambda}(x,y))\pi(y)dV_g(y)\,, \text{ and } w_{\epsilon,\lambda} = \frac{w_\epsilon(x,y)}{t^\lambda_\epsilon(x)t^\lambda_\epsilon(y)}\,, \text{ while}\\
t_\epsilon(x) &= \int_{\mathcal{M}} w_\epsilon(x,y)\pi(y)dV_g(y)\,,
\end{aligned}$$

Coifman and Lafon [2006] showed that $\tilde{\mathcal{D}}_{\epsilon,\lambda}(f) = f + \epsilon c\Delta_{\mathcal{M}} f + \mathcal{O}(\epsilon^2)$ provided $\lambda = 1$, $f \in \mathcal{C}^3(\mathcal{M})$, and where c is a constant that depends on the choice of kernel w_ϵ[5]. Here, λ is introduced to guarantee the appropriate limit in cases where the sampling density π is not uniform on $\mathcal{M}$.

Now that we have obtained an operator that we know will converge to $\Delta_{\mathcal{M}}$, i.e.

$$\tilde{\mathcal{L}}_{\epsilon,1}(f) \equiv \frac{\tilde{\mathcal{D}}_{\epsilon,1}(f) - f}{c\epsilon} = \Delta_{\mathcal{M}} f + \mathcal{O}(\epsilon)\,, \quad (2.18)$$

we turn to the discretized problem, since we are dealing with a finite sample of points from $\mathcal{M}$.

Discretizing (2.17) entails using the finite sample approximation:

$$\begin{aligned}
\tilde{\mathcal{D}}_{\epsilon,n}(f)(x) &= \sum_{p_i \in \mathcal{P}} \frac{w_{\epsilon,\lambda}(x,p_i)}{\tilde{t}_{\epsilon,\lambda,n}(x)} f(p_i)\,, \text{ with } x \in \mathcal{M} \text{ and where} \quad (2.19)\\
\tilde{t}_{\epsilon,\lambda,n}(x) &= \sum_{p_i \in \mathcal{P}} \tilde{w}_{\epsilon,\lambda}(x,p_i)\,, \text{ and } \tilde{w}_{\epsilon,\lambda} = \frac{w_\epsilon(x,y)}{t^\lambda_\epsilon(x)d^\lambda_\epsilon(y)}\,, \text{ while}\\
t_\epsilon(x) &= \sum_{p_i \in \mathcal{P}} w_\epsilon(x,p_i)\,,
\end{aligned}$$

and (2.18) now takes the form

$$\tilde{\mathcal{L}}_{\epsilon,n}(f) \equiv \frac{\tilde{\mathcal{D}}_{\epsilon,1}(f) - f}{c\epsilon} = \Delta_{\mathcal{M}} f + \mathcal{O}(\epsilon)\,. \quad (2.20)$$

Operator $\tilde{\mathcal{L}}_{\epsilon,n}$ is known as the ***geometric graph Laplacian*** (Zhou and Belkin [2011]). We will refer to it simply as graph Laplacian in our discussion, since it is the only type of graph Laplacian we will need.

---

[5] In the case of heat kernel (2.1), $c = 1/4$, which - crucially - is independent of the dimension of $\mathcal{M}$.

Algorithm 1 GraphLaplacian

Input: Weight matrix W, bandwidth $\sqrt{\epsilon}$, and λ

$D \leftarrow \text{diag}(W\mathbf{1})$

$\tilde{W} \leftarrow D^{-\lambda} W D^{-\lambda}$

$\tilde{D} \leftarrow \text{diag}(\tilde{W}\mathbf{1})$

$L \leftarrow (c\epsilon)^{-1}(\tilde{D}^{-1}\tilde{W} - I_n)$

Return L

Note that since is $\mathcal{M}$ unknown, it is not clear when $x \in \mathcal{M}$ and when $x \in \mathbb{R}^r \setminus \mathcal{M}$. The need to define $\tilde{\mathcal{L}}_{\epsilon,n}(f)$ for all x in $\mathcal{M}$ is mainly to study its asymptotic properties. In practice however, we are interested in the case of $x \in \mathcal{P}$. The kernel $w_\epsilon(x, y)$ then takes the form of weighted neighborhood graph $\mathcal{G}_{w_\epsilon}$, which we denote by the weight matrix $W_{i,j} = w_\epsilon(p_i, p_j)$, $p_i, p_j \in \mathcal{P}$. In fact, (2.19) and (2.20) can be expressed in terms of matrix operations when $x \in \mathcal{P}$ as done in Algorithm 1.

With $\tilde{\mathcal{L}}_{\epsilon,n}$, the discrete analogue to (2.5), clarified, we are now ready to introduce the central algorithm of this chapter.

2.4.2 The LearnMetric *Algorithm*

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

1. construct a weighted neighborhood graph $\mathcal{G}_{w_\epsilon}$

2. calculate the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$

3. map the data $p \in \mathcal{P}$ to $f_n(p) \in \mathbb{R}^s$ by GENERICEMBED

4. apply the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$ to the coordinates f_n to obtain the embedding metric h

Figure 3 contains the LEARNMETRIC algorithm in pseudocode. The subscript n in the notation indicates that these are discretized, "sample" quantities, (i.e. f_n is a vector and $\tilde{\mathcal{L}}_{\epsilon,n}$ is a matrix) as opposed to the continuous quantities (functions, operators) that we were considering in the previous sections. We now move to describing each of the steps of the algorithm.

The first two steps, common to many manifold learning algorithms, have already been described in subsections 2.1.1 and 2.4.1 respectively. The third step calls for obtaining an embedding of the data points $p \in \mathcal{P}$ in $\mathbb{R}^s$. This can be done using any one of the many existing manifold learning algorithms (GENERICEMBED), such as the Laplacian Eigenmaps, Isomap or Diffusion Maps.

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, LLE, and LTSA use a neighborhood graph and/or similarities in order to obtain an embedding. In addition, Diffusion Maps and Eigemaps obtain an embedding for the eigendecomposition $\tilde{\mathcal{L}}_{\epsilon,n}$ 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. Step 4, applies the $n \times n$ Laplacian matrix $\tilde{\mathcal{L}}_{\epsilon,n}$ obtained in Step 2 to pairs $f_n^i,\ f_n^j$ of embedding coordinates of the data obtained in Step 3. We use the symbol $\cdot$ to refer to the elementwise product between two vectors. 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 $\mathcal{P} \subset \mathcal{M}$. Hence, equation (2.21) is equivalent to applying equation (2.13) to all the points of $p \in \mathcal{P}$ at once. The result are the vectors $\tilde{h}_n^{ij}$, each of which is an n-dimensional vector, with an entry for each $p \in \mathcal{P}$. Then, in Step 4, b, at each embedding point $f(p)$ the embedding metric $h_n(p)$ is computed as the matrix (pseudo) inverse of $[\tilde{h}_n^{ij}(p)]^{ij=1:s}$.

If the embedding dimension s is larger than the manifold dimension d, we will obtain the rank d embedding metric h_n; otherwise, we will obtain the Riemannian metric g_n. For the former, $h_n^\dagger$ will have a theoretical rank d, but numerically it might have rank between d and s. As such, it is important to set to zero the $s - d$ smallest singular values of $h_n^\dagger$ when computing the pseudo-inverse. This is the key reason why s and d need to be known in advance. Failure to set the smallest singular values to zero will mean that h_n will be dominated by noise. Although estimating d is outside the scope of this work, it is interesting to note that the singular values of $h_n^\dagger$ may offer a window into how to do this by looking for a "singular value gap".

In summary, the principal novelty in our LearnMetric 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* f. Thus, once augmented with their corresponding h, all embeddings become geometrically equivalent to each other, and to the orginal data manifold $(\mathcal{M}, g)$.

2.4.3 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

Algorithm 2 PseudoInverse

Input: Embedding metric $\tilde{h}_n(p)$ and intrinsic dimension d

$[U, \Lambda] \leftarrow$ EigenDecomposition$(\tilde{h}_n(p))$ where U is the matrix of column eigenvectors of $\tilde{h}_n(p)$ ordered by their eigenvalues Λ from largest to smallest.

$\Lambda \leftarrow \Lambda(1:d)$ (keep d largest eigenvalues)

$\Lambda^\dagger \leftarrow \text{diag}(1/\Lambda)$

$h_n(p) \leftarrow U\Lambda^\dagger U^t$ (obtain rank d pseudo-inverse of $\tilde{h}_n(p)$)

Return $h_n(p)$

(Ram et al. [2009]). In any event, this computationally intensive preprocessing step is required by all of the well known embedding algorithms, and would remain necessary even if one's goal were solely to embed the data, and not to compute the Riemannian metric.

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

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

Finally, the newly introduced Step 4 involves obtaining an $s \times s$ matrix for each of the n points, and computing its pseudoinverse. Obtaining the $\tilde{h}_n$ matrices takes $\mathcal{O}(n^2)$ operations ($\mathcal{O}(nk)$ for sparse $\tilde{\mathcal{L}}_{\epsilon,n}$ matrix) times $s \times s$ entries, for a total of $s^2 n^2$ 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 quadratic in the data set size n and embedding dimension s, and cubic in the intrinsic dimension d.

Algorithm 3 LearnMetric

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

1. Construct the weighted neighborhood graph with weight matrix W given by $W_{i,j} = \exp\left(-\frac{1}{\epsilon}||p_i - p'_j||^2\right)$ for every points $p_i, p_j \in \mathcal{P}$.

2. Construct the Laplacian matrix $\tilde{\mathcal{L}}_{\epsilon,n}$ using Algorithm 1 with input W, $\sqrt{\epsilon}$, and $\lambda = 1$.

3. Obtain the *embedding coordinates* $f_n(p) = (f_n^1(p), \ldots, f_n^s(p))$ of each point $p \in \mathcal{P}$ by

$$[f_n(p)]_{p\in\mathcal{P}} \;=\; \textsc{GenericEmbed}(\mathcal{P}, s, d, \sqrt{\epsilon})$$

4. Calculate the *embedding metric* h_n for each point

   (a) For i and j from 1 to s calculate the column vector $\tilde{h}_n^{ij}$ of dimension $n = |\mathcal{P}|$ by

$$\tilde{h}_n^{ij} \;=\; \frac{1}{2}\left[\tilde{\mathcal{L}}_{\epsilon,n}(f_n^i \cdot f_n^j) - f_n^i \cdot (\tilde{\mathcal{L}}_{\epsilon,n} f_n^j) - f_n^j \cdot (\tilde{\mathcal{L}}_{\epsilon,n} f_n^i)\right] \tag{2.21}$$

   (b) For each data point $p \in \mathcal{P}$, form the matrix $\tilde{h}_n(p) = [\tilde{h}_n^{ij}(p)]_{i,j\in 1,\ldots,s}$. The embedding metric at p is then given by $h_n(p) =$ PseudoInverse$(\tilde{h}_n(p), d)$

Return $(f_n(p), h_n(p))_{p\in\mathcal{P}}$

2.5 Convergence of the Pushforward Riemannian Metric Estimator

The convergence of graph Laplacians to the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ 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]). However, having an estimator $\tilde{\mathcal{L}}_{\epsilon,n}$ that converges to $\Delta_{\mathcal{M}}$ is not sufficient to guarantee that Algorithm 6 produces an estimator h_n that converges to the embedding metric h. Because the embedding metric estimator h_n depends on having an embedding algorithm to start with, any result about the convergence of the estimator requires guarantees about the embedding algorithm itself:

1. The embedding algorithm must construct a proper embedding $f : \mathcal{M} \to \mathbb{R}^s$ of $\mathcal{M}$ as in Definition 3

2. The estimator f_n of f constructed by the embedding algorithm must converge to f.

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 in the sense of Definition 3. It is not the aim of this section to address this problem for all the popular embedding algorithms; we simply underline the fact that estimating the embedding metric h in manifold learning algorithms involves two assumptions that are not always verified. Nevertheless, for illustrative purposes, we will consider the special case of Laplacian Eigenmap and show that it can be constructed so as to satisfy condition 1, while condition 2 will follow readily from (Giné and Koltchinskii [2006], Belkin and Niyogi [2007]).

However, the two conditions on the embedding algorithm described above and the convergence of $\tilde{\mathcal{L}}_{\epsilon,n}$ to $\Delta_{\mathcal{M}}$ are not sufficient to prove that h_n will converge to

the embedding metric h. 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 LearnMetric, means that to establish convergence of h_n, it must also be true that $\forall i, j \in \{1, \ldots, s\}$

$$\tilde{\mathcal{L}}_{\epsilon_n,n}\left(\bar{f}_n^i(x)\right)\left(\bar{f}_n^j(x)\right)|_{x=p} \;\rightarrow\; \Delta_{\mathcal{M}}\left(\bar{f}^i(x)\right)\left(\bar{f}^j(x)\right)|_{x=p}\,, \tag{2.22}$$

where we made use of the shorthand notation $\bar{f}_n(x) \equiv f_n(x) - f_n(p)$ and $\bar{f}(x) \equiv f(x) - f(p)$.

Note that we have used $\tilde{\mathcal{L}}_{\epsilon_n,n}$ instead $\tilde{\mathcal{L}}_{\epsilon,n}$ in (2.22) to highlight the fact that for $\tilde{\mathcal{L}}_{\epsilon,n}$ to converge to $\Delta_{\mathcal{M}}$, it is necessary for the bandwidth $\sqrt{\epsilon}_n \to 0$ as $n \to \infty$. The rate at which $\sqrt{\epsilon}_n$ needs to go to zero depends on how $\tilde{\mathcal{L}}_{\epsilon,n}$ is constructed and on the type of convergence considered. For example, if $\tilde{\mathcal{L}}_{\epsilon_n,n}$ is constructed as in equation (2.20), then $\sqrt{\epsilon}_n \to 0$ and $n\sqrt{\epsilon}_n^{d+2} \to 0$ is sufficient to prove that $\tilde{\mathcal{L}}_{\epsilon_n,n}$ converges uniformly a.s. to $\Delta_{\mathcal{M}}$, i.e. $||\tilde{\mathcal{L}}_{\epsilon_n,n}(f)(x) - \Delta_{\mathcal{M}}(f)(x)||_\infty \overset{a.s.}{\to} 0\,,\ \forall f \in C^2(\mathcal{M})$ (Ting et al. [2010]).

As this example illustrates, the consistency of $\tilde{\mathcal{L}}_{\epsilon_n,n}$ is generally established for a "test" function $f \in C^2(\mathcal{M})$, or $f \in C^2(\mathcal{M})$ as the case might be, 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 is why extra care is needed to prove that h_n converges to embedding metric h.

Theorem 14 (Convergence of Embedding Metric) *Let $f : \mathcal{M} \to \mathbb{R}^s$ be an embedding of $\mathcal{M}$ satisfying conditions 1 and 2. Let $f_n : \mathcal{P} \to \mathbb{R}^s$ be an estimator of f such that $||f - f_n||_{\infty,\mathcal{P}} \overset{p}{\to} 0$ for the sup-norm $||\cdot||_\infty$ on $C^3(\mathcal{M})$ restricted to points in $\mathcal{P}$. If*

$$\left|\left|\Delta_{\mathcal{M}} f^i - \tilde{\mathcal{L}}_{\epsilon_n,n} f_n^i\right|\right|_{\infty,\mathcal{P}} \overset{p}{\to} 0 \tag{2.23}$$

as $n \to \infty$ where $\tilde{\mathcal{L}}_{\epsilon_n,n}$ is constructed as in (2.18), then estimator h_n constructed from LearnMetric *converges element-wise to embedding metric h in the sense that*

$$||h_{ij} - h_{n,ij}||_{\mathcal{P}} \overset{p}{\to} 0\,,\ \forall i, j \in \{1, \ldots, s\}. \tag{2.24}$$

Proof From 2.23 it follows that $\left|\left|\Delta_{\mathcal{M}} f^i f^j - \tilde{\mathcal{L}}_{\epsilon_n,n} f_n^i f_n^j\right|\right|_{\infty,\mathcal{P}} \xrightarrow{p} 0$ (see Lemma 25) and hence Equation 2.22 follows, in probability in the $||\cdot||_{\infty,\mathcal{P}}$ sense, from the fact that the graph Laplacian satisfies $\tilde{\mathcal{L}}_{\epsilon_n,n}(f_n(x) - f_n(p)) = \tilde{\mathcal{L}}_{\epsilon_n,n}(f_n(x))$.

Equation 2.22 is equivalent to element-wise convergence of $\tilde{h}_n$. Because $\tilde{h}_n$ is finite-dimensional (i.e. $s \times s$), then element-wise convergence implies convergence of $\tilde{h}_n$.

Finally, a minor point remains: the convergence of $\tilde{h}_n$ does not guarantee convergence of its pseudoinverse, i.e. h_n. This is because the pseudoinverse $\dagger$ is not a continuous map. This does not present a problem in our case, however, since $\tilde{h}$ and the sequence $\tilde{h}_n$ has rank d for any n. In other words, the probability of $\tilde{h}_n$ being at a discontinuity point of $\dagger$ is zero. ■

We now turn we to the special case of the Laplacian Eigenmap, where (2.23) 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 that is necessary. The fact that the Eigenmap and Diffusion Maps are some of the most interesting embeddings is merely an added bonus (Belkin et al. [2006], Lu et al. [2007]).

2.5.1 Consistency of Laplacian Eigenmap

The goal of this section is to show that the conditions of Theorem 14 are indeed satisfied in this case 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 Laplacian Eigenmap on a compact manifold with uniform sampling density and $\text{Vol}(\mathcal{M}) = 1$. 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. To prove condition 1, we simply need to show that there is a map $\Phi = (\phi_1, \ldots, \phi_s)$ constructed from a finite number of eigenfunctions of $\Delta_{\mathcal{M}}$ 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 $\mathcal{M}$ into $\mathbb{R}^s$.

Proposition 15 *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 separability, 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

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

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

Defining the map $\Phi|_{L_j} \equiv (\phi_1, \phi_2 \ldots, \phi_{l_d})$ 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 $R \subset \{1, 2, \ldots\}$ 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 $\Phi|_m = (\phi_1, \phi_2 \ldots, \phi_s)$. 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 Laplacian Eigenmap. 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 (2.23):

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

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

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 $\mathcal{M}$.

2.6 Experiments

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

2.6.1 Embedding Metric as a Measure of Local Distortion

The first set of experiments is intended to illustrate the output of the LearnMetric algorithm. Figure 2.2 shows the embedding of a 2D hourglass-shaped manifold. Diffusion Maps, the embedding algorithm we used (with $s = 3$, $\lambda = 1$) distorts the shape by excessively flattening the top and bottom. LearnMetric outputs a $s \times s$ quadratic form for each point $p \in \mathcal{P}$, represented as ellipsoids centered at p. Practically, this means that the ellipsoids are flat along one direction $T_{f_n(p)} f_n(\mathcal{M})^{\perp}$, and two-dimensional because $d = 2$, i.e. h_n has rank 2. If the embedding correctly recovered the local geometry, $h_n(p)$ would equal $I_3|_{f_n(\mathcal{M})}$, the identity matrix restricted to $T_{f_n(p)} f_n(\mathcal{M})$: it would define a circle in the tangent plane of $f_n(\mathcal{M})$, for each p. We see that this is the case in the girth area of the hourglass, where the ellipses are circular. Near the top and bottom, the ellipses' orientation and elongation points in the direction where the distortion took place and measures the amount of (local) correction needed.

The more the space is compressed in a given direction, the more elongated the embedding metric "ellipses" will be, so as to make each vector "count for more". Inversely, the more the space is stretched, the smaller the embedding metric will be. This is illustrated in Figure 2.2.

We constructed the next example to demonstrate how our method applies to the popular Sculpture Faces data set. This data set 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 data set consists of $n = 698$ 64×64 gray images of faces. The faces are allowed to vary

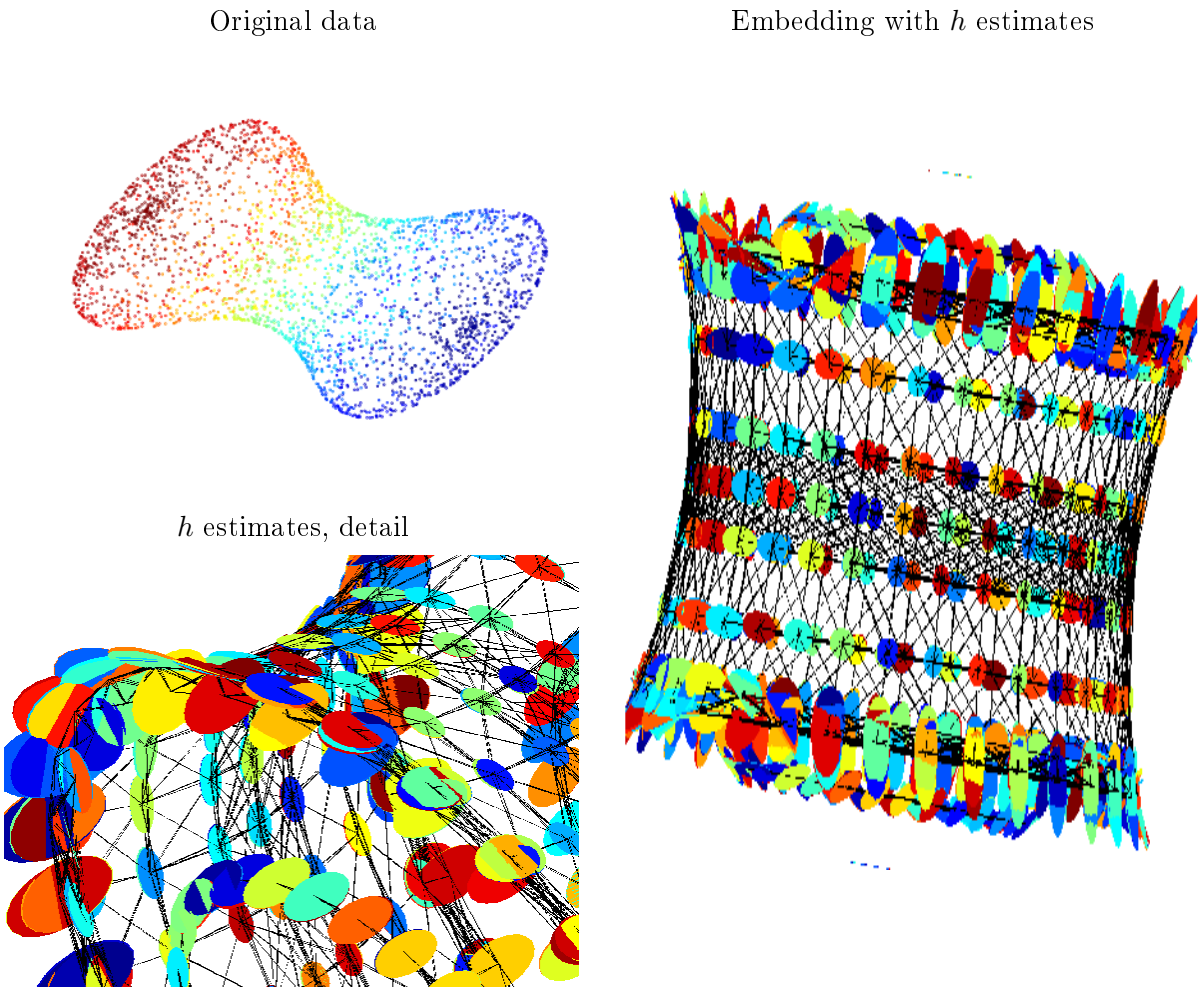


Figure 2.2: Estimation of h for a 2D hourglass-shaped manifold in 3D space. The embedding is obtained by Diffusion Maps. The ellipses attached to each point represent the embedding metric h_n estimate for this embedding. At each data point $p \in \mathcal{P}$, $h_n(p)$ is a 3×3 symmetric semi-positive 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.

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 $[0,1]^{4096}$.

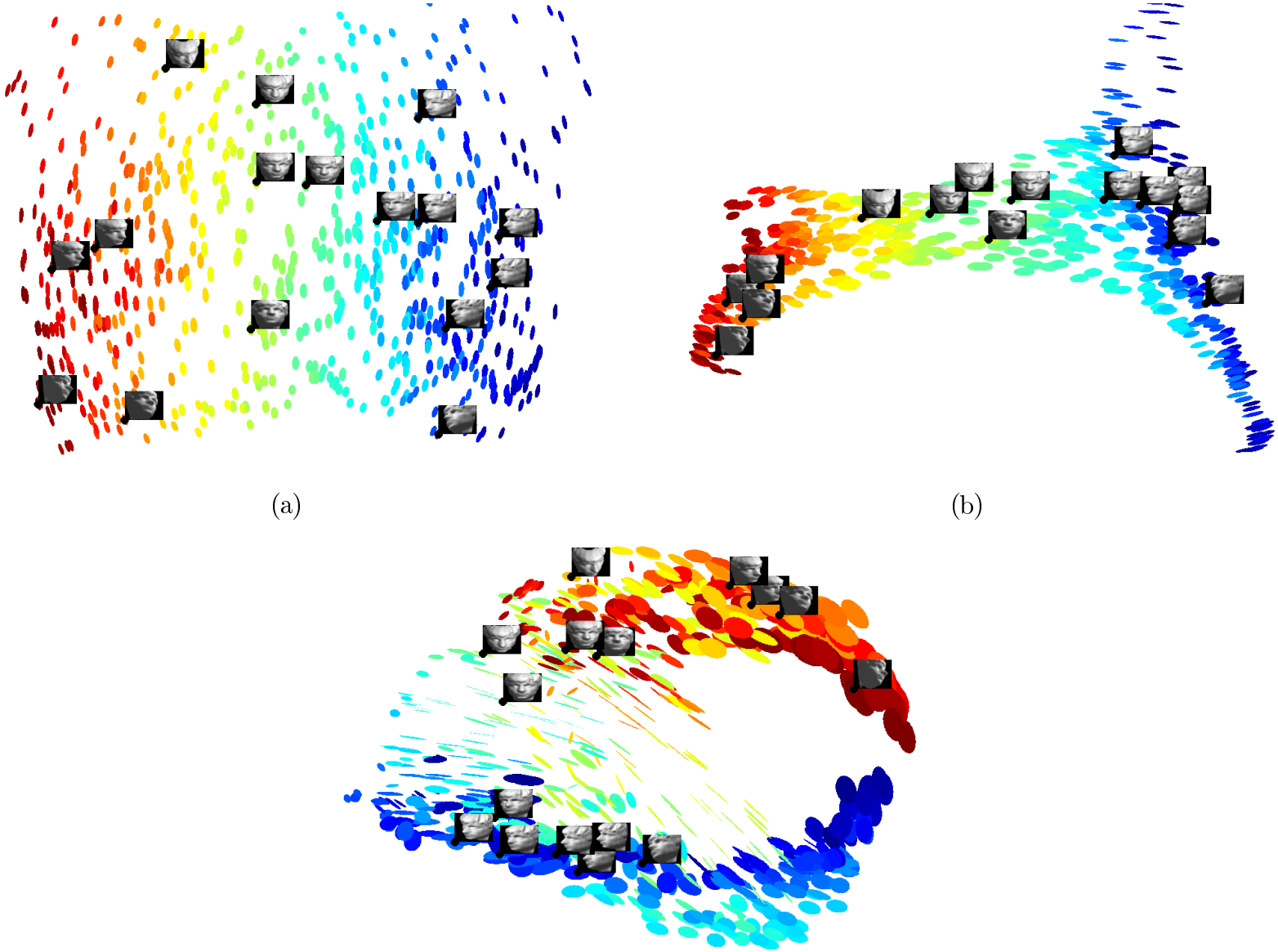
As expected given its focus on preserving the geodesic distances, the Isomap seems to recover the simple rectangular geometry of the data set, as Figure 2.3 shows. LTSA, on the other hand, distorts the original data, particularly in the corners, where the Riemannian metric takes the form of thin ellipses. Diffusion Maps distorts the original geometry the most. The fact that the embedding for which we have theoretical guarantees of consistency causes the most distortion highlights, once more, that consistency provides no information about the level of distortion that may be present in the embedding geometry.

Our next example, Figure 2.4, shows an almost isometric reconstruction of a common example, the Swiss roll with a rectangular hole in the middle. This is a popular test data set 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 2.4 (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 along with a few two-dimensional examples in the context of data visualization in the following section.

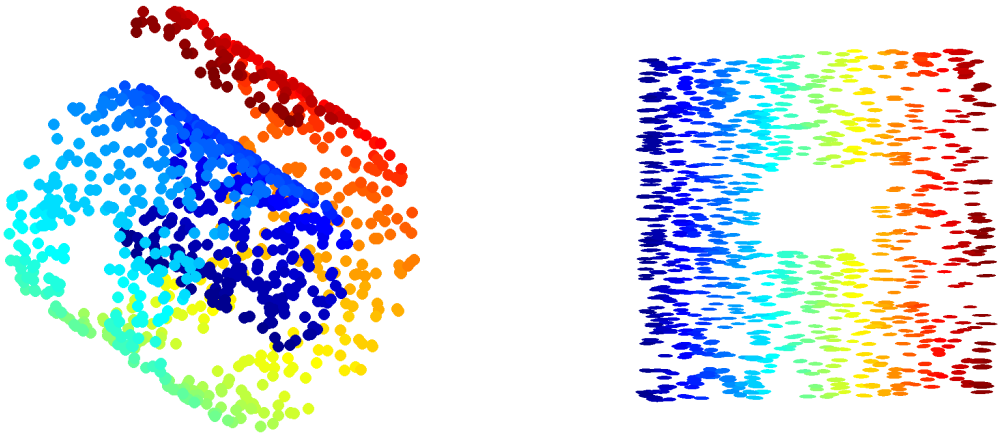
2.6.2 Locally Isometric Visualization

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



(a) (b) (c)

Figure 2.3: 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 Diffusion Maps ($\lambda = 1$) (c).



(a) (b)

Figure 2.4: (a) Swissroll with a hole in $\mathbb{R}^3$. (b) LTSA embedding of the manifold in $\mathbb{R}^2$ along with metric.

for all manifolds; in particular, only flat manifolds with intrinsic dimension below 3 can be "correctly visualized" according to this definition. This problem has been long known in cartography: a wide variety of *cartographic projections* of the Earth have been developed to map parts of the 2D sphere onto a plane, and each aims to preserve a different family of geometric quantities. For example, projections used for navigational, meteorological or topographic charts focus on maintaining angular relationships and accurate shapes over small areas; projections used for radio and seismic mapping focus on maintaining accurate distances from the center of the projection or along given lines; and projections used to compare the size of countries focus on maintaining accurate relative sizes (Snyder [1987]).

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

However, it offers a natural way of producing *locally* isometric embeddings, and

therefore locally correct visualizations for two- or three-dimensional manifolds. The procedure is based on the transformation of the points that will guarantee that the embedding is the identity matrix.

Given $(f_n(\mathcal{P}), h_n(\mathcal{P}))$ Metric Embedding of $\mathcal{P}$

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

Display $\mathcal{P}$ in coordinates $\tilde{f}_n$

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

Figures 2.5, 2.6 and 2.7 exemplify this procedure for the Swiss roll with a rectangular hole of Figure 2.4 embedded respectively by LTSA, Isomap and Diffusion Maps. In these figures, we use the Procrustes method (Goldberg and Ritov [2009]) to align the original neighborhood of the chosen point p with the same neighborhood in an embedding. The Procrustes method minimizes the sum of squared distances between corresponding points between all possible rotations, translations and isotropic scalings. The residual sum of squared distances is what we call the *Procrustes dissimilarity*. Its value is close to zero when the embedding is locally isometric around p.

2.6.3 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

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

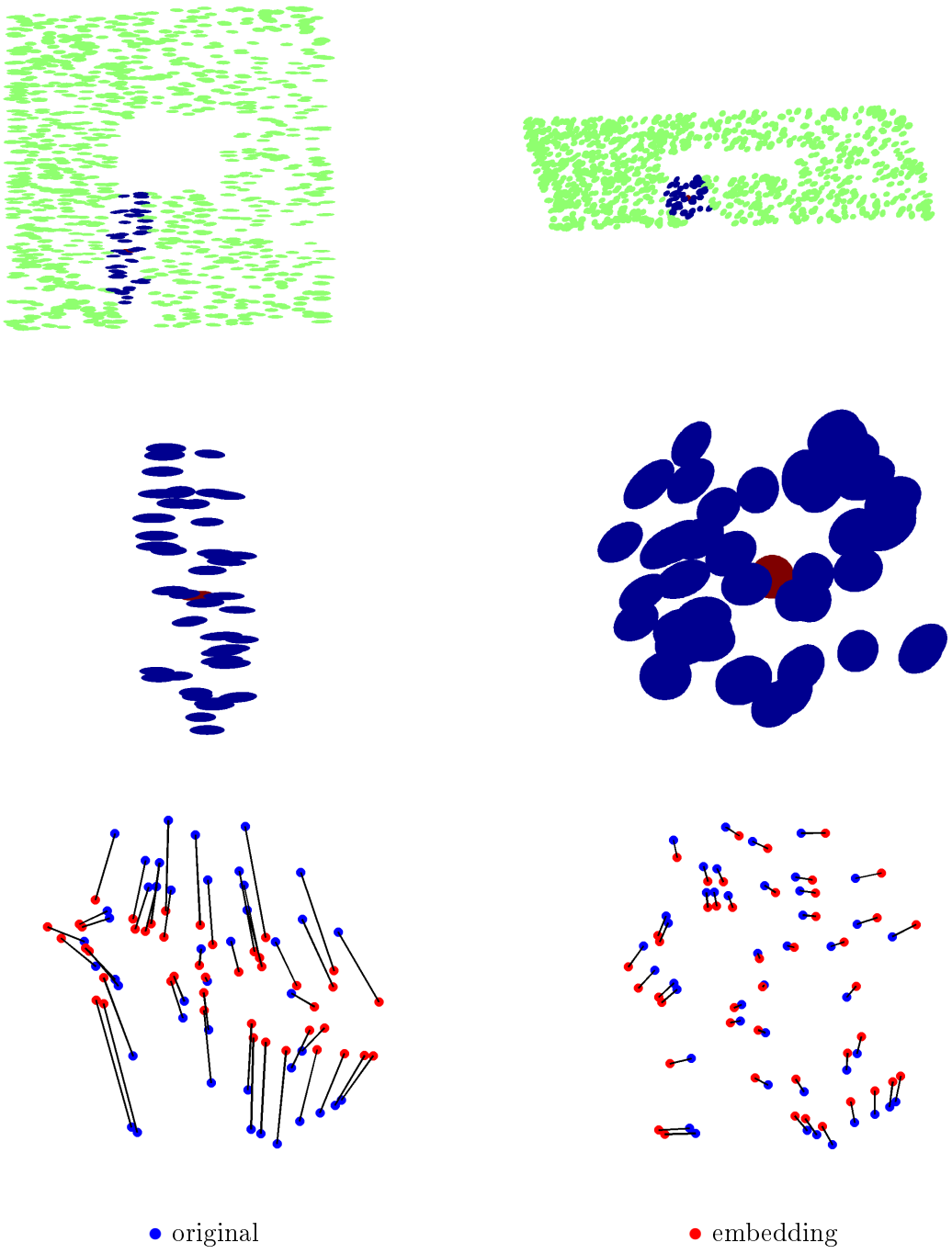


Figure 2.5: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by LTSA. Top left: LTSA embedding with selected point p (red) and its neighbors (blue). Top right: locally isometric embedding. Middle left: Neighborhood of p for the LTSA embedding. Middle right: Neighborhood of p for the locally isometric embedding. Bottom left: Procrustes between the neighborhood of p for the LTSA embedding and the original manifold projected on $T_p\mathcal{M}$; dissimilarity measure: $D = 0.30$. Bottom right: Procrustes between the locally isometric embedding and the original manifold; dissimilarity measure: $D = 0.02$.

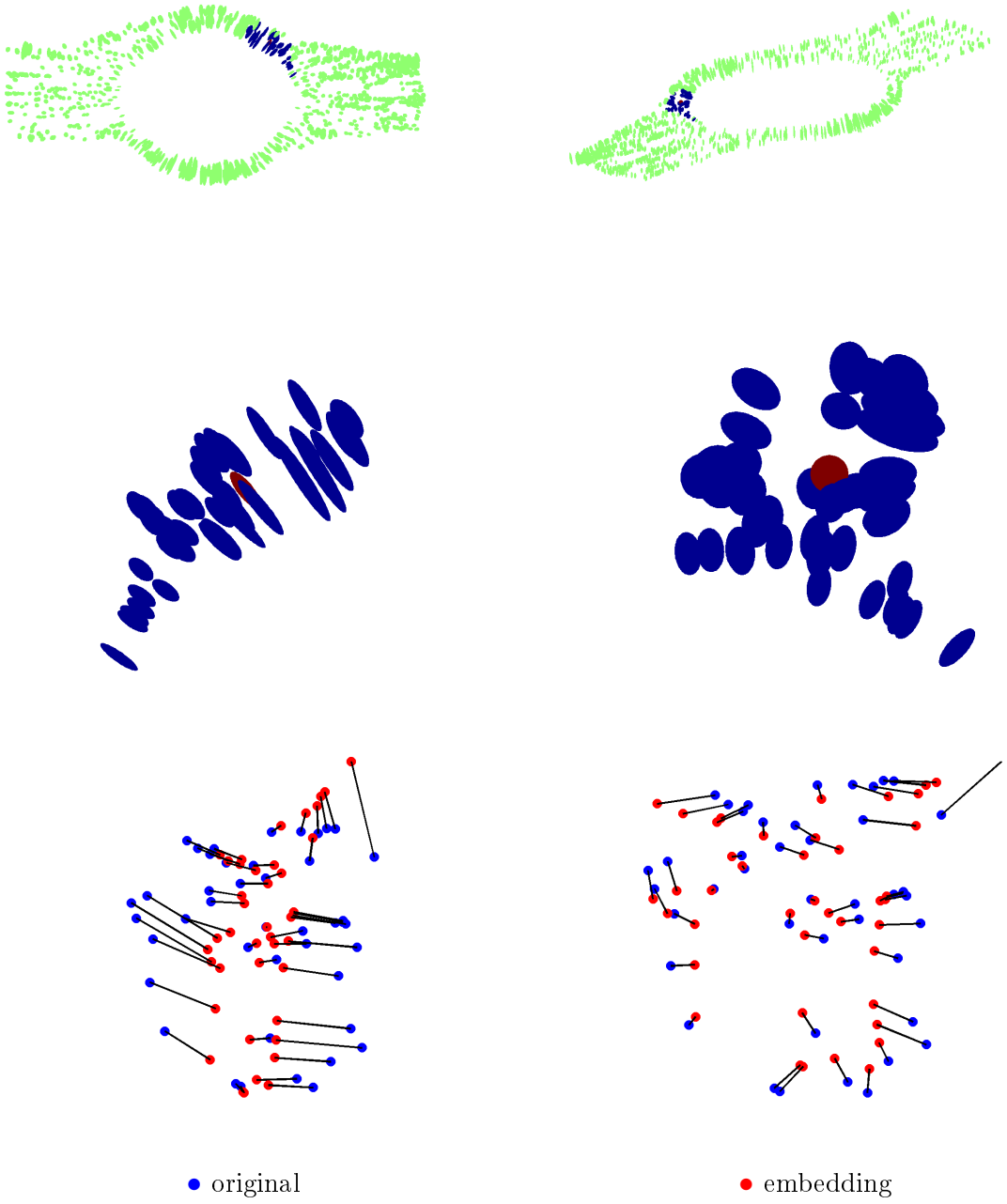


Figure 2.6: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by Isomap. Top left: Isomap embedding with selected point p (red), and its neighbors (blue). Top right: locally isometric embedding. Middle left: Neighborhood of p for the Isomap embedding. Middle right: Neighborhood of p for the locally isometric embedding. Bottow left: Procrustes between the neighborhood of the p for the Isomap embedding and the original manifold projected on $T_p\mathcal{M}$; dissimilarity measure: $D = 0.21$. Bottom right: Procrustes between the locally isometric embedding and the original manifold; dissimilarity measure: $D = 0.06$.

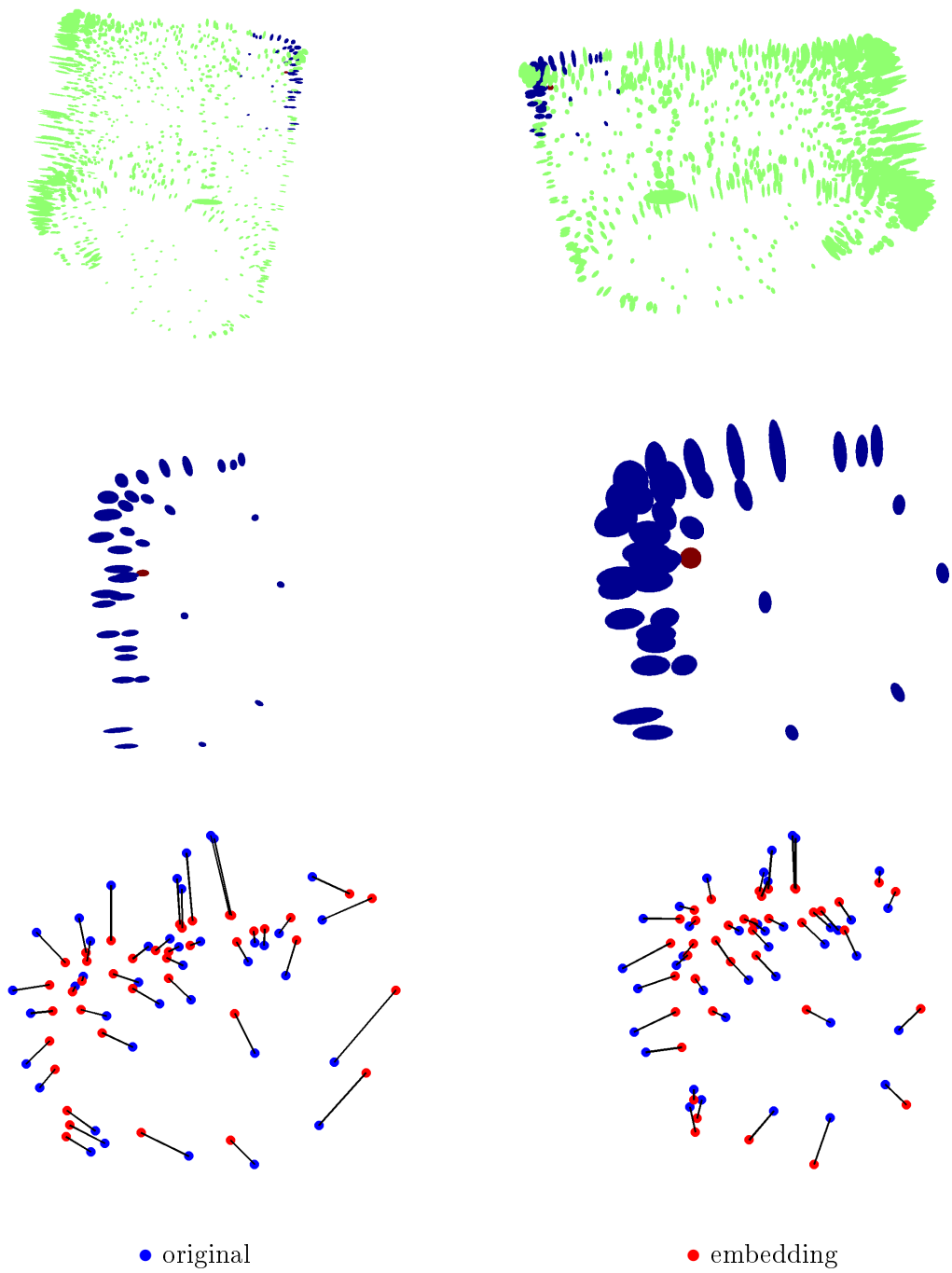


Figure 2.7: Locally isometric visualization for the Swiss roll with a rectangular hole, embedded in $d = 2$ dimensions by Diffusion Maps ($\lambda = 1$). Top left: DM embedding with selected point p (red), and its neighbors (blue). Top right: locally isometric embedding. Middle left: Neighborhood of p for the DM embedding. Middle right: Neighborhood of p for the locally isometric embedding. Bottow left: Procrustes between the neighborhood of the p for the DM embedding and the original manifold projected on $T_p\mathcal{M}$; dissimilarity measure: $D = 0.10$. Bottom right: Procrustes between the locally isometric embedding and the original manifold; dissimilarity measure: $D = 0.07$.

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 $\mathcal{P}$, obtaining the Isomap, LTSA and DM 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 $||f_n(p) - f_n(p')||$. 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, $||f_n(p) - f_n(p')||$ 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 DM, 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 $||f_n(q_i) - f_n(q'_{i-1})||$:

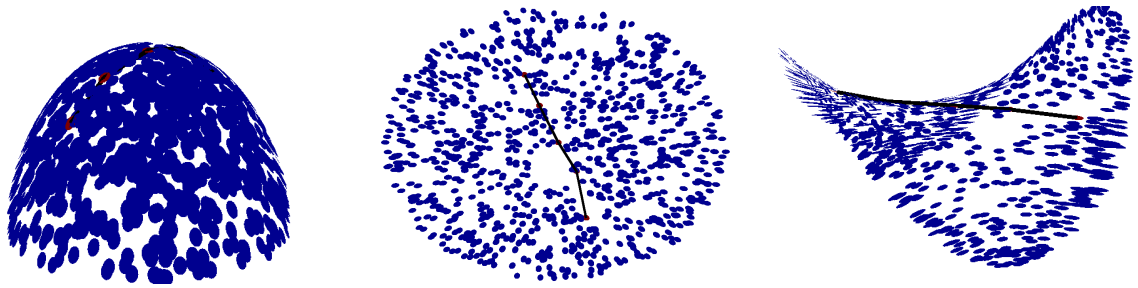
$$d_{\mathcal{G}}(p, p') \;=\; \min_{\text{paths}}\{\sum_{i=1}^{l} ||f_n(q_i) - f_n(q_{i-1})||,\ (q_0 = p,\ q_1,\ q_2, \ldots\ q_l = p') \text{ path in } \mathcal{G}\}. \tag{2.25}$$

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:

$$\hat{d}_{\mathcal{M}}(p, p') \;=\; \min_{\text{paths}}\{\sum_{i=1}^{l} \mathcal{H}(q_i, q_{i-1}),\ (q_0 = p,\ q_1,\ q_2, \ldots\ q_l = p') \text{ path in } \mathcal{G}\} \tag{2.26}$$



(a) (b) (c)

Figure 2.8: 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) Diffusion Maps (c) Isomap.

where

$$\begin{aligned}\mathcal{H}(q_i, q_{i-1}) &= \frac{1}{2}\sqrt{(f_n(q_i) - f_n(q_{i-1}))^t h_n(q_i)(f_n(q_i) - f_n(q_{i-1}))} \\ &\quad + \frac{1}{2}\sqrt{(f_n(q_i) - f_n(q_{i-1}))^t h_n(q_{i-1})(f_n(q_i) - f_n(q_{i-1}))} \qquad (2.27)\end{aligned}$$

is the discrete analog of the path-length formula (2.3) for the Voronoi tesselation of the space. By Voronoi tesselation, we mean the partition of the space into sets based on $\mathcal{P}$ such that each set consists of all points closest to a single point in $\mathcal{P}$ than any other. Figure 2.8 shows the manifolds that we used in our experiments, and Table 2.1 displays the estimated distances.

As expected, for the original data, $||p - p'||$ necessarily underestimates $d_{\mathcal{M}}$, while $d_{\mathcal{G}}$ is a very good approximation of $d_{\mathcal{M}}$, since it follows the manifold more closely. Meanwhile, the opposite is true for Isomap. The naive distance $||f_n(p) - f_n(p')||$ is close to the geodesic by construction, while $d_{\mathcal{G}}$ overestimates $d_{\mathcal{M}}$ since $d_{\mathcal{G}} \geq ||f_n(p) - f_n(p')||$ by the triangle inequality. Not surprisingly, for LTSA and Diffusion Maps, the estimates $||f_n(p) - f_n(p')||$ and $d_{\mathcal{G}}$ have no direct correspondence with the distances of the original data since these algorithms make no attempt at preserving absolute

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

Table 2.1: Distance estimates (mean and standard deviation) for a sample size of $n = 2000$ points was used for all embeddings while the standard deviations were estimated by repeating the experiment 5 times. The relative errors in the last column were computed with respect to the true distance $d = \pi/2 \simeq 1.5708$.

distances.

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.

2.6.4 Volume Estimation

The last set of our experiments demonstrates the use of the Riemannian metric in estimating two-dimensional volumes: areas. We used an experimental procedure similar to the case of geodesic distances, in that we created a two-dimensional manifold, and selected a set W on it. We then estimated the area of this set by generating a sample from the manifold, embedding the sample, and computing the area in the embedding space using a discrete form of (2.4).

One extra step is required when computing areas that was optional when com-

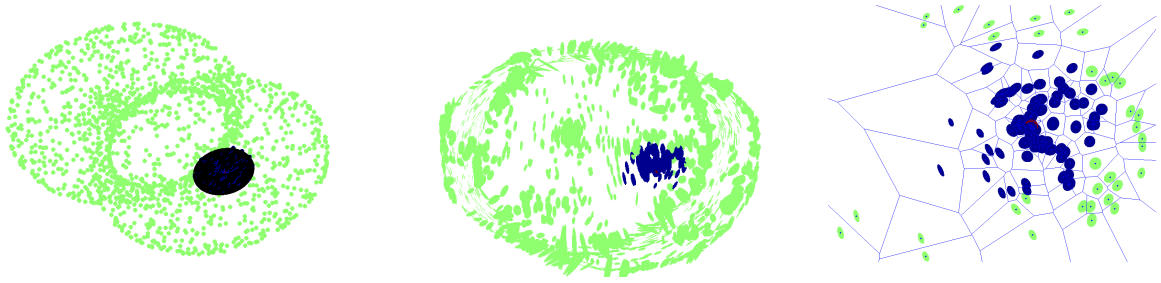
puting distances: we need to construct coordinate chart(s) to represent the area of interest. Indeed, to make sense of the Euclidean volume element $dx^1 \dots dx^d$, we need to work in $\mathbb{R}^d$. Specifically, we resort to the idea expressed at the end of Section 2.3.2, which is to project the 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 $h_n(p)$ and its singular vectors with non-zero singular values. It is then straightforward to 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 metric h_n 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 $\sqrt{\det(h_n)}$ to obtain $\triangle \mathrm{Vol} \simeq d\mathrm{Vol}$. Summing over all points of the desired set gives the appropriate estimator:

$$\hat{\mathrm{Vol}}(W) = \sum_{p \in W} \sqrt{\det(h_n(p))} \triangle x^1(p) \dots \triangle x^d(p)\,. \tag{2.28}$$

Table 2.2 reports the results of our comparison of the performance of $\hat{\mathrm{Vol}}(W)$, described in 2.28 , and a "naive" volume estimator that computes the area on the Voronoi tessellation once the manifold is projected onto the tangent plane. We find that $\hat{\mathrm{Vol}}(W)$ performs better for all embeddings, as well as for the original data. The latter is explained by the fact that when we project the set W onto the tangent plane $T_{f(p)}f(\mathcal{M})$, we induce a fair amount of distortion, and the naive estimator has no way of correcting for it.

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



(a) (b) (c)

Figure 2.9: (a) Manifold along with W, the computed area in black. (b) Diffusion Maps ($\lambda = 1$) embedding with embedding metric h. (c) A locally isometric coordinate chart constructed from the Diffusion Maps 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	$2.63 \pm 0.10^{\dagger}$	2.70 ± 0.10	2.90%
Isomap	$6.53 \pm 0.34^{\dagger}$	2.74 ± 0.12	3.80%
LTSA	8.52e-4 $\pm$ 2.49e-4	2.70 ± 0.10	2.90 %
DM	6.70e-4 $\pm$ 0.464e-04†	2.62 ± 0.13	4.35 %

Table 2.2: Estimates of the volume of W on the hourglass depicted in Figure 2.9, based on 1000 sampled points. The experiment was repeated five times to obtain a variance for the estimators. †The naive area/volume estimator is obtained by projecting the manifold or embedding on $T_p\mathcal{M}$ and $T_{f(p)}f(\mathcal{M})$, respectively. This requires manually specifying the correct tangent planes, except for LTSA, which already estimates $T_{f(p)}f(\mathcal{M})$. Similarly to LTSA, $\hat{\text{Vol}}(W)$ is constructed so that the embedding is automatically projected on $T_{f(p)}f(\mathcal{M})$. Here, the true area is 2.658

2.7 Conclusion and Discussion

In this chapter, we have described a new method for preserving the important geometrical information in a data manifold embedded using any embedding algorithm. We showed that the Laplace-Beltrami operator can be used to augment *any* reasonable embedding so as to allow for the correct computation of geometric values of interest in the embedding's own coordinates.

Specifically, we showed that the Laplace-Beltrami operator allows us to recover a Riemannian manifold $(\mathcal{M}, g)$ from the data and express the Riemannian metric g in any desired coordinate system. We first described how to obtain the Riemannian metric from the mathematical, algorithmic and statistical points of view. Then, we proceeded to describe how, for any mapping produced by an existing manifold learning algorithm, we can estimate the Riemannian metric g in the new data coordinates, which makes the geometrical quantities like distances and angles of the mapped data (approximately) equal to their original values, in the raw data. We conducted several experiments to demonstrate the usefulness of our method.

Our work departs from the standard manifold learning paradigm. 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.

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 through the Laplace-Beltrami operator. 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 and are thus embedding-invariant in the limit of $n \to \infty$.

Of course, not everyone agrees that the original geometry is interesting in and of itself; sometimes, it should be discarded in favor of a new geometry that better highlights the features of the data that are important for a given task. For example, clustering algorithms stress the importance of the dissimilarity (distance) between different clusters regardless of what the original geometry dictates. This is in fact one of the arguments advanced by Nadler et al. [2006a] 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 Lee [2003] - could be of value and offer further insight. Indeed, while we explained in Section 2.6 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.[8] 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.

Having established the correspondence between the Laplace-Beltrami operator and the geometry of the embedding, we can now use the operator on any embedding of the data to find a metric that defines either how the embedding modifies the original manifold or how it differs from another embedding in its effect on the original

[8] 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.

manifold. In the next chapter, we move on to the realm of supervised and semi-supervised learning, and exploit the advantages of the Laplace-Beltrami operator for statistical analyses.

Chapter 3

GAUSSIAN PROCESSES ON MANIFOLDS: A SEMI-SUPERVISED LEARNING PERSPECTIVE

3.1 Unsupervised, Supervised and Semi-Supervised Learning and Tasks

Having discussed manifold geometry and its properties in some detail, we now aim to apply these concepts to common tasks, such as classification or regression. Specifically, we will address these tasks as semi-supervised learning problems.

For the sake of completeness and ease of reference, we begin by defining the concepts of unsupervised, supervised and semi-supervised learning (see Chapelle et al. [2006], Belkin et al. [2006])[1].

Unsupervised learning involves a set of n points, $\mathcal{P} = \{x_1, ..., x_n\}$, often assumed to be drawn i.i.d. from a distribution $\pi(x)$ on some sample space $\mathcal{X}$. The objective of unsupervised learning is to find interesting *structure* in the data - often some features of $\pi(x)$, such as its support in $\mathcal{X}$, clusters, quantiles, outliers, etc.

The LearnMetric algorithm in Chapter 2 is an example of unsupervised learning. It operates under the assumption that the support of $\pi(x)$ lies on or close to a manifold $\mathcal{M} \subset \mathbb{R}^r$ (i.e. $\mathcal{X} = \mathcal{M}$), and focuses on recovering the intrinsic geometry of $\mathcal{M}$.

Supervised learning, in contrast, considers training pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ drawn i.i.d. from some joint distribution $\pi(x, y)$ on sample space $\mathcal{X} \times \mathbb{R}$ or $\mathcal{X} \times \{1, \ldots, r\}$ and aims to learn a predictive map ψ such that $y_i = \eta(\psi(x_i))$ for some predetermined *link* function η. In the statistical literature, the variables $x = (x^1, \ldots, x^r)$ are generally known as the predictors or covariates, while y is known as the response. From a statistical perspective, the task of supervised learning aims to infer the conditional distribution $\pi(y|x)$ from a sample of the joint distribution of the predictors and response $\pi(x, y)$.

Semi-supervised learning lies somewhere in between: it is based on a sample

[1] We note a slight change in notation required to maintain consistency with the literature in this domain. In our previous discussions, we referred to our data points as p, whereas x designated the map $x : \mathcal{M} \to \mathbb{R}^d$. Although we can still think of x as a map, we will only consider it as a map evaluated at a fixed point $p \in \mathcal{M}$. That is to say, we will use x to mean $x = x(p)$ and x_i to mean $x_i = x(p_i)$.

$\{(x_1, y_1), \ldots, (x_n, y_n)\}$ from the joint distribution $\pi(x, y)$ and an additional sample $\mathcal{P}^* = \{x_1^*, ..., x_{n^*}^*\}$ from the marginal distribution $\pi(x)$. The objective is to exploit the additional information contained in $\pi(x)$ in order to improve upon learning $\pi(y|x)$. As explained by Belkin et al. [2006], this will only occur if an identifiable relation between $\pi(x)$ and $\pi(y|x)$ actually exists.

For example, a natural assumption that leads to semi-supervised learning is that if two points x_1, $x_2 \in \mathcal{X}$, where $\mathcal{X}$ is some metric space, are *close*, then the conditional distributions $\pi(y|x_1)$ and $\pi(y|x_2)$ should be similar.

A closely related concept is that of **transductive learning** (Chapelle et al. [2006]). In transductive learning, the goal is to predict the labels of the observed input points $\mathcal{P}^*$ or $\mathcal{P} \cup \mathcal{P}^*$, and there is no intention of generalizing the predictive map ψ between x and y to the entire space sample $\mathcal{X}$. The latter is the goal of **inductive learning**. When considering semi-supervised learning problems, it is natural to employ a transductive learning approach, as we will do here.

Semi-Supervised Learning and Manifolds

As mentioned above, we have so far considered an unsupervised learning problem. Specifically, we have been concerned with determining the intrinsic geometry of the support of $\pi(x)$ under the assumption that the support lies on or is close to a low-dimensional Riemannian manifold $(\mathcal{M}, g)$ with $\mathcal{M} \subset \mathbb{R}^r$ and g the induced metric of the embedding map. From a statistical point of view, this manifold hypothesis can be understood as the assumption that the predictors are locally collinear (Aswani et al. [2011]). The idea, then, is to exploit this local collinearity by performing any learning task with respect to the low-dimensional intrinsic geometry of the data rather than with respect to high-dimensional space. Typical applications are those where descriptive data is abundant and high-dimensional, but labelling is expensive or time-consuming. This yields a learning task that is rich in information on the collinearity of the predictors x, but poor in information on the response variable y.

When faced with a high-dimensional data set of the type described in the previous paragraph, it is natural to think of resorting to sparse type regularization, since sparse estimates can both help with variable selection and generate improved estimates. However, the fact that the data are collinear makes this an unfruitful and misguided exercise. Indeed, producing sparse estimates is difficult when the predictors lie on a manifold, and Lasso-type regularization, the Dantzig selector, and the RODEO (regularization of derivative expectation operator) simply cannot deal with such situations (Aswani et al. [2011]).

A more useful approach - and the one we employ here - is to first learn the manifold $\mathcal{M}$ formed by the predictors, and then perform the learning task with respect to $\mathcal{M}$. An elegant way of achieving this is to regularize the learning task with respect to $\mathcal{M}$; that is, define a kernel tailored to $\mathcal{M}$ and use it to regularize the learning task.

Kernel Regularization on Manifolds: Possible Approaches

The development of this approach follows a series of studies on kernel regularization on different types of spaces. Smola and Kondor [2003] began by introducing the idea that the input space for kernel regularization could be extended beyond Euclidean space. Smola and Kondor [2003] argued that as long as a kernel can be defined on the input space, all the kernel-based algorithms such as Support Vector Machines, Gaussian Processes, and Kernel PCA can be imported "wholesale", together with their error analysis and theoretical guarantees[2]. While these authors were mainly concerned with graphs and discrete spaces when making this claim, their idea could also be extended to another type of non-Euclidean space: manifolds.

Because every manifold can be embedded isometrically in Euclidean space by Nash's embedding theorem Nash [1956], we can always resort to the cordal distance between points and a radial basis function kernel to define a kernel regularizer on

[2]For an overview of kernels, Gaussian Processes (GP), and their interplay, see Section 3.2 below.

$\mathcal{M}$. However, this type of kernel is not particularly useful for two reasons: first, it is susceptible to the curse of dimensionality for high-dimensional data; and second, it discards any information contained in the geometry induced by the manifold and any structure found in the unlabelled points. Another option would be to use a radial basis function kernel, but replace the cordal distance with the geodesic along the manifold. While this may be an intuitively appealing solution, Gneiting [1998] has shown that we cannot expect, in general, that this approach will yield a valid symmetric positive semi-definite kernel, or in context of the GP, a valid covariance function, making it a somewhat risky choice.

We therefore find ourselves in need of a kernel that both takes advantage of information contained in the geometry of the data and that is well defined. [Belkin et al., 2006] provide us with such a kernel. They construct it implicitly via the Laplace-Beltrami operator on the intrinsic geometry of the data, used as a semi-norm of the form

$$< \psi, -\Delta_{\mathcal{M}}\psi >=< \nabla\psi, \nabla\psi > \ . \tag{3.1}$$

Here we have implicitly used Green's first identity and vanishing boundary conditions Evans [2010]. This semi-norm ends up penalizing large gradients and thereby produces a solution that is smooth with respect to $\mathcal{M}$. This idea is also exploited by Sindhwani et al. [2007], Zhu et al. [2003] who make similar use of $\Delta_{\mathcal{M}}$ to define a covariance function for GP.

The Laplace-Beltrami operator used as a kernel regularizer, though useful, is ill-defined in the asymptotic limit because the first order gradient is not a sufficient penalty in high dimensions (Nadler et al. [2009]). Specifically, this means that the solution will not be continuous if the intrinsic dimension of $\mathcal{M}$ is greater than one. However, this can be remedied by regularizing with respect to the iterated Laplace-Beltrami operator, as shown in Zhou and Belkin [2011], where the iteration parameter is used to control the smoothness of the solution.

This approach of using differential operators to construct kernels is formalized in

Steinke and Schölkopf [2008], where the authors clarify the correspondence between kernels and linear differential operators. Their contribution leads to the conclusion that searching for a good kernel regularizer can be tantamount to searching for the most sensible differential operator for a particular context.

One such operator of particular interest, which is also based on $\Delta_{\mathcal{M}}$, was developed by Lindgren et al. [2011]. Their paper considers a stochastic partial differential equation which extends the notion of the Matérn GF on $\mathbb{R}^d$ to any Riemannian manifold. This yields a flexible operator that is defined on the manifold and has a familiar interpretation when applied to Euclidean space. We will explore this model extensively in this chapter.

3.1.1 Alternatives to Laplacian-based Regressions

In the approaches we have discussed thus far, the selected kernel regularized the solution on the entire data set $\mathcal{P}$ at once, retaining as much information as possible from $\mathcal{M}$. An alternative approach, proposed by Aswani et al. [2011], is to compute the tangent plane at every observed point x_i and perform a local regression that penalizes the components that are perpendicular to the tangent plane.

The Aswani et al. [2011] approach presents several challenges.

The first challenge is that it requires that the dimension of the data manifold be known. However, in most cases, it is not known, and it can be difficult to estimate, though some sources provide methods for doing so (Levina and Bickel [2005], Hein and Audibert [2005]). Comparatively, knowing the exact dimension of the data manifold is less central to the Laplacian based regularization approach discussed earlier.

The second challenge is that the approach is very computationally intensive since it requires that the dual to the tangent plane be estimated at each point of interest so as to penalize regressors in that space. In their own experiment, the authors find themselves needing to scale down the resolution of the images they are working with: they scaled down an ambient space of $64 \times 64 = 4096$ to $7 \times 7 = 49$. In

other words, the technique requires performing non-linear dimensionality reduction as a pre-processing step, which we have established is not guaranteed to preserve the geometry of the data.

The idea of using the various definitions of derivatives on manifolds has led Lin et al. [2011] to consider regularizing the *second order smoothness* of the predictor function so that the solution can be as close to linear as possible. In contrast to the Laplacian-based approaches, which mostly favor constant solutions by virtue of the gradient penalty, the regularizer proposed by Lin et al. [2011] favors linearity by requiring that the gradient of the solution be as close as possible to a parallel vector field on $\mathcal{M}$. Being penalized to be close to linear, it is reasoned that the solution will be better suited for picking up trends with relative ease. In turn, this suggests that the Laplacian-based regularizers will be better suited for classification tasks whereas the Lin et al. [2011] approach is better suited for regression tasks.

3.2 Gaussian Processes and Kernel Regularization

3.2.1 Gaussian Processes

Rasmussen and Williams [2006] and Stein [1999] provide an excellent introduction to Gaussian processes, and we draw heavily on their work to cover the concepts required to fully discuss our contributions in this Chapter.

The term *Gaussian process* usually refers to a stochastic process which is jointly Gaussian; when the input space of the process is more than one-dimensional, it is referred to as a *Gaussian field*. However, in the the machine learning literature, like Rasmussen and Williams [2006] for example, Gaussian processes also refer to non-parametric models where a Gaussian field prior is defined over function values (also known as *Kriging* in the statistics literature).

To avoid confusion between the Gaussian process as a statistical model and Gaussian processes as stochastic processes, we will refer to the non-parametric model as Gaussian process (GP) and the stochastic process used in defining the model's prior as Gaussian field (GF).

We start by defining a GF over an input space $\mathcal{X}$:

Definition 16 (Gaussian Field) *Definition: A Gaussian field over $\mathcal{X}$ is a collection of random variables $\psi(x)|x \in \mathcal{X}$ indexed by the points in $\mathcal{X}$ such that for any finite sample $\mathcal{P} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$, the random vector $\mathbf{\Psi}(\mathcal{P}) = (\psi(x_1), \ldots, \psi(x_n))^t$ is jointly Gaussian*[3].

We write $\psi \sim \mathcal{GF}(m, k)$ to mean that ψ is a GF with mean function

$$E\left[\psi(x)\right] = m(x) \tag{3.2}$$

[3]Note that one can extend GF to irregular processes, such as white noise, where the field's joint distribution on a finite set is no longer well defined. This is done by studying the joint distribution of any finite number of linear functionals of ψ rather than ψ restricted to $\mathcal{P}$. See Lindgren et al. [2011] for details.

and covariance function

$$cov(\psi(x), \psi(z)) = k(x, z)\,. \tag{3.3}$$

The GF is completely determined by the mean function $m(x)$ and covariance function $k(x, z)$ in that for any finite sample $\mathcal{P} = \{x_1, \ldots, x_n\}$, we have $\mathbf{\Psi} \sim \mathcal{N}(\mathbf{m}, K)$. By this, we mean that $\mathbf{\Psi}$ is Gaussian with mean $\mathbf{m} = (m(x_1), \ldots, m(x_n))^t$ and covariance matrix $K_{i,j} = k(x_i, x_j)$. Specifically, $\mathbf{\Psi}$ has multivariate normal density given by

$$p(\mathbf{\Psi}|\mathbf{m}, K) = \frac{1}{(2\pi)^{n/2}\,|K|^{1/2}} \exp\left(-\frac{1}{2}\,(\mathbf{\Psi} - \mathbf{m})^t\,K^{-1}\,(\mathbf{\Psi} - \mathbf{m})\right). \tag{3.4}$$

However, to guarantee that the stochastic process $\mathcal{GF}(m, k)$ exists, we must impose a restriction on the covariance function $k(x, z)$: it must be symmetric semi-positive definite.

Definition 17 (Symmetric Semi-Positive Definite Function) *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, is said to be symmetric semi-positive definite if it is symmetric, $k(x, z) = k(z, x)$, and if for any finite sample $\mathcal{P} = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ and real numbers $c_1, \ldots, c_n \subset \mathbb{R}$, k satisfies*

$$\sum_{i,j}^{n} c_i c_j k(x_i, x_j) \geq 0 \tag{3.5}$$

In fact, this is a necessary and sufficient condition for $\mathcal{GF}(m, k)$ to exist (Stein [1999]).

Gaussian Process Regression

The GF definition in 16 induces a measure on the space of functions on $\mathcal{X}$, but we are not so much interested in sampling a function from $\mathcal{GF}(m, k)$ as in predicting the value of a function ψ given observed values $\{(x_1, \psi_1), \ldots, (x_n, \psi_n)\}$, where $\mathcal{P} = \{x_1, \ldots, x_n\}$ is known as the **training set**. When one assumes that

$$\psi \sim \mathcal{GF}(m, k_\Theta)\,, \tag{3.6}$$

i.e. that the generative process for the target ψ is a GF, then this problem is known as **Gaussian Process Regression** (GPR). In expressing GPR as in (3.6), we introduce

hyperparameters Θ for the covariance function k_Θ. We employ the convention of referring to Θ as *hyperparameters* since they are the parameters of the GF prior on the parameters of the GP model.

Note that we have taken the function-space view of GP, as it offers a natural framework in which to study how stochastic partial differential equations lead to useful GPs in Section 3.3 below. In the function-space view of GP, the GF is interpreted as a prior on functions rather than a prior on the parameters of the GP. This means that we will never explicitly need to work with the parameters of the GP model, but only with its hyperparameters Θ. However, there exists an equivalent interpretation of GP - known as the weight-space or feature-space view - which explicitly considers the GF as a prior on the parameters of the GP. We direct the interested reader to Rasmussen and Williams [2006] for an extensive discussion of the subject.

Having introduced hyperparameters for the covariance function k_Θ in (3.6), we now turn to clarifying why we did not attribute hyperparameters to the mean function m. In general, $m(x)$ will also have hyperparameters, for example when introducing linear terms of the form $m_\alpha(x) = \sum_i \alpha_i f_i(x)$ where the $f_i(x)$'s are functions in the null space of k_Θ and the α_i's are the hyperparameters of $\mathbf{m}$. For the purposes of our discussion in this Chapter, however, we will take the common approach of only considering GF priors with zero mean function ($m(x) \equiv 0$).

Since the definition of GF is in terms of a finite sample, the natural thing to do is to determine the distribution of ψ for any finite set of points, known as the **test set** $\mathcal{P}^* = \{x_1^*, \ldots, x_{n^*}^*\} \subset \mathcal{X}$, given the observed data. From the definition of GF, we know that the joint distribution of $(\mathbf{\Psi}(\mathcal{P}), \mathbf{\Psi}(\mathcal{P}^*))^t$ is given by

$$\begin{pmatrix} \mathbf{\Psi}(\mathcal{P}) \\ \mathbf{\Psi}(\mathcal{P}^*) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(\mathcal{P},\mathcal{P}) & K(\mathcal{P},\mathcal{P}^*) \\ K(\mathcal{P}^*,\mathcal{P}) & K(\mathcal{P}^*,\mathcal{P}^*) \end{pmatrix} \right),$$

where $K(\mathcal{P},\mathcal{P})_{i,j} = k_\Theta(x_i, x_j)$ and similarly for $K(\mathcal{P}^*,\mathcal{P})$, $K(\mathcal{P},\mathcal{P}^*)$, and $K(\mathcal{P}^*,\mathcal{P}^*)$. Obtaining the posterior distribution of $\mathbf{\Psi}(\mathcal{P}^*)$ given the observed points is easily computed in the conditional form of the multivariate normal distribution:

$$p\left(\mathbf{\Psi}(\mathcal{P}^*)|\mathbf{\Psi}(\mathcal{P}),\mathcal{P}^*,\mathcal{P},\Theta\right) \sim \mathcal{N}(K(\mathcal{P}^*,\mathcal{P})K(\mathcal{P},\mathcal{P})^{-1}\mathbf{\Psi}(\mathcal{P}), K(\mathcal{P}^*,\mathcal{P}^*) - K(\mathcal{P}^*,\mathcal{P})K(\mathcal{P},\mathcal{P})^{-1}K(\mathcal{P},\mathcal{P}^*))\,. \quad (3.7)$$

This is a rather idealized situation where we assume that we actually observe ψ on $\mathcal{P}$. A more sensible model is to assume that we observe a noisy version of $\mathbf{\Psi}(\mathcal{P})$. That is, we model the observed $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, where $\mathbf{y}=(y_1,\ldots,y_n)^t$, as

$$\mathbf{y} = \mathbf{\Psi}(\mathcal{P}) + \eta \quad (3.8)$$

where $\eta \sim \mathcal{N}(0,\sigma^2 I_n)$ represents independent identically distributed Gaussian noise which is also independent of ψ. Under this noisy obervation model, the distribution of Y is given by

$$p(\mathbf{y}|\mathbf{\Psi}(\mathcal{P}),\sigma^2) \sim \mathcal{N}(\mathbf{\Psi}(\mathcal{P}),\sigma^2 I_n)\,.$$

Since we now aim to find the distribution of $\mathbf{\Psi}(\mathcal{P}^*)$ given the noisy observations, we need to find the posterior $p\left(\mathbf{\Psi}(\mathcal{P}^*)|\mathbf{y},\mathcal{P}^*,\mathcal{P},\Theta,\sigma^2\right)$. The joint distribution between $\mathbf{\Psi}(\mathcal{P}^*)$ and the observations $\mathbf{y}$ now takes the form

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{\Psi}(\mathcal{P}^*) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(\mathcal{P},\mathcal{P})+\sigma^2 I_n & K(\mathcal{P},\mathcal{P}^*) \\ K(\mathcal{P}^*,\mathcal{P}) & K(\mathcal{P}^*,\mathcal{P}^*) \end{pmatrix}\right).$$

From the conditional multivariate normal distribution we now have the posterior

$$p\left(\mathbf{\Psi}(\mathcal{P}^*)|\mathbf{y},\mathcal{P}^*,\mathcal{P},\Theta,\sigma^2\right) \sim \mathcal{N}(K(\mathcal{P}^*,\mathcal{P})\left(K(\mathcal{P},\mathcal{P})+\sigma^2 I_n\right)^{-1}\mathbf{y}, K(\mathcal{P}^*,\mathcal{P}^*) - K(\mathcal{P}^*,\mathcal{P})\left(K(\mathcal{P},\mathcal{P})+\sigma^2 I_n\right)^{-1}K(\mathcal{P},\mathcal{P}^*)) \quad (3.9)$$

Using Equation 3.7, we can compute the distribution of ψ on any points $x^* \in \mathcal{X}$ from observations $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, and in this very limited sense, we have solved the GPR problem. Indeed, we still need to specify a form for the covariance function k_Θ and choose a method for estimating the proper hyperparameters; we

discuss the latter of these problems in more detail in Section 3.2.3 below. As for selecting k_Θ, this is a problem-specific choice that is ideally guided by prior knowledge about the problem and an understanding of the available covariance functions. There are many well-documented covariance functions to select from when solving problems in Euclidean space ($\mathcal{X} = \mathbb{R}^r$), but this is sadly not the case when the input space is a general Riemannian manifold ($\mathcal{X} = \mathcal{M}$), especially when the actual geometry of the manifold is unknown. In fact, defining a covariance function over a manifold of unknown geometry from a finite sample that is both interpretable and flexible is the main objective of this chapter and will be discussed in detail in Sections 3.3 and 3.4 below.

Although we delay the discussion of how to define covariance functions on Riemannian manifolds, it is worth highlighting the fact that the choice of k_Θ also determines whether a GP falls under the inductive learning paradigm or under the transductive learning paradigm. From Equation (3.9) we observe that the posterior distribution of $\mathbf{\Psi}(\mathcal{P}^*)$ depends on two things: the labelled points $\mathbf{y}$ on $\mathcal{P}$ and the values of the covariance function k_Θ on $\mathcal{P} \cup \mathcal{P}^*$, i.e. $K(\mathcal{P} \cup \mathcal{P}^*, \mathcal{P} \cup \mathcal{P}^*)$. This means that if k_Θ is known to us *a priori* on the whole of $\mathcal{X} \times \mathcal{X}$, then (3.9) is a form of inductive learning. Indeed, if the form of covariance k_Θ is independent of the observed training set $\mathcal{P}$ and test set $\mathcal{P}^*$, then one can readily extend the posterior of $\mathbf{\Psi}$ to any point of $\mathcal{X}$. Meanwhile, if the form of k_Θ is only known or, as will be our case, *estimated* on $\mathcal{P} \times \mathcal{P}^*$, then (3.9) will be a form of transductive learning.

The fact that the boundary between inductive and transductive learning is determined by our knowledge of the covariance function (or lack thereof) is not specific to GP and applies also to kernel regularization discussed in Section 3.2.2 below. In general, this boundary applies to any learning approach that relies on a symmetric semi-positive definite function to induce structure on the input space $\mathcal{X}$.

Gaussian Process Classification

Another common problem is that of classification, in which one hopes to predict the class of each point $x \in \mathcal{X}$ rather than a real-value function on $\mathcal{X}$. In our discussion, we limit ourselves to the two-class problem with $y \in \{-1, 1\}$[4]

The problem here is that the Gaussian assumption is not particularly natural for a function that only takes two values. In defining a Gaussian Process Classifier (GPC), we can nonetheless appeal to the main idea behind GPR, which is to use a GF as a prior on the linear predictor of a generalized linear model (GLM). That is, given a GLM with predictor $\psi(x)$ and mean function $E[Y|\psi(x)] = \gamma(\psi(x))$, we impose a GF prior on $\psi \sim \mathcal{GF}(0, k_\Theta)$ just as for GPR.

For a two-class classifier, the natural choice is to use ψ as the predictor to a logistic regression for which the mean function is given by

$$\gamma(\psi) = \frac{1}{1 + \exp(-\psi)}\,.$$

Just as the model for GPR was given by 3.8, we now have the classification model

$$\begin{aligned} p(y = 1|\psi(x)) &= \gamma(\psi(x))\,, \text{ and} \\ p(y = -1|\psi(x)) &= 1 - p(y = 1|\psi(x)). \end{aligned}$$

The goal now is to find the posterior distribution over the two classes at a point x^* given observations on $\mathcal{P}$. Formally, we are looking for $p(y^* = 1|x^*\mathcal{P}, \mathbf{y})$. As the conditional multivariate distribution cannot be of assistance here, we follow the usual Bayesian approach of conditioning on the latent function ψ and integrating it out:

$$\begin{aligned} p(y^* = 1|x^*, \mathcal{P}, \mathbf{y}, \Theta) &= \int p(y^* = 1|\psi(x^*), x^*, \mathcal{P}, \mathbf{y}) p(\psi(x^*)|x^*, \mathcal{P}, \mathbf{y}, \Theta) d\psi(x^*) \\ &= \int \gamma(\psi(x^*)) p(\psi(x^*)|x^*, \mathcal{P}, \mathbf{y}, \Theta) d\psi(x^*) \\ &= E[\gamma(\psi(x^*))|x^*, \mathcal{P}, \mathbf{y}, \Theta]\,, \end{aligned} \tag{3.10}$$

[4]For a multiclass problem with c classes, we can always construct c independent two-class classifiers such that each classifier determines which elements are in its class and which are not.

where the posterior distribution of $\psi(x^*)$ is given by

$$\begin{aligned} p(\psi(x^*)|x^*,\mathcal{P},\mathbf{y},\Theta) &= \int p(\psi(x^*)|x^*,\mathcal{P},\mathbf{\Psi}(\mathcal{P}),\Theta)p(\mathbf{\Psi}(\mathcal{P})|\mathcal{P},\mathbf{y},\Theta)d\mathbf{\Psi}(\mathcal{P}) \qquad (3.11)\\ &= \int p(\psi(x^*)|x^*,\mathcal{P},\mathbf{\Psi}(\mathcal{P}),\Theta)\frac{p(\mathbf{y}|\mathbf{\Psi}(\mathcal{P}))p(\mathbf{\Psi}(\mathcal{P})|\mathcal{P},\Theta)}{p(\mathbf{y}|\mathcal{P},\Theta)}d\mathbf{\Psi}(\mathcal{P})\,. \end{aligned}$$

In general, there is no closed form for these integrals and one must resort to Monte Carlo sampling (which can be computationally intensive) or use approximations.

Popular approximations, such as the Laplace or the Expectation Propagation methods, are based on approximating $p(\mathbf{\Psi}(\mathcal{P})|\mathcal{P},\mathbf{y})$ as a normal distribution centered at

$$\hat{\mathbf{\Psi}}(\mathcal{P}) = \text{argmax}_{\,\mathbf{\Psi}(\mathcal{P})} p(\mathbf{\Psi}(\mathcal{P})|\mathcal{P},\mathbf{y},\Theta)\,. \qquad (3.12)$$

Under this Gaussian assumption, it is straightforward to compute the posterior in Equation (3.11). The problem is then reduced to computing integral 3.10, which is one-dimensional, and therefore much simpler to compute numerically.

It is also common to approximate the problem further by replacing the posterior $p(\psi(x^*)|x^*,\mathcal{P},\mathbf{y})$ by a point mass at $\hat{\mathbf{\Psi}}(\mathcal{P})$ rather than using a Gaussian approximation. This makes 3.10 trivial to compute and amounts to using $\gamma(E[\psi(x^*)|x^*,\mathcal{P},\mathbf{y},\Theta])$ as the prediction instead of $E[\gamma(\psi(x^*))|x^*,\mathcal{P},\mathbf{y},\Theta]$. This approximation is known as the maximum *a posteriori* (MAP) predictor and is closely linked to kernel regularization, which we will discuss in the next section. It is worth pointing out that even though the MAP predictor retains a probabilistic interpretation, replacing $p(\psi(x^*)|x^*,\mathcal{P},\mathbf{y},\Theta)$ by a point mass means that we are no longer in a position to assess the uncertainty of the prediction.

3.2.2 Kernel Regularization

Reproducing Kernel Hilbert Space

Before discussing how kernel regularization is used in statistical learning theory and how it relates to GP discussed in Section 3.2.1, we first need to introduce the theo-

retical framework behind this approach: reproducing kernel Hilbert space (RKHS).

Definition 18 (Reproducing Kernel Hilbert Space) *Let $\mathcal{H}$ be a Hilbert space of real functions f defined on an input space $\mathcal{X}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with inner product $< \cdot, \cdot >_{\mathcal{H}}$ and associated norm $||f||^2_{\mathcal{H}} =< f, f >_{\mathcal{H}}$ if there exists a symmetric semi-definite kernel (definition 17) $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which satisfies:*

1. *For every $x \in \mathcal{X}$, $k(x, z)$ as a function of y is in $\mathcal{H}$.*

2. *k has the reproducing property $< k(x, \cdot), f(\cdot) >= f(x)$ for any $f \in \mathcal{H}$.*

In practice, a Hilbert space $\mathcal{H}$ will be a RKHS if, for every $x \in \mathcal{X}$, the linear functional $h_x : \mathcal{H} \to \mathbb{R}$ such that $\delta_x(f) = f(x)$ is continuous (Belkin et al. [2006]). Indeed, this property guarantees, via the Riez representation theorem, that there exists a unique $h_x \in \mathcal{H}$ such that $\delta_x(f) =< h_x, f >_{\mathcal{H}}$. One can then construct the kernel k via

$$k(x, z) =< h_x, h_y >_{\mathcal{H}}, \tag{3.13}$$

which will necessarily be symmetric semi-positive definite and satisfy conditions 1 and 2 from Definition 18 (Belkin et al. [2006]).

This construction of the kernel hints at a key fact of RKHS: the kernel k is uniquely determined by the RKHS and, conversely, the RKHS is uniquely determined by the kernel k. This is formalized by the Moore-Aronszajn theorem:

Theorem 19 (Moore-Aronszajn) *For every symmetric semi-positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists a RKHS with kernel k, and vice versa (Aronszajn [1950]).*

Given the equivalence between RKHS $\mathcal{H}$ and kernel k, we will henceforth explicitly identify a RKHS with its kernel k as $\mathcal{H}_k$.

Equation (3.13) shows that the inner product $< \cdot, \cdot >_{\mathcal{H}_k}$ characterizes the kernel. However, to fully appreciate Theorem 19, we need to clarify in what sense the kernel characterizes the RKHS. There are two ways to go about doing this: Mercer's Theorem and the Representer Theorem. We will use Mercer's Theorem in our discussion below, as it provides the most insight into RKHS.

We first note that if $\mathcal{X} \subset \mathbb{R}^r$ is closed and ν is a measure on $\mathcal{X}$ with k a symmetric semi-definite kernel on $\mathcal{X} \times \mathcal{X}$ satisfying

$$\int_{\mathcal{X}} \int_{\mathcal{X}} |k(x,z)|^2 d\nu(x) d\nu(y) < \infty \,, \tag{3.14}$$

then the induced operator

$$T_k(f)(x) = \int_{\mathcal{X}} k(x,z) f(y) d\nu(y) \tag{3.15}$$

is self-adjoint, positive, and compact. By the spectral theory of compact operators, this means that the eigenfunctions $\{\phi_i\}_{i=1}^{\infty}$ of T_k form an orthonormal basis for $L^2_\nu(\mathcal{X})$ and that the corresponding eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$ have finite multiplicity, are non-negative, and are square summable $\sum_{i=1}^{\infty} \lambda_i^2 < \infty$ (Minh et al. [2006], Belkin et al. [2006]).

If we only consider measures that are strictly positive on $\mathcal{X}$, we have:

Theorem 20 (Mercer) *Let $\mathcal{X} \subset \mathbb{R}^r$ be closed and ν a strictly positive Borel measure on $\mathcal{X}$. Let k be a symmetric semi-definite kernel on $\mathcal{X} \times \mathcal{X}$ satisfying* (3.14)*, then*

$$k(x,z) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(y) \tag{3.16}$$

where the series converges absolutely for any point $(x,z) \in \mathcal{X} \times \mathcal{X}$ and uniformly on any compact subset of $\mathcal{X}$ (Minh et al. [2006]).

We are now in a position to determine how $< \cdot, \cdot >_{\mathcal{H}_k}$ relates to the kernel function k. Working from the two alternative constructions of k, (3.13) and (3.16):

$$k(x,z) = < k(x,\cdot), k(y,\cdot) >_{\mathcal{H}_k} \tag{3.17}$$

$$\sum_{k=1}^{\infty} \lambda_k \phi_k(x)\phi_k(y) = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \lambda_i\lambda_j\phi_i(x)\phi_j(y) < \phi_i, \phi_j >_{\mathcal{H}_k} . \tag{3.18}$$

Since the eigenfunctions $\{\phi_i\}_{i=1}^{\infty}$ form an orthonormal basis for $L^2_\nu(\mathcal{X})$, we conclude that

$$< \phi_i, \phi_j >_{\mathcal{H}_k} = \begin{cases} \frac{1}{\lambda_i} & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

Thus, in general, for any two functions $f, q \in L^2_\nu(\mathcal{X})$ with decomposition $f = \sum_{i=1}^{\infty} f_i\phi_i(x)$ and $q = \sum_{i=1}^{\infty} q_i\phi_i(x)$, their inner product in $\mathcal{H}_k$ is given by

$$< f, q >_{\mathcal{H}_k} = \sum_{i=1}^{\infty} \frac{f_i q_i}{\lambda_i} . \tag{3.20}$$

This also means that $f \in \mathcal{H}_k$ if and only if $< f, f >_{\mathcal{H}_k} = \sum_{i=1}^{\infty} \frac{f_i^2}{\lambda_i} < \infty$ (in the same way that $f \in L^2_\nu(\mathcal{X})$ if and only if $\sum_{i=1}^{\infty} f_i^2 < \infty$).

Equation (3.13) expresses the inner product $< f, q >_{\mathcal{H}_k}$ in terms of the eigendecomposition of T_k. To appreciate this decomposition, it is worth considering the finite vector space equivalent of (3.13). Specifically, (3.13) means that for a finite RKHS with kernel matrix K, the inner product takes the form $< f, q >_{\mathcal{H}_k} = f^t K^{-1} q$. Note that $< f, f >_{\mathcal{H}_k} = f^t K^{-1} f$ defines the same quadratic form as the one used in the multivariate normal distribution (3.4) with covariance matrix K. This connection will be further explored below, but it already suggests that the kernel of a RKHS will play a role similar to that of the covariance function of a GP.

Finally, it is also worth noting that the inner product $< f, q >_{\mathcal{H}_k}$ is independent of the measure ν in Theorem 20 and that the norm (3.20) is solely a property of the kernel (Rasmussen and Williams [2006]).

Empirical Risk Minimization

Once again, many authors have proposed excellent summaries of concepts we need to cover, but we draw on Bousquet et al. [2004], Rasmussen and Williams [2006] in particular in our exposition below.

The standard problem in statistical learning is different from that of GP described in Section 3.2.1 above, although the two will be shown to be similar. Instead of specifying a generative model for the target y, the idea is to specify a loss function $Q(x, y, \psi)$ that measures the loss incurred in predicting ψ for a given value of y. The choice of loss function Q is determined by the problem one is trying to solve. For example, the natural choice for a regression problem is the squared error loss $Q(x, y, \psi) = ||y - \psi(x)||^2$, while for a classification problem the natural choice is the 0-1 loss $Q(x, y, \psi) = \mathbf{1}[y = \text{sign}(\psi(x)]$. However, the choice of loss function is not always determined solely by the type of problem addressed; other considerations can be relevant as well. For example, in the case of classification, Hinge's loss $Q(x, y, \psi) = \max(0, 1 - y\psi(x))$ is also very commonly used, as it leads to a convex problem that is easier to optimize.

In theory, one would like to select ψ so that it minimizes the expected loss $E[Q(y, \psi)]$ - commonly referred to as risk. The expectation $E[Q(x, y, \psi)]$ is taken over the joint distribution of the target and input space $p(x, y)$. Since the joint distribution $p(x, y)$ is unknown, it is not possible to compute $E[Q(x, y, \psi)]$ explicitly. Rather, the common approach is to assume that observations $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ constitute a random sample from $p(x, y)$ and define the **empirical risk** as follows:

$$R_n(\psi) = \frac{1}{n} \sum_{i=1}^{n} Q(x_i, y_i, \psi(x_i)) \,. \tag{3.21}$$

The empirical risk (3.21) can then be minimized over $\mathcal{F}$, the space of all functions on $\mathcal{X}$:

$$\hat{\psi}_n = \underset{\psi \in \mathcal{F}}{\text{argmin}} R_n(\psi) \,. \tag{3.22}$$

Problem (3.22) is ill-posed however, since minimizing 3.21 over all possible functions on $\mathcal{X}$ will generally lead to a situation where ψ_n predicts y perfectly on the training set $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ but fails to generalize to test set $\mathcal{P}^*$. In other words, minimizing 3.21 without additional constraints will lead to overfitting.

The two approaches used to prevent overfitting either a) explicitly limit the space of functions $\mathcal{F}$ over which to find (3.22), or b) add a term to (3.22) that penalizes overly complex functions in $\mathcal{F}$. Kernel regularization takes the latter approach. It replaces $\mathcal{F}$ by a RKHS $\mathcal{H}_k$ that is deemed appropriate for the problem, and penalizes functions with large values of the square norm $||\cdot||^2_{\mathcal{H}_k}$.

Specifically, the problem now takes the form of

$$\hat{\psi}_n \quad = \quad \underset{\psi \in \mathcal{H}_k}{\operatorname{argmin}} \left(R_n(\psi) + \mu^2 \, ||\psi||^2_{\mathcal{H}_k} \right) , \tag{3.23}$$

where μ is referred to as the "regularization" parameter. Just as in GP, we might choose to have some hyperparameters Θ on the kernel k_Θ to add flexibility to the model. In that sense, μ can be thought of as one of these hyperparameters.

Even with the added penalty, the task of solving (3.23) might still seem quite daunting as the RKHS space $\mathcal{H}_k$ is often very large. The Representer Theorem provides an elegant tool to address this problem.

Theorem 21 (Representer Theorem) *Let $\mathcal{H}_k$ be a RKHS on input set $\mathcal{X}$ with kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Given a training sample $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ and empirical risk function $R_n : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$[5] and a monotonically increasing function $c : [0, \infty) \to \mathbb{R}$, then any $\tilde{\psi} \in \mathcal{H}_k$ satisfying*

$$\tilde{\psi} = \underset{\psi \in \mathcal{H}_k}{argmin} \left(R_n(\psi) + c \left(||\psi||^2_{\mathcal{H}_k} \right) \right) \tag{3.24}$$

admits a representation of the form

$$\tilde{\psi}(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i) \, , \tag{3.25}$$

[5]Note that this is a more general definition of empirical risk than the one we offered in (3.21) as it is not necessarily expressed as a sum over the training pairs.

where $\{\alpha_1, \ldots, \alpha_n\} \in \mathbb{R}$ *(Schölkopf et al. [2001]).*

In other words, the Representer Theorem reduces the complexity of the minimization problem (3.23) from a search over a potentially infinite space $\mathcal{H}_k$ to search over a finite set of parameters $\{\alpha_1, \ldots, \alpha_n\}$. Equation (3.25) then gives us the predicted solution $\hat{\psi}_n$ for the whole of $\mathcal{X}$, assuming we know the form of the kernel k on the whole $\mathcal{X} \times \mathcal{X}$. This highlights the fact that the boundary between inductive learning and transductive learning is determined by whether k is known to us *a priori* on the whole of $\mathcal{X} \times \mathcal{X}$ or whether it depends on the observed data $\mathcal{P} \cup \mathcal{P}^*$, as first discussed in Section 3.2.1.

Relation to GP

The next element of interest is the link between kernel regularization and GP. The framework that links kernel regularization - or, more accurately, RKHS - and GP has been formalized in recent years (Hein and Bousquet [2004], Steinke and Schölkopf [2008]). Not surprisingly, it centers on symmetric semi-definite functions to induce structure on $\mathcal{X}$. Our aim here is to present a more intuitive explanation of the connection; specifically, we are interested in how the solution to 3.23 and the predictions from GP ((3.9) or (3.10)) are related.

To discuss how Equation (3.23) relates to GP, we consider the case where the loss $Q(x, y, \psi)$ can be normalized and hence has an alternative interpretation as a negative log-likelihood $p(y|\psi, x)$. This is important to ensure that the solution to the regularized empirical risk minimization problem (3.23) has a probabilistic interpretation. If that is the case, one can take the exponential of the regularized empirical risk and provide an interpretation via Bayes' theorem:

$$\begin{aligned} p(\mathbf{\Psi}(\mathcal{P})|\mathcal{P}, \mathbf{y}) &\propto p(\mathbf{y}|\mathbf{\Psi}(\mathcal{P}), \mathcal{P}, \Theta) p(\mathbf{\Psi}(\mathcal{P})|\mathcal{P}, \Theta) \\ &\propto \exp(-nR_n(\mathbf{\Psi}(\mathcal{P}))) \cdot \exp(-\mu^2 \mathbf{\Psi}(\mathcal{P})^t K(\mathcal{P}, \mathcal{P})^{-1} \mathbf{\Psi}(\mathcal{P})^t) \,. \end{aligned} \quad (3.26)$$

As expected, the empirical risk serves as *evidence* via the likelihood $p(y|\psi, x)$, while the regularizer $||\psi||_{\mathcal{H}_k} = \mathbf{\Psi}(\mathcal{P})^t K(\mathcal{P}, \mathcal{P})^{-\mathbf{1}} \mathbf{\Psi}(\mathcal{P})^t$ takes the form of a GF prior on ψ. In other words, we can interpret kernel regularization as a penalty on the complexity of ψ or as a GF prior on ψ with mean $m \equiv 0$, covariance function k_Θ and model $p(y|\psi, x) \propto \exp(-Q(x, y, \psi))$.

Meanwhile, the penalized empirical risk minimizer $\hat{\psi}_n$ in (3.23) is now the maximizer of (3.26), so that $\hat{\psi}_n$ is really the MAP estimate $\hat{\mathbf{\Psi}}(\mathcal{P})$ from (3.12) used to approximate (3.10). Note that this does not mean that the penalized empirical risk minimizer $\hat{\psi}_n$ will agree with the test set posterior $\hat{\mathbf{\Psi}}(\mathcal{P}^*)$ for any general problem. Indeed, (3.11) is not generally equivalent to the solution (3.25) prescribed by the Representer Theorem. One notable exception is GPR, for which the expected value of the posterior on the test set (3.9) will be equal to the solution of the penalized empirical risk with mean squared error loss (Rasmussen and Williams [2006]).

3.2.3 Estimating the Hyperparameter

One key element we have yet to address in either the GP or the kernel regularization framework is the selection of the covariance function or regularization kernel k_Θ. As we have stated before, the choice of k_Θ is usually inspired by the the context of the problem and the desired properties of k_Θ. Even when a reasonable choice for k_Θ has been made however, one still needs to determine the more specific properties of k_Θ, e.g. length scale and smoothness, encoded in the hyperparameters Θ.

Bayesian Approach

In a purely Bayesian framework, the natural approach is not to estimate the hyperparameter of the prior, but rather to add another inference layer in the form of a prior $p(\Theta)$ over the hyperparameters to express our knowledge (or lack thereof) regarding the hyperparameter's values. For a general GP, this means that the posterior

distribution on $\mathbf{y}$ is given by

$$p(y^*|x^*, \mathcal{P}, \mathbf{y}) = \int p(y^*|x^*, \mathcal{P}, \mathbf{y}, \Theta)p(\Theta)d\Theta\,, \tag{3.27}$$

and the same applies to $\mathbf{\Psi}(\mathcal{P}^*)$ for GPR when we are interested in the latent function ψ rather than $\mathbf{y}$. Evaluating $p(y^*|x^*, \mathcal{P}, \mathbf{y})$ means selecting an appropriate prior $p(\Theta)$ as well as computing the generally intractable intergral in 3.27. A popular approach in statistics is to use Markov chain Monte Carlo (MCMC) to obtain a sample from $p(y^*|x^*, \mathcal{P}, \mathbf{y})$. MCMC is computationally intensive, especially in the GP context, which requires inverting the kernel matrix $K(\mathcal{P}, \mathcal{P})$. This means that as n grows, the complexity of sampling will grow as $O(n^3)$.

Marginal Likelihood

An alternative to a purely Bayesian approach is to use the hyperparameters that best explain the observed data by maximizing the marginal likelihood

$$p(\mathbf{y}|\mathcal{P}, \Theta) = \int p(\mathbf{y}|\mathcal{P}, \mathbf{\Psi}(\mathcal{P}))p(\mathbf{\Psi}(\mathcal{P})|\Theta)d\mathbf{\Psi}(\mathcal{P})\,, \tag{3.28}$$

over Θ. We then get that the posterior of y^* given the observed data is $p(y^*|x^*, \mathcal{P}, \mathbf{y}, \hat{\Theta})$ where

$$\hat{\Theta} = \text{argmin}_{\Theta} p(\mathbf{y}|\mathcal{P}, \Theta)\,.$$

Although elegant and easy to compute, this approach should be taken with caution, however, since it may lead to overfitting (Rasmussen and Williams [2006]).

Cross-Validation

When working with the MAP estimate or empirical risk, however, we cannot compute the marginal likelihood. Instead we use an alternative approach to estimating the hyperparameters - cross-validation. Obviously, one could also use cross-validation to estimate parameters in a GP context even if we have access to the marginal likelihood

(3.28); and indeed, as mentioned in Rasmussen and Williams [2006], cross-validation and marginal likelihood maximization tend to perform equally well. However, cross-validation takes considerably longer, especially if the learning is done on several subsets, as we will explain below.

The idea of cross-validation to estimate hyperparameters, or more generally to test models, is to use a subset of the data available for learning, i.e. the labels $\mathbf{y}$ on the training set $\mathcal{P}$, and test how well the solution for a given set of hyperparamters generalizes to the remainder of $\mathcal{P}$. The assumption is that the error on the remaider of $\mathcal{P}$ provides a good proxy for the test error (also known as generalization error): the error of the solution on the test set $\mathcal{P}^*$.

At its simplest, this means that the training set would be split into two subsets, with the first half being used for training and the second for validation. Then, the roles of the two subsets would be reversed, and the final cross-validation error would be the average of the previous two operations. The natural generalization of this idea is to split the available data into more than two equally-sized subsets, for example 5, 10 or even n subsets. Then, one of these subsets is set aside for validation purposes while learning or training occurs on the remaining sets; this can be repeated as many times as there are subsets, each time using a different one of the subsets for validation. While the latter approach may reduce the variability of the estimates obtained, it is only practical if computation time is not a limitation. It is also subject to the risk of overfitting Ng [1997]. Nonetheless, cross-validation is quite popular because of its simplicity and it wide range of applicability. Indeed, it can be used for any model or loss function.

3.3 Differential Operators as Regularizers

The connection between GP and kernel regularization has already been discussed, but another revealing connection is the correspondence with linear differential operators (Steinke and Schölkopf [2008]). Specifically, given a differential linear operator $\mathcal{L}$, the stochastic partial differential equation (SPDE) on manifold $\mathcal{M}$

$$\mathcal{L}(\psi) \quad = \quad \xi \text{ in } \mathcal{M}\,, \tag{3.29}$$

with appropriate boundary conditions on $\partial\mathcal{M}$, and where ξ represents a Gaussian white noise source term, has solution

$$\psi \sim \mathcal{GF}\left(0, \mathcal{L}^{-1}(\mathcal{L}^t)^{-1}\right)\,. \tag{3.30}$$

Here, $\mathcal{L}^t$ is the adjoint of $\mathcal{L}$, and $\mathcal{L}^{-1}$, if it exists, is the inverse of the differential operator and in usually referred to as the Green function of problem (3.29).

This correspondence between differential operators and GF or kernels is particularly powerful because one can derive intuition from either the SPDE, the GF or the resulting RKHS. Of particular interest to us is the following SPDE:

$$\left(\kappa^2 - \Delta_{\mathbb{R}^d}\right)^{\alpha/2}\psi \quad = \quad \xi\,, \text{ in } \mathbb{R}^d \tag{3.31}$$

with $\alpha > d/2$ (Lindgren et al. [2011]), where d is the dimension of the input space $\mathbb{R}^r$.

One can show by Fourier analysis (Whittle [1954, 1963]) that the covariance function defined by this operator, just as in (3.30), is the Matérn covariance function

$$k(x,z) \quad = \quad \frac{\left(\kappa\,||x-z||\right)^{\nu} K_{\nu}\left(\kappa\,||x-z||\right)}{2^{\nu-1}\Gamma(\nu+d/2)\,(4\pi)^{d/2}\,\kappa^{2\nu}}\,, \tag{3.32}$$

where K_ν is the modified Bessel function of the second kind of order ν. In addition, the functional form of $k(x,z) = k(||x - mbz||)$ means that $1/\kappa$ controls the **length scale** of the GF. Meanwhile, $\alpha = \nu + d/2$ controls the **smoothness** of the GF ψ via (3.31).

What makes the interplay between the above SPDE and the Matérn GF so powerful is the flexibility (3.31) offers in terms of generalizing the notion of what constitutes a Matérn GF. This is the approach taken in Lindgren et al. [2011], where the authors extend the Matérn fields to manifolds, introduce non-stationary Matérn fields, and introduce oscillating Matérn fields, amongst other things. The most interesting idea for our purposes is the extension of Matérn fields to Riemannian manifolds via

$$\left(\kappa^2 - \Delta_{\mathcal{M}}\right)^{\alpha/2} \psi = \xi \,, \text{ in } \mathcal{M} \tag{3.33}$$

$$\partial_n \left(\kappa^2 - \Delta_{\mathcal{M}}\right)^{j} \psi = 0 \,, \text{ on } \partial\mathcal{M} \,, \text{ for } j = 1, \ldots, \lfloor (\alpha - 1)/2 \rfloor \tag{3.34}$$

where we have replaced the Laplacian $\Delta_{\mathbb{R}^d}$ in (3.31) with the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$, and we have added the boundary condition (3.34) where ∂_n is the directional derivative along the normal to the boundary $\partial\mathcal{M}$. Equations (3.33) and (3.34) give rise to the Matérn field

$$\psi \sim \mathcal{GF}\left(0, \left(\kappa^2 - \Delta_{\mathcal{M}}\right)^{-\alpha}\right), \text{ on } \mathcal{M}. \tag{3.35}$$

We are not imposing any constraints on α other than $\alpha > d/2$, which, as will be discussed below, is necessary to guarantee that the process ψ is continuous in the mean square[6]. For non-integer values of $\alpha/2$, $\left(\kappa^2 - \Delta_{\mathcal{M}}\right)^{\alpha/2}$ is a pseudo-differential operator defined in terms of the spectral decomposition of $\Delta_{\mathcal{M}}$.

Lindgren et al. [2011] solve the boundary value problem in (3.33) and (3.34) and obtain (3.35) by decomposing $\Delta_{\mathcal{M}}$ via finite element method. While this is an appealing approach, it requires that the input space geometry be known. Since this is not our case, we propose instead to work with the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$ as a proxy for $\Delta_{\mathcal{M}}$. Given that, in practice, $\tilde{\mathcal{L}}_{\epsilon,n}$ is known on the observed points $\mathcal{P} \cup \mathcal{P}^*$ rather than on the whole of $\mathcal{M}$, this means that any regression or classification performed using $\tilde{\mathcal{L}}_{\epsilon,n}$ is a form of *transductive learning*. This is in contrast with the *inductive* learning approach of Lindgren et al. [2011], where, because the manifold is known *a*

[6] See Definition 22.

priori, the discrete approximation to $\Delta_{\mathcal{M}}$ is defined for every point $x \in \mathcal{M}$ and hence so is the value of the Matérn field (3.35).

As a final note, it is worth pointing out that the boundary value problem in (3.33) and (3.34) generalizes and unifies previous ideas in Laplacian regularization, such as regularization on graphs (Smola and Kondor [2003]) or Laplacian and Iterated Laplacian regularization (Belkin et al. [2006], Zhou and Belkin [2011]). The case $\alpha = 1$ leads to the same kernel as the *Regularized Laplacian* from Smola and Kondor [2003] or the same covariance function as the GP used by Zhu et al. [2003]. Meanwhile $\kappa = 0$ with integer values of α leads to the same kernel as the Iterated Laplacian regularization (Zhou and Belkin [2011]).

3.4 The Matérn GF

Since we are now extending the notion of Matérn GF to manifolds, it is interesting to discuss the intuition that can be gained from the Matérn covariance function (3.32) and the Matérn SPDE (3.33). Specifically, there are three elements of interest in the pseudo-differential operator $(\kappa^2 - \Delta_{\mathcal{M}})^{\alpha/2}$: the smoothness α, the length scale $1/\kappa$ and the geometry $\Delta_{\mathcal{M}}$. We discuss these in turn below.

3.4.1 Smoothness α

The Matérn GF is a popular process in spatial statistics, in large part due to Stein [1999], who recommended its use. One of the key benefits Stein cites in favor of the Matérn GF is control over the smoothness of the stochastic process.

To clarify in what sense α controls the smoothness of the GF, we first need to define mean square continuity and mean square differentiability.

Definition 22 (Mean Square Continuity Adler [1981]) *Let $\{x_i\}_{i=1}^{\infty} \subset \mathcal{X}$ be a sequence converging to $x^* \in \mathcal{X}$, i.e. $||x_i - x^*|| \to 0$ as $i \to \infty$. Then if the GF ψ satisfies*

$$E[||\psi(x_i) - \psi(x)||^2] \to 0\,, \text{ as } i \to \infty\,, \tag{3.36}$$

we say that ψ is continuous in the mean square at x^*. If this holds for all $x^* \in A \subset \mathcal{M}$, we say that ψ is continuous in the mean square over A.

An important result about mean square continuity is its equivalence with continuity of the covariance function. Indeed, from the definition of mean square continuity one readily observes that ψ is continuous in the mean square at x^* if and only if the covariance function $k(x, z)$ of ψ is continuous at $x = z = x^*$.

One can also define the mean square derivative of ψ at x^* in direction e_j as

$$\psi_i(x^*) \equiv \underset{h\to 0}{\text{l.i.m.}} \frac{\psi(x + he_j) - \psi(x)}{h}\,, \tag{3.37}$$

where l.i.m. is the *limit in the mean square*. Just as for mean square continuity, $\psi_i(x)$ exits if and only if $\partial^2 k(x,z)/\partial x_j \partial z_j$ exists at $x = z = x^*$ (Adler [1981]).

Going back to the Matérn GF, we have that if ψ is a Matérn GF, then ψ will be continuous in the mean square if and only if the order parameter in (3.32) satisfies $\nu > 0$ (Rasmussen and Williams [2006]). This property of the Matérn GF extends to mean square differentiability: a ψ will be $k-$times mean square differentiable if and only if the order parameter in (3.32) satisfies $\nu > k$.

In light of (3.31), (3.30), and the fact that $\alpha = \nu + d/2$, it is clear that α must be greater than $d/2$ to ensure that the resulting stochastic process will be continuous in the mean square. Not surprisingly, this result extends to differentiability, whereby if $\alpha - d/2 > k$, the process will be $k-$times mean square differentiable. In the limit of $\nu \to \infty$, the process is infinitely mean square differentiable and if $\mathcal{M} = \mathbb{R}^d$, then the Matérn convariance function (3.32) converges to a squared exponential covariance function (Rasmussen and Williams [2006]). On a general manifold, one achieves the same effect by using the pseudo-differential operator $\mathcal{L} = \exp\left(-\Delta_{\mathcal{M}}/4l^2\right)$ in (3.29), where $1/l$ plays the same role of characteristic length scale as $1/\kappa$ in (3.31).

Interestingly, Zhou and Belkin [2011] also arrived at the conclusion that one should choose $\alpha > d/2$, for the $\kappa = 0$ case. Specifically, they show that the Sobolev space

$$H^{\alpha}(\mathcal{M}) = \left\{ \psi : \frac{\partial^{|r|}\psi}{\partial^{r_1} x_1 \cdots \partial^{r_d} x_d} \in \mathcal{L}^2(\mathcal{M}), \forall r \in \mathbb{N}^d \text{ s. t. } \sum_{i=1}^{d} r_i \leq \alpha \right\}, \tag{3.38}$$

associated with iterated Laplacian $\Delta^{\alpha}_{\mathcal{M}}$, defines a RKHS if and only if $\alpha > d/2$.

3.4.2 Length Scale $1/\kappa$

One notable difference between the differential operator $(\kappa^2 - \Delta_{\mathcal{M}})^{\alpha/2}$ from Equation (3.33) and the iterated Laplacian regularization of Zhou and Belkin [2011] is the introduction of κ^2. This hyperparameter controls the length scale, or rate of decay, of the Matérn covariance function, as can be seen in (3.32) for the $\mathcal{M} = \mathbb{R}^d$ case. In the

absence of this hyperparameter, the resulting differential operator, $\Delta_{\mathcal{M}}^{\alpha/2}$, has a non-trivial null space. Furthermore, the resulting stochastic process has no characteristic length scale, at least on input spaces that do not have a characteristic length scale of their own, such as $\mathbb{R}^d$.

In practice, one can ensure a trivial null space when $\kappa^2 = 0$ by adding a constant diagonal term to the resulting covariance function. Note that this amounts to adding measurement noise, as we did for GPR in Equation (3.8). This is the approach taken by Zhu et al. [2003], whose model is otherwise equivalent to the $\kappa \neq 0$ and $\alpha = 1$ case.

Another option for dealing with the non-trivial null space of $\Delta_{\mathcal{M}}^{\alpha}$ is to define the covariance function in (3.35) by using the pseudo-inverse of $\Delta_{\mathcal{M}}^{\alpha}$. One can then reintroduce the functions in the null space in the form of a mean function to the GF as described in Section 3.2.1. Proceeding in this manner introduces as many new hyperparameters as there are dimensions in the null space.

In any event, setting $\kappa^2 > 0$ clearly offers important benefits: a) from a technical point of view, by ensuring that the null space is trivial, and b) from an interpretation point of view, by providing a characteristic length scale for the regression or classification task.

3.4.3 Geometry $\Delta_{\mathcal{M}}$

As discussed in Chapter 2, the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ is equivalent to the intrinsic geometry $(\mathcal{M}, g)$ of the Riemannian manifold, in that either one can be recovered from the other. We are already implicitly exploiting this fact to extend the definition of the Matérn GF to manifolds, but we can make an observation of further interest here. Indeed, this correspondence means that transformations of the geometry $(\mathcal{M}, g)$ are tantamount to transformations of $\Delta_{\mathcal{M}}$, and vice versa.

Lindgren et al. [2011] formalize this by extending the deformation method developed by Sampson and Guttorp [1992] to manifolds and show how any embedding

of a manifold transforms the boundary value problem of (3.33) and (3.34) for the $\alpha = 1$ case. That is, Lindgren et al. [2011] define non-stationary Matérn GF on $\mathcal{M}$ by implying that the only stationary Matérn GF on $\mathcal{M}$ is the solution to the boundary value problem (3.33) and (3.34)[7]. Consequently, any Matérn GF on $\mathcal{M}$ that is not a solution to that boundary value problem is seen as non-stationary. In other words, any embedding $\phi : \mathcal{M} \to \mathcal{N}$ produces a transformation $\Delta_{\mathcal{M}} \to \Delta_{\mathcal{N}}$. Therefore, using $\Delta_{\mathcal{N}}$ in (3.35) can be interpreted either as a new Matérn GF on the embedding $\mathcal{N}$, or as a non-stationary GF on the original manifold $\mathcal{M}$.

Interestingly, this gives a new interpretation to non-linear dimensionality reduction algorithms that do not produce isometric embeddings: any regression or classification performed on such an embedding is a non-stationary version of that task performed on the original manifold. This obviously opens the door to finding transformations (embeddings) of the data that lead to useful non-stationary processes.

In practice, this can be done in two ways. First, the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$ could itself be transformed. The problem with this approach is that most variations of the graph Laplacian do not actually converge to the Laplace-Beltrami operator - they define a completely different differential operator that may or may not be geometrically grounded. Second, the data could be transformed through some embedding algorithm and then a new graph Laplacian could be computed in that space, ensuring that the new graph Laplacian is geometrically grounded. Obviously, if the embedding lends itself to it, one could simply perform the task directly in the (hopefully low-dimensional) ambient space of the embedding.

[7]Note that in Euclidean space, a stationary GF is a GF whose covariance function only depends on the distance between points.

3.5 Experiments

3.5.1 Data and Learning Tasks

For the experiments performed in this chapter, we use seven learning tasks: six classification tasks drawn from the work of Chapelle et al. [2006], and the regression on `faces` data set presented earlier in Section 2.6.

Chapelle et al. [2006] created several SSL data sets and associated learning tasks with the intention of providing a useful set of reference points, or benchmarks, against which researchers could measure their methods and explore areas for improvement. All the problems were designed to reflect and probe different assumptions that various existing algorithms build upon. They are all publicly available at:
http://www.kyb.tuebingen.mpg.de/ssl-book/.

These standard data sets and accompanying learning tasks include five derived from real data, and three artificially created ones. We limit ourselves to the problems in which the data is sampled from a Riemannian manifold, so we do not consider those with sparse discrete and sparse binary spaces. We also exclude the multiclass classification problem. The remaining six problems are described below, based largely on the text and figures provided in Chapelle et al. [2006].

- **Data Set 1:** `Digit1` - This artificial data set and classification problem were designed to consist of points close to a low-dimensional manifold embedded into a high-dimensional space, but not to show a pronounced cluster structure. The authors started from a system that generates artificial writings (images) of the digit "1" developed by Hein and Audibert [2005]. The images are constructed based on the image of an abstract "1" implemented as a function $[0,1]^2 \rightarrow \{0,1\}$, with the main vertical line ranging from $y = 0.2$ to $y = 0.8$ at $x = 0.5$. There are five degrees of freedom in the function: two for translations ($[-0.13, +0.13]$ each), one for rotation ($[-90°, +90°]$ each), one for line thickness ($[0.02, 0.05]$),

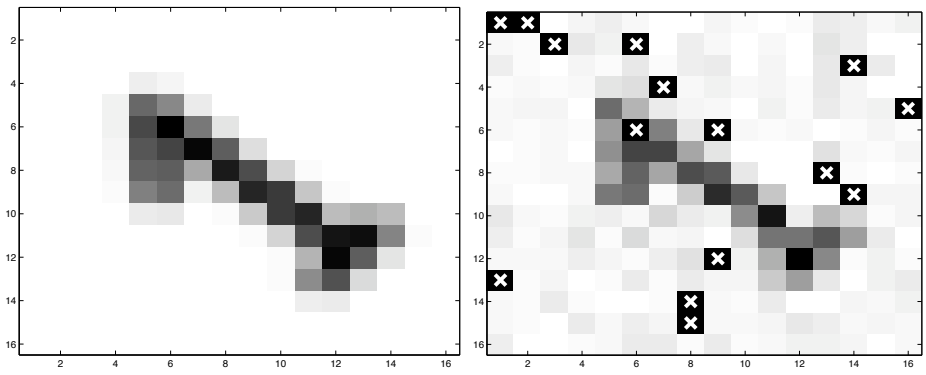


Figure 3.1: First data point from `Digit1`. (Left) Original image. (Right) After rescaling, adding noise, and masking dimensions (x). Figure and caption taken from Chapelle et al. [2006].

and Figure 3.1 displays the first data point as an example.

In order to construct the data set, the authors randomly sampled 1500 such images. The class label was set according to the tilt angle, with the boundary corresponding to an upright digit. To increase the difficulty of the task, the authors applied a sequence of transformations to the data as shown in Algorithm 21.1 from Chapelle et al. [2006] with σ set to 0.05. The result of this transformation (except for bias and permutation) applied to the first data point is shown in the right part of Figure 3.1. The data set has 2 classes, 241 dimensions, and 1500 data points.

- **Data Set 2:** `USPS` - This data set and classification problem are based on real data from the "famous" set of USPS handwritten digits. To construct **USPS**, the authors randomly drew 150 images of each of the ten digits. The digits "2" and "5" were assigned to the class $+1$, and all the others formed class -1. This produced two imbalanced classes, with relative sizes of 1:4.

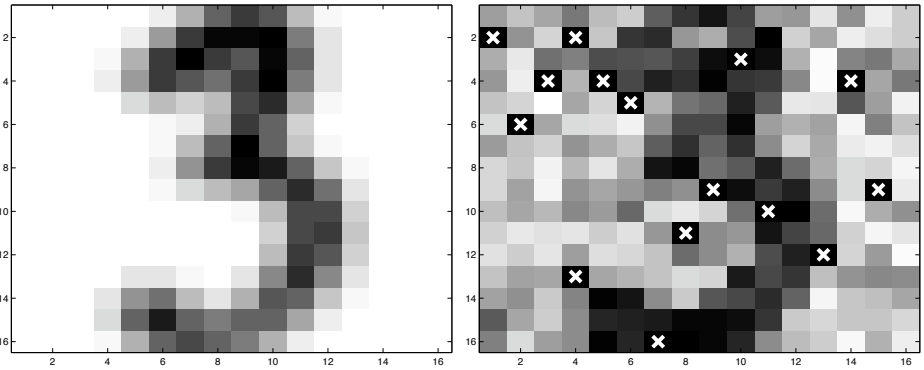


Figure 3.2: Fourth data point from `USPS`. (Left) Original image. (Right) After rescaling, adding noise, and masking dimensions (x). Figure and caption taken from Chapelle et al. [2006].

To prevent researchers from exploiting the known structure of the data, the authors again obscured it by applying Algorithm 21.1 from Chapelle et al. [2006], this time with $\sigma = 0.1$. The result of such an intervention is displayed in Figure 3.2.

- **Data Set 3:** `COIL`

  The Columbia object image library `COIL` is a set of color images of 100 different objects taken from different angles. To create this data set and related classification problem, the authors downsampled each image and then randomly selected 24 of the 100 objects, without revealing which objects were part of the selection. The selected objects were then partitioned into two classes of twelve objects each, further downscaled and obscured with Algorithm 21.1 from Chapelle et al. [2006]. The process is shown in Figure 3.3 below. The final data set has 2 classes, 241 dimensions, and 1500 points.

- **Data Set 4:** `BCI` - This data set and related classification problem are based on

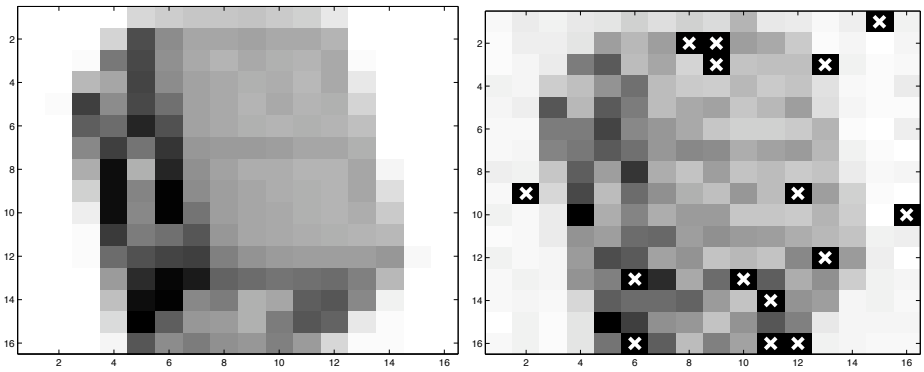


Figure 3.3: First data point from `COIL`. (Left) Original image. (Right) After rescaling, adding noise, and masking dimensions (x). Figure and caption taken from Chapelle et al. [2006].

real data resulting from research toward the development of a brain computer interface (hence the name BCI) (Lal et al. [2004]). A single person performed 400 trials in each of which he imagined movements with either the left hand (class -1) or the right hand (class $+1$). In each trial, EEG (electroencephalography) was recorded from 39 electrodes. An autoregressive model of order 3 was fitted to each of the resulting 39 time series. The trail was represented by the total of $117 = 39 \times 3$ fitted parameters.

- **Data Set 5:** `g241c` - This artificial data set and related classification problem were generated such that the classes correspond to clusters, but the manifold assumption does not hold in that the data does not lie on a low-dimensional subspace of the ambient space. First, 750 points were drawn from each of two unit-variance isotropic Gaussians, the centers of which had a distance of 2.5 in a random direction (i.e $\parallel \mu_1 - \mu_2 \parallel = 2.5$). The class label of a point represents the Gaussian it was drawn from. Finally, all dimensions are standardized: shifted and rescaled to zero-mean and unit variance. Figure 3.4 shows a two-

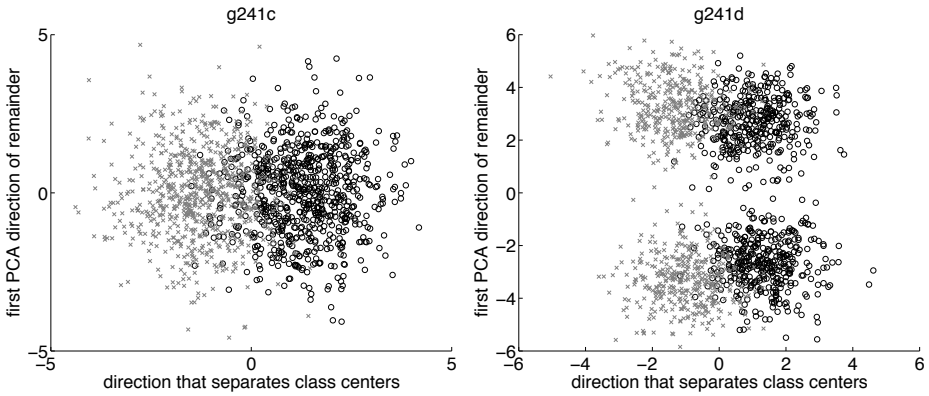


Figure 3.4: Two-dimensional projections of `g241c` (left) and `g241d` (right). Black circles indicate class +1, while gray crosses indicate class -1. Figure and caption taken from Chapelle et al. [2006].

dimensional projection of the data.

- **Data Set 7:** `g241d` - This artificial data set and related classification task were constructed to exhibit a potentially misleading cluster structure, and no manifold structure. The first 375 points were drawn from each of two unit-variance isotropic Gaussians, the centers of which have a distance of 6 in a random direction; these were grouped into the class +1. Then, the centers of two other Gaussians for class −1 were fixed by moving a distance of 2.5 from each of the former centers in a random direction. Again, the identity matrix was used as a covariance matrix, and 375 points were sampled from each new Gaussian. The right panel of Figure 3.4 shows a two-dimensional projection of `g241d` data.

3.5.2 Models

For the classification problems described above, we use the Logistic GPC approach detailed in Section 3.2.1. For the `faces` data - a regression problem - we use the GPR approach detailed in Section 3.2.1.

We also implement the Iterated Laplacian penalized empirical risk model

$$\min_{\psi} \sum_{x_i \in \mathcal{P}} (\psi(x_i) - y_i)^2 + \mu \, ||\psi||^2_{k_\Theta} \, , \tag{3.39}$$

where $k_\Theta = \Delta_{\mathcal{M}}^{-\alpha}$ but also consider the $k_\Theta = (\kappa^2 - \Delta_{\mathcal{M}})^{-\alpha}$ case. Note that just as in Zhou and Belkin [2011], we use the squared error loss for both classification and regression problems.

We select the (3.39) model primarily because the minimization problem has a relatively easy solution:

$$\hat{\mathbf{\Psi}}(\mathcal{P} \cup \mathcal{P}_*) = (S + \mu K_\theta)^\dagger S \mathbf{y} \, , \tag{3.40}$$

where the superscript † stands for the pseudoinverse and $S = \text{diag}\{ 1_{[x \in \mathcal{P}]}\}$, i.e. the diagonal matrix with ones for the points in the training set $\mathcal{P}$ and zeroes for the points in the test set $\mathcal{P}_*$.

Obviously, our choice is also motivated by our wish to compare our results to those of Zhou and Belkin [2011]. We note that even though we are using the least squared loss function rather than the 0-1 or the hinge loss functions, which are better suited for classification problems, prior studies indicate that their performance is comparable in this task (Rasmussen and Williams [2006]).

3.6 Results

In this section, we present the results we obtain when applying the models presented in previous sections to the standard SSL classification problems from Chapelle et al. [2006], and to the additional `faces` regression problem. We begin with a pictorial representation of our task, and then move on to discuss the models' performance.

Figure 3.5 provides an example of our analysis in a regression setting. It is based on data from `faces`, where the dependent variable y is the position of the head from right to left, as measured in degrees. We embedded the data using Isomap and then colored in black the points for which the labels y (the degree values) are known. We chose the Isomap embedding because it best displays the structure of the data in this case and is most useful for illustration. Information obtained from the covariance function allows us to map the influence any given point has on its neighbors, and this is portrayed in the lower left panel. Then, once a regression is performed, the regression errors can be visualized as in the lower right panel. For the purposes of this example, we used the GPR (Gaussian Process Regression).

Figure 3.6 provides an example of our analysis in a classification setting. It is based on `Digit1`, where we again colored in black the points for which the labels y are known. This time, we embedded the data embedded using LTSA because it was the embedding that best displayed the structure of the data and was most useful for illustration. Information obtained from the covariance function allows us to map the influence any given point has on its neighbors, and this is portrayed in the lower left panel. The misclassification errors are shown in the lower right panel, with colors corresponding to the "correct" outcome. For the purposes of this example, we used GPC (Gaussian Process Classification).

Table 3.1 shows the performances of the GP model (GPR for regression and GPC for classification, see Section 3.2.1) and the kernel model (3.23). In both cases, the bandwidth parameter $\sqrt{\epsilon}$ is optimized separately for each data set and model using

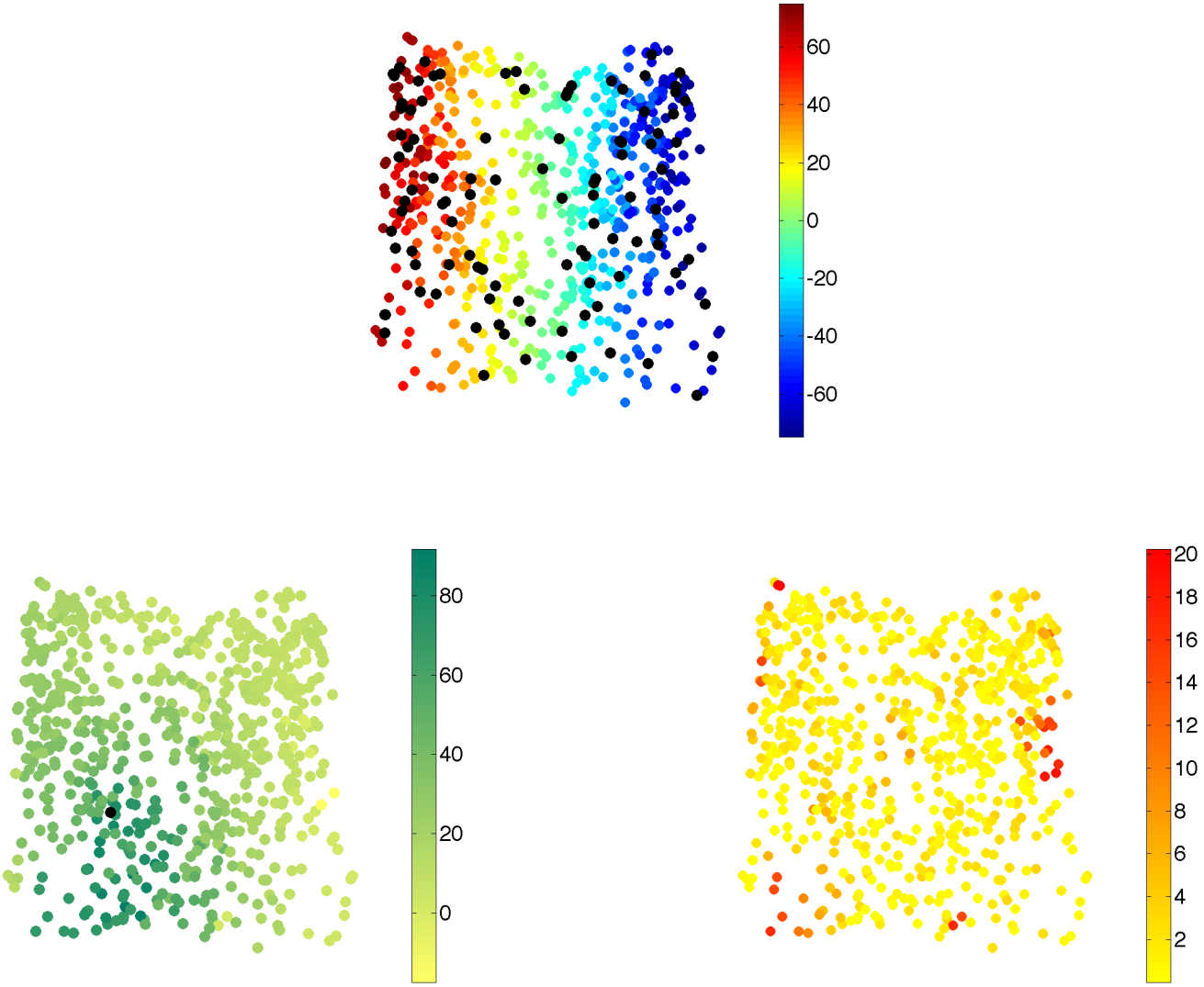


Figure 3.5: Data set `faces` embedded using Isomap. The top panel is a representation of the data, with known labels as black dots. The bottom left panel shows the impact of the covariance matrix: the strength of the influence of the data point in black on neighboring points, with darker shades representing greater influence. The bottom right panel shows the absolute value of the prediction errors as a percentage of the range of the data.

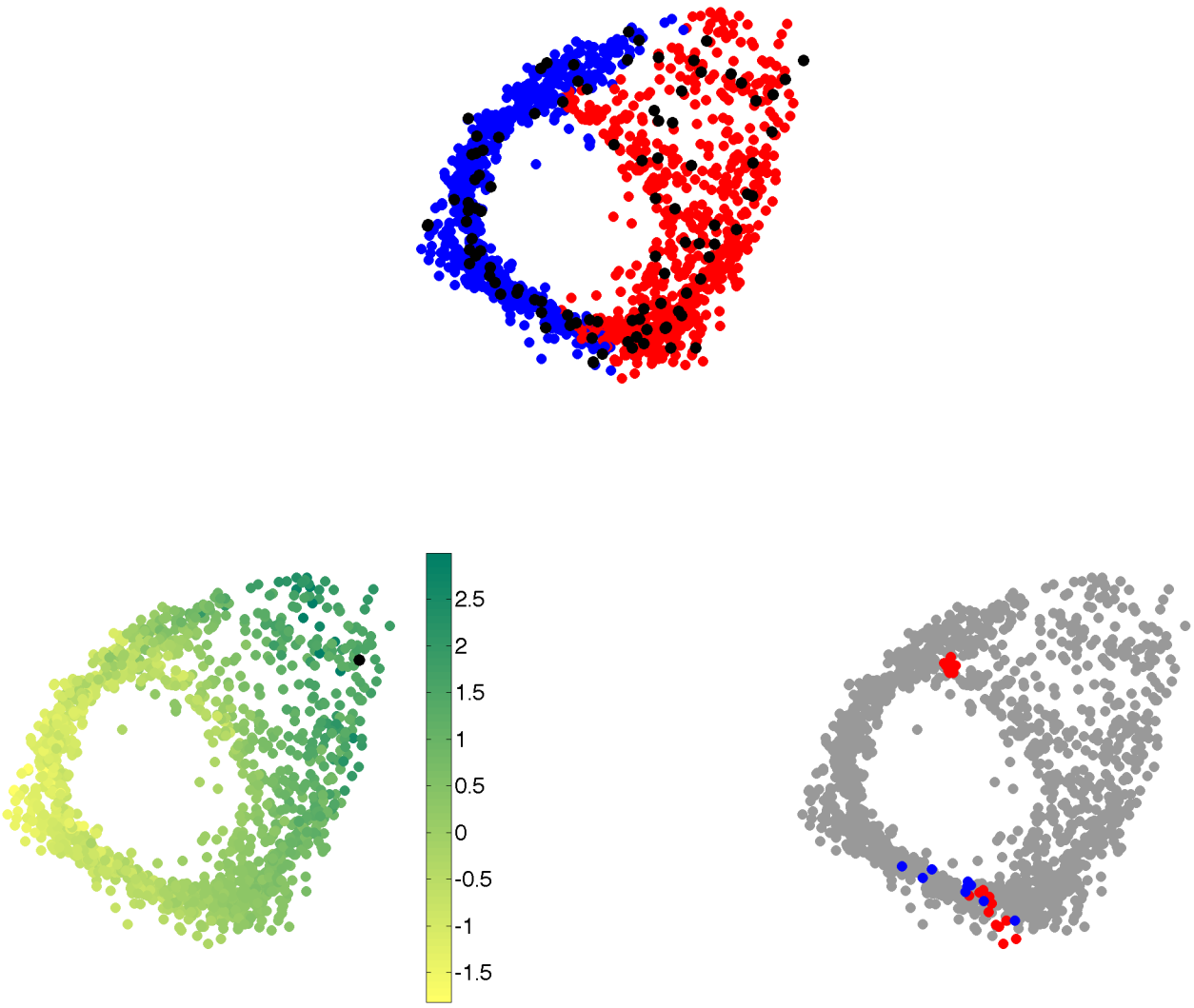


Figure 3.6: Data set `Digit1` embedded using LTSA. The top panel is a representation of the data, with known labels as black dots. The bottom left panel shows the impact of the covariance matrix: the strength of the influence of the data point in black on neighboring points, with darker shades representing greater influence. The bottom right panel shows the misclassification error.

	Digit1	USPS	COIL	BCI	g241c	g241d	faces
GP	1.67	4.01	8.05	45.72	13.67	11.52	9.44
Kernel	2.13	3.91	7.58	49.36	12.83	20.94	15.80

Table 3.1: Comparison of classification error (SSL classification tasks) and mean squared error (faces), using GP and Kernel model. The best performing model is shown in red for each data set and corresponding learning task.

the method we discuss in detail in Chapter 4.

While it is difficult to declare that one method performs better than the other overall, we note that in cases where the kernel method outperforms the GP, the difference in the results is not large; however, in cases where the GP outperforms the kernel method, the improvement is significant, sometimes cutting the classification error or root mean square error by as much as 50% (except in the BCI case, which is very difficult to learn for all methods). While it might be expected that GP would have an advantage in classification tasks since it uses a loss function tailored to classification, we see that it also does very well for regression (faces data set), where both methods use the same loss function.

Next, we examine the impact of the new parameters that have been made available through the application of the Matérn GF. We perform the same classification and regression tasks, but this time we fix either the length scale ($1/\kappa$) or the smoothness (α). Table 3.2 shows the results of this experiment.

Setting $\kappa = 0$ appears to lead to a worsening of the performance of the GP model for all data sets, with the exception of Digit1, for which there is no change, and g241d, for which there is some improvement. Restricting the smoothness parameter to integer values leads to better performance for data sets Digit1, COIL, and faces. This is might be due to the fact that integer values of α help avoid overfitting by testing fewer hypotheses.

	Restriction	Digit1	USPS	COIL	BCI	g241c	g241d	faces
GP	None	1.67	4.01	8.05	45.72	13.67	11.52	9.44
	$\kappa^2 = 0$	1.67	6.22	15.38	46.19	13.84	9.53	18.50
	α integer	1.64	4.05	7.97	46.83	15.43	13.25	6.61
Kernel	None	2.13	3.91	7.58	49.36	12.83	20.94	15.80
	$\kappa^2 = 0$	2.11	3.89	8.81	48.67	12.77	8.76	35.98
	α integer $\kappa^2 = 0$	2.22	4.91	9.51	49.33	12.95	8.84	26.95

Table 3.2: Comparison of classification error (SSL classification tasks) and mean squared error (faces) obtained with GP and Kernel models, with restrictions on parameters learned. The best performing GP model is shown in red, and the best performing Kernel model is shown in blue for each data set and corresponding learning task.

The kernel model responds better to setting $\kappa = 0$: it leads to better performance in all cases except for data sets COIL and faces. Fixing the smoothness parameter α leads to a worsening of the performance in all cases except for g241d.

However, it is important to remind the reader that setting $\kappa = 0$ or using very small values of κ means that the k_Θ will have a non-trival null-space; thus, obtaining the solution will require computing the pseudo-inverse of the kernel. This additional computational cost should be balanced against any observed improvement in performance.

Overall, we note that the best results we obtained in the scenarios we presented outperform previously published results (Chapelle et al. [2006]), including the $\alpha = 1$ and $\kappa = 0$ case from Belkin et al. [2006], for all datasets with the exception of BCI and g241d. In the latter case, the superior performance of the cluster kernel method can be explained by the fact that g241d contains clustered data by design. Our lack

of improvement over current methods for the `BCI` case will be addressed in Section 4.4.1 below.

3.7 Discussion

In this chapter, we exploited the fact that stochastic partial differential equations based on the Laplace-Beltrami operator can be used to define the popular Matérn GF on the manifold in order to introduce geometrically motivated non-parametric Bayesian regression or - equivalently - geometrically motivated kernel regularization.

Our main contribution is to define a covariance function over a manifold of unknown geometry from a finite sample that is both interpretable and flexible.

Indeed, the three elements of interest in a Matérn GF - the smoothness α, the length scale $1/\kappa$ and the geometry $\Delta_{\mathcal{M}}$ - all have interesting implications in learning tasks.

First, assuming some control over the smoothness parameter α allows us to determine the number of mean square derivatives of the process and avoid the risk of over-smoothing the data. The latter is a common problem with other popular covariance functions, such as the squared exponential, which, as we stated before, has an infinite number of mean square derivatives.

Second, using a non-trivial length scale parameter $1/\kappa$ also has important benefits from an interpretation point of view, since it provides a characteristic length scale for the regression or classification task. In addition, from a technical point of view, $\kappa^2 > 0$ ensures that the null space is trivial.

Finally, the geometry element $\Delta_{\mathcal{M}}$ may be the most interesting of all, and is worth studying further. As discussed in Chapter 2, the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ is equivalent to the intrinsic geometry $(\mathcal{M}, g)$ of the Riemannian manifold, in that either one can be recovered from the other. This correspondence allows us to understand that transformations of the geometry $(\mathcal{M}, g)$ are tantamount to transformations of $\Delta_{\mathcal{M}}$. More importantly, it also leads us to conclude that any non-isometric embedding of a manifold effectively defines a non-stationary process when compared to the Matérn GF on the original geometry. Ultimately, it means that determining the appropriate

non-stationary process for a particular task is essentially equivalent to determining the appropriate geometry on which a task is to be performed.

One element of the geometry that we have not yet discussed is the bandwidth $\sqrt{\epsilon}$ and its effect on $\tilde{\mathcal{L}}_{\epsilon,n}$. That is to say, we have not addressed the question of how the bandwidth affects the geometry the graph Laplacian encodes from the observed data. This "observed" geometry plays a substantial role in the success of semi-supervised learning based on the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$. Given its significance, we will devote the next chapter to its study.

Chapter 4

EXPLOITING GEOMETRY TO LEARN THE KERNEL BANDWIDTH

4.1 The Problem of Estimating the Bandwidth of the Graph Laplacian

As discussed in previous chapters, numerous methods for both semi-supervised (Belkin et al. [2006], Zhu et al. [2003], Zhou and Belkin [2011], Sindhwani et al. [2007], Smola and Kondor [2003]) and unsurpervised learning (dimensionality reduction and clustering - see Belkin and Niyogi [2002], von Luxburg et al. [2008]) rely on the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$. This has led to sustained interest in determining the properties of the graph Laplacian, especially when it is constructed from data points $\mathcal{P} = \{x_1, \ldots, x_n\}$ in high-dimensional Euclidean space $\mathbb{R}^r$. Specifically, a great deal of attention has been devoted to establishing methods for constructing $\tilde{\mathcal{L}}_{\epsilon,n}$, and then studying its asymptotic limit (as well as that of its eigendecomposition) as the number of data points n goes to infinity (Giné and Koltchinskii [2006], Belkin and Niyogi [2007], Hein et al. [2007], Ting et al. [2010]). We have explored the most important $\tilde{\mathcal{L}}_{\epsilon,n}$ constructions in Section 2.1.

Considerably fewer studies have focused on the optimality of the free parameters used to construct $\tilde{\mathcal{L}}_{\epsilon,n}$ in a finite sample problem. For instance, there is no principled method for setting one of the most important parameters - the scale, or bandwidth $\sqrt{\epsilon}$. The common approaches consist of "tuning" it through cross-validation in the semi-supervised context. However, as non-linear dimensionality reduction is an unsurpervised problem, there is no context in which to apply cross-validation and, *a priori*, no clear way to estimate the optimal $\sqrt{\epsilon}$. Indeed, most treatments of this problem rely on guesswork, ocular inspection or various ad-hoc methods such as minimizing reconstruction error in order to set $\sqrt{\epsilon}$ (Lee and Verleysen [2007], Chen and Buja [2009], Levina and Bickel [2005], Carter et al. [2007], Chen et al. [2011]). While these approaches may yield a usable parameter, they offer no *principled* reason for its selection and generally do not aim to improve the graph Laplacian estimator *per se*.

The most interesting theoretical result for choosing $\sqrt{\epsilon}$ (Singer [2006]) derives the

optimal rate at which $\epsilon \to 0$ as the sample size $n \to \infty$.

$$\epsilon = \frac{C(\mathcal{M})}{n^{1/(3+d/2)}} \tag{4.1}$$

by optimizing the bias-variance trade-off for the graph Laplacian. The problem is that there is no prescribed method for calculating $C(\mathcal{M})$, a constant that depends on the yet unknown geometry of the data.

In practice, one is left with tuning ϵ until a "reasonable" embedding is obtained. An alternative is to set $\epsilon = \infty$ and replace $k(x_i, x_j)$ by a $k-$nearest neighbors graph. This amounts to replacing the problem of selecting ϵ for that of selecting k. While it might be easier to guess k than it is to guess ϵ, this does not compensate for the fact a graph Laplacian constructed from a k-nearest neighbors graph will not converge to the expected operator as discussed before.

In this Chapter, we present a new, principled approach to selecting the bandwidth parameter of $\tilde{\mathcal{L}}_{\epsilon,n}$, mainly inspired by the fact that the graph Laplacian, when properly constructed, converges to the Laplace-Beltrami $\Delta_{\mathcal{M}}$ operator on the data manifold $\mathcal{M}$ (Hein et al. [2007]). Indeed, because we have established in Chapter 2 that $\Delta_{\mathcal{M}}$ is equivalent to the intrinsic geometry of $(\mathcal{M}, g)$, i.e. the data manifold and its Riemannian metric g (Rosenberg [1997]), we can choose to select the bandwidth parameter that yields the graph Laplacian that is most faithful to the geometry of the original data. This approach presents the advantage of relying purely on the original representation of the data and on the property that the $\tilde{\mathcal{L}}_{\epsilon,n}$ encodes the intrinsic geometry of the data in a graph structure. As such, it can be used both in the unsupervised setting, to improve embeddings obtained via eigendecomposition of $\tilde{\mathcal{L}}_{\epsilon,n}$, and in the semi-supervised setting where $\tilde{\mathcal{L}}_{\epsilon,n}$ is a popular regularizer, but where estimating $\sqrt{\epsilon}$ by cross-validation can be costly and can increase the risk of overfitting (Ng [1997]).

4.2 Geometrical Consistency

We begin our discussion of our proposed method for estimating $\sqrt{\epsilon}$ by appealing to Proposition 10 from Chapter 2. We remind the reader that Proposition 10 means that we can recover the Riemannian metric of a manifold for any local coordinates or - equivalently - for any diffeomorphism, provided that the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$ is known. This also applies to the trivial diffeomporhism, i.e. the inclusion map $\jmath$[1]. What makes the inclusion map particularly interesting is that its metric is immediately known to us, following the standard manifold learning assumption according to which $\mathcal{M} \in \mathbb{R}^r$ inherits its metric from $\mathbb{R}^r$. Specifically, the Riemannian metric of $\mathcal{M}$ is $\delta_r|_{T\mathcal{M}}$, where $\delta_r|_{T\mathcal{M}}$ stands for the restriction of the induced metric δ_r, i.e. the natural metric of the ambient space $\mathbb{R}^r$, to the tangent bundle $T\mathcal{M}$ of the manifold $\mathcal{M}$.

This allows us to estimate the Riemannian metric of $\mathcal{M}$ in two different ways: first, by obtaining $\delta_r|_{T\mathcal{M}}$ from the inclusion map as described above; and second, by using Proposition 10 on the inclusion map to obtain $g_\jmath(x)$. Because these are really the same metric, we have:

$$g_\jmath(x) \equiv \delta_r|_{T_x\mathcal{M}} \, \forall x \in \mathcal{M} \, . \tag{4.2}$$

Since we do not expect the estimators of $g_\jmath(x)$ and $\delta_r|_{T_x\mathcal{M}}$ to be equal in practice, we need a way to evaluate how similar they are. For this purpose, we define a distortion between two Riemannian metrics.

Definition 23 (Riemannian Metric Distortion) *Let h, k, and g_0 be Riemannian metrics on $\mathcal{M}$, then the distortion between h and k with respect to g_0 is given by*

$$\mathcal{D}_{g_0}(h,k) \;=\; \int_{\mathcal{M}} ||h-k||^2_{g_0} \, dV_{g_0} \, . \tag{4.3}$$

---

[1] The inclusion map is the isometric embedding of the manifold $\mathcal{M}$ in Euclidean space

In (4.3), $||\cdot||_{g_0}$ is the Riemannian metric norm and dV_{g_0} is the volume element with respect to the Riemannian metric g_0. Specifically, the Riemannian metric norm for h with respect to g_0 is defined via the Riemannian metric inner product

Definition 24 (Riemannian Inner Product and Norm) *The Riemannian inner product between two Riemannian metrics h and k with respect to a third Riemannian metric g_0 is given in local coordinates as*[2]

$$< h, k >_{g_0} = g_0^{i_1 j_1} g_0^{i_2 j_2} h_{i_1 j_1} k_{i_2 j_2} \, . \tag{4.4}$$

The Riemannian norm of Riemannian metric h with respect to g_0 is then given by

$$||h||_{g_0} = < h, h >_{g_0} \ . \tag{4.5}$$

Meanwhile, the volume element dV_{g_0} is given in local coordinates by $dV_{g_0} = \sqrt{\det(g_0)} dx^1 \dots x^d$, just as in Section 2.2.1.

In (4.3) we introduced a third Riemannian metric g_0 to ensure that $\mathcal{D}_{g_0}(h, k)$ is symmetric in h and k and hence defines a distance function on the space of all Riemannian metrics on $\mathcal{M}$. In practice, however, nothing prevents us from using either k or h as the base metric for defining $\mathcal{D}$, as long as we understand $\mathcal{D}$ to represent the distance of its *arguments*.

Combining (4.2) and (4.3), we now have a principled approach to evaluating how close a "Laplacian-like" operator is to encoding the geometry of the manifold $\mathcal{M}$.

Indeed, if the the graph Laplacian is to encode the geometry of the data manifold, its parameter(s) should correspond to those that minimize the distortion with respect to the true original geometry.

Recalling the operator used to define $\mathcal{L}_{\epsilon,\lambda}$ from the heat kernel (2.1)

---

[2]Note that this is a special case of what is known as a tensor inner product, see Lee [1997] for details.

$$\begin{aligned}
\mathcal{L}_{\epsilon,\lambda}(f)(x) &= \int_{\mathcal{M}} \frac{w_{\epsilon,\lambda}(x,y)}{d_{\epsilon,\lambda}})f(y)dV_g(y)\,, \text{ where} \\
d_{\epsilon,\lambda}(x) &= \int_{\mathcal{M}} w_{\epsilon,\lambda}(x,y))dV_g(y)\,, \text{ and } w_{\epsilon,\lambda} = \frac{w_\epsilon(x,y)}{d_\epsilon^\lambda(x)d_\epsilon^\lambda(y)}\,, \text{ while} \\
d_\epsilon(x) &= \int_{\mathcal{M}} w_\epsilon(x,y)dV_g(y)\,,
\end{aligned} \tag{4.6}$$

we would expect that the "best" choice for ϵ would be given by

$$\hat{\epsilon} = \text{argmin}_\epsilon \mathcal{D}\left(g_J(\epsilon,\lambda), \delta_r|_{T\mathcal{M}}\right)\,, \tag{4.7}$$

with $g_J(\epsilon,\lambda)$ obtained by using $\mathcal{L}_{\epsilon,\lambda,n}$ instead of $\Delta_{\mathcal{M}}$ in (2.6).

4.2.1 The Finite Sample Problem

In the asymptotic limit of $\epsilon \to 0$ and with $\lambda = 1$, it is well established that $\mathcal{L}_{\epsilon,1,n} \to \Delta_{\mathcal{M}}$ and hence $\mathcal{D}\left(g_J(0,1), \delta_r|_{T\mathcal{M}}\right) = 0$, i.e. $\hat{\epsilon} = 0$ (Hein et al. [2007]).

For a finite sample and a family of graph Laplacians $\tilde{\mathcal{L}}_{\epsilon,n}$ constructed using the heat kernel (2.1), the graph Laplacian that best captures the geometry of the data will be the one indexed by $\hat{\epsilon}_n$, obtained by minimizing the empirical distortion equation (4.9) below. Note, however, that one cannot simply use the theoretical result $\hat{\epsilon} = 0$ to construct the graph Laplacian, since doing so yields a degenerate result when working with a finite sample. Indeed, the distortion (4.3) serves as a geometrically motivated objective function to estimate what value of $\hat{\epsilon}$ strikes the optimal balance between the two degenerate graph Laplacians: the discrete geometry of n disjoint points for $\epsilon \to 0$ and the degenerate singular point geometry/graph Laplacian when $\epsilon \to \infty$.

As far as the other parameters, e.g. λ, are concerned, one could consider estimating them using the same distortion minimizing argument (4.7). However, there is is a prescribed value of λ ($\lambda = 1$) that guarantees the desired asymptotic limit in our case: the Laplace-Beltrami operator. As this value is well defined in the finite limit, and since using other values of λ has been shown to produce differential operators that

are not purely geometric (Nadler et al. [2006a]), we have chosen to focus on learning ϵ and refraining from extending to λ in our analysis.

Our discussion thus far leads us to the claim that, within the family of graph Laplacians constructed using the heat kernel, the graph Laplacian that best captures the geometry of the data will be the one indexed by $\hat{\epsilon}_n$. Even though we limit ourselves to $\sqrt{\epsilon}$ here, the success of this approach suggests that geometrical faithfulness, as expressed by (4.3), is a *principled* criterion to estimate the free parameters used to construct $\tilde{\mathcal{L}}_{\epsilon,n}$ in a finite sample problem for both unsupervised and semi-supervised learning.

4.2.2 Alternative Approaches

Naturally, there are alternatives to the approach we propose. We specifically address three of these below: 1) using the Laplacian as an operator; 2) using alternative definitions of geometric distortion; and 3) using cross-validation for semi-supervised learning.

1) Laplacian as an Operator - The obvious alternative choice is to require that the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$ be as "close" as possible to $\Delta_{\mathcal{M}}$ in an operator sense rather than in a geometric sense. That is, we could aim to minimize some operator norm $\left|\left|\Delta_{\mathcal{M}}|_{\mathcal{P}} - \tilde{\mathcal{L}}_{\epsilon,n}\right|\right|$ over ϵ, where $\Delta_{\mathcal{M}}|_{\mathcal{P}}$ stands for the restriction of the Laplace-Beltrami operator to the observed data points $\mathcal{P} = \{x_1, \ldots, x_n\}$. The problem with this approach is that we only have one empirical estimator of the Laplace-Beltrami operator; however, in order to compute this operator norm in a finite sample, we would need two independent ways of finding an empirical estimator that would converge to $\Delta_{\mathcal{M}}$ in the $n \to \infty$ limit. In contrast, the power of our proposed approach to estimating (4.7) resides in the fact that we know the theoretical value of the Riemannian metric on the manifold; as such, it can be estimated independently from the Riemannian metric recovered from the Laplace-Beltrami operator, $g_j(\epsilon, \lambda)$.

We also note that employing the geometry-based approach is, in the limit, equiv-

alent to the idea of minimizing $\left|\left|\Delta_{\mathcal{M}}|_{\mathcal{P}} - \tilde{\mathcal{L}}_{\epsilon,n}\right|\right|$ over ϵ. Indeed, if the Riemannian metric is recovered, then, by definition of the Laplacian in local coordinates:

$$\Delta_{\mathcal{M}} f \equiv \frac{1}{\sqrt{|g|}} \partial_i \left(\sqrt{|g|} g^{ij} \partial_j \right) f \in C^2(\mathcal{M}), \tag{4.8}$$

we can also reconstruct the $\Delta_{\mathcal{M}}$. Thus, we have implicitly recovered the operator.

2) Alternative Definitions of Geometric Distortion - Using the distortion or the reconstruction error relies on the specific embedding algorithms used. As such, the estimated $\hat{\epsilon}$ won't be an intrinsic property of the observed data $\mathcal{P}$, but rather will depend on the designated embedding algorithm used. Indeed, as we have established in Chapter 2, these algorithms, for the most part, are not guaranteed to provide a true isometric embedding of the data.

3) Cross-Validation - In the semi-supervised learning context, there exists a common alternative to estimating the parameters of the graph Laplacian: using a learning task, such as in Laplacian regularization (Zhu et al. [2003], Belkin et al. [2006], Sindhwani et al. [2007], Zhou and Belkin [2011]). This approach allows the user to leverage cross-validation to find the parameters of the graph Laplacian best suited to the specific task at hand.

Of course, the first limitation of this approach is that it cannot be applied in an unsupervised setting. In addition, the need to minimize an additional cross-validation parameter, the bandwidth $\sqrt{\epsilon}$, increases both the complexity of the task (the extra parameter often introduces a non-linear optimization problem) and the risk of overfitting (Ng [1997]). In fact, as we discuss in section 4.4 below, we have observed overfitting empirically when taking this approach. Finally, the cross-validation approach requires recomputing the Laplacian for every single value of the parameter; since computing the Laplacian - especially if an eigendecomposition is required - is already a computationally costly operation in itself, this approach requires considerable computer power. For example, the iterated Laplacian method of Zhou and Belkin [2011] requires taking multiple powers of the graph Laplacian, or its eigendecomposition in

the case of fractional powers of the graph Laplacian.

4.2.3 Estimating the Distortion

Now that we have established the principle for setting ϵ, we will discuss how to apply it in practice.

The distortion between two metrics g and h, defined in (4.3) above, takes the form of an expectation over $(\mathcal{M}, g)$. Therefore, appealing to the law of large numbers, the natural choice for a distortion estimator in the finite sample case is the empirical distortion

$$\hat{D}\left(g_{\jmath,n}, \delta_r|_{T_x\mathcal{M},n}, \epsilon\right) = \frac{1}{n}\sum_{i=1}^{n}\left|\left|g_{\jmath,n}(x_i) - \delta_r|_{T_{x_i}\mathcal{M},n}\right|\right|^2_{\delta_r|_{T_{x_i}\mathcal{M},n}}. \tag{4.9}$$

where $g_{\jmath,n}(x_i)$ is the discretized version of $g_\jmath(x)$ obtained by applying (2.6) to the sampled data $\mathcal{P}$ with $\Delta_\mathcal{M}$ replaced by the graph Laplacian $\tilde{\mathcal{L}}_{\epsilon,n}$ and where $\delta_r|_{T_{x_i}\mathcal{M},n}$ is the discretized version of $\delta_r|_{T_x\mathcal{M}}$. In the following paragraphs, we describe how each of these is estimated.

Estimating $g_{\jmath,n}(x_i)$ and $\delta_r|_{T_x\mathcal{M}}$

When computing $g_{\jmath,n}(x_i)$, we are working with the inclusion map, and we must - just as before - find the dual metric by applying (2.6) on every coordinate pair of the ambient space. Having obtained the $r \times r$ dual metric, we then need to compute its pseudo-inverse to obtain the metric g. Repeating this procedure for every point in the sample renders the approach not only computationally intensive, but also numerically unstable.

Indeed, as discussed in Chapter 2, when performing dimensionality reduction, we can assume that the ambient space dimension r is very high and computing (2.6) at every point would entail $n \cdot r^2$ computations. What's more, $g_{\jmath,n}(x_i)$ is a r^2 matrix with theoretical rank $d = \dim(\mathcal{M})$, where $d \ll r$. In practice, this means that the estimated rank of $g_{\jmath,n}(x_i)$ will be dominated by $r - d$ superfluous dimensions.

Another problem is that computing $\delta_r|_{T_{x_i}\mathcal{M},n}$ actually requires knowing the tangent plane to the yet unknown manifold. In other words, the theoretical value of the of the Riemannian metric is only useful once we have an estimate of the tangent bundle $T\mathcal{M}$.

We solve both problems by estimating $T\mathcal{M}$. Specifically, we evaluate the tangent plane around each sampled point using local Principal Component Analysis (PCA) and then use the resulting tangent bundle to find the restriction of δ_r, thus obtaining an estimate of $\delta_r|_{T_{x_i}\mathcal{M},n}$. The tangent bundle also serves to define a local coordinate chart, which allows us to compute $g_{\jmath,n}(x_i)$ on the projection of the data on the new plane rather than on the original set. We have thus reduced the number of required computations in applying (2.6) from $n \cdot r^2$ to $n \cdot d^2$ where $d \ll r$, as mentioned above.

A new problem now arises: how do we define the appropriate neighborhood size when performing the local PCA for the purpose set out in the previous paragraph? Recall that our primary objective is to find the bandwidth / scale parameter of the graph Laplacian that allows the graph Laplacian to best encode the geometry of the original data. We have established that one of the intermediary steps requires us to project the data onto a tangent bundle $T\mathcal{M}$ through local PCA due to computational and numeric challenges. However, the choice of neighborhood in the local PCA defines the geometry of the projected data. As such, for the sake of consistency, and to ensure that the geometry we encode is common to all the transformations we perform, we opt to equate the notion of neighborhood in the local PCA with that embodied in the heat kernel by equating the two bandwidths. This is in keeping with the approach of Aswani et al. [2011], who also choose to equate neighborhoods for local PCA and local kernel regression. In the experiments, for the sake of completeness, we also implement a version of our method that does not equate the two bandwidths and compare the results.

Equating the two notions of neighborhood means that, rather than using the $k-$nearest neighbors or $\sqrt{\epsilon}$-balls for conducting local PCA, as is common practice,

we conduct a weighted local PCA (wlPCA), where the weights are defined by those of the heat kernel used to produce the graph Laplacian (2.1) centered around x_i. This approach is similar to sample-wise weighted PCA of Yue et al. [2004], with two important constraints: the weights must decay rapidly away from x_i, and the data must be centered to have zero mean such that all the points far from x_i are mapped close to the origin. The first requirement is obviously satisfied by the heat kernel, while the second one is achieved by performing PCA using the recentered design matrix Z where each row of Z is given by

$$\begin{aligned} z_j &= \tilde{w}_{i,j}(x_j - \bar{x}(w)), \text{ with} \\ \bar{x}(\tilde{w}) &= \frac{1}{n}\sum_{j=1}^{n} \tilde{w}_{i,j} x_j, \text{ and} \\ \tilde{w}_{i,j} &= w_{i,j} / \frac{1}{n}\sum_{j=1}^{n} w_{i,j}. \end{aligned} \tag{4.10}$$

Here $w_{i,j}$ is given by the heat kernel (2.1).

This is essentially equivalent to the approach taken by Aswani et al. [2011], who also prove that the wlPCA using the heat kernel is a consistent estimator of $T\mathcal{M}_p$.

Now that we have explained our approach, we summarize it in Algorithm 4 for computing (4.9) for fixed ϵ, λ, and manifold of intrinsic dimension d.

4.2.4 Algorithm complexity

We have already discussed the complexity of computing the Riemannian metric in Section 2.4.3. Even though the algorithm we propose here is based on the same idea, it is sufficiently distinct to warrant a discussion of some aspects of its complexity as well.

Using Algorithm 4 to compute the empirical distortion can be relatively expensive. Depending on the size of n and r, one of the most expensive steps is computing the n Local Weighted PCA in r dimensions, i.e. computing d-dimensional principal subspaces of r^2 matrices, which is $\mathcal{O}(r^2 d)$.

Algorithm 4 DISTORTION

Input: $\mathcal{P}$ as set of n points in $\mathbb{R}^r$, $\sqrt{\epsilon}$, dimension d, and metric type $\tau \in \{-1, 1\}$

Compute the weighted graph W using the heat kernel for each pair of points in $\mathcal{P}$

$L \leftarrow$ GRAPHLAPLACIAN$(W, \sqrt{\epsilon}, \lambda = 1)$

for $i = 1 \rightarrow n$ **do**

$Y \leftarrow$ TANGENTPLANEPROJECTION$(\mathcal{P}, W_{i,:}, d)$

$G \leftarrow$ LEARNMETRICPOINTWISE(Y, L, τ)

$\mathcal{D} \leftarrow \mathcal{D} + \frac{1}{n}||G - I_d||^2$ (where I_d is the $d \times d$ identity matrix)

end for

Return $\mathcal{D}$

Algorithm 5 TANGENTPLANEPROJECTION

Input: $\mathcal{P}$, weight vector w, dimension d

Construct $n \times r$ design matrix X from $\mathcal{P}$

Compute Z using (4.10)

$[V, \Lambda] \leftarrow \text{eig}(Z^t Z)$ (PCA of Z)

Center X around its weighted mean $\bar{x}_w$ from (4.10)

$Y \leftarrow X \cdot V_{:,1:d}$ (Projects X on d first principal components)

Return Y

Another expensive operation is to compute the Riemannian metric or its inverse. This involves computing the graph Laplacian, $\mathcal{O}(n^2)$, computing the dual Riemannian metric on every tangent plane individually, $\mathcal{O}(d^2n^2)$, and computing n inverses to obtain the Riemannian metric, $\mathcal{O}(d^3n)$. Note that to compute the dual Riemannian metric, we can use a sparse version of the graph Laplacian and hence reduce the complexity of the first two steps above to $\mathcal{O}(nk)$ and $\mathcal{O}(d^2nk)$.

If the intrinsic dimension is large, computing time could also be reduced by skipping the last step of Algorithm 6, which requires inverting G. Indeed, relation (4.2) is trivially satisfied for the dual Riemannian metric g_J^{-1} since $\delta_r|^{-1}_{T_x\mathcal{M}} = \delta_r|_{T_x\mathcal{M}}$. This,

Algorithm 6 LearnMetricPointwise

Input: $n \times d$ projected data Y, graph Laplacian L, and metric type $\tau \in \{-1, 1\}$

for $u = 1 \to d$ **do**

  for $v = 1 \to d$ **do**

    $G_{u,v} \leftarrow L \cdot (Y_{:,u} \circ Y_{:,v})$ ($\circ$ denotes element-wise product)

  end for

end for

Return G^{τ}

along with the continuity of the inverse, means that $g_{\jmath,n}^{-1} \to g_{\jmath}^{-1}$ if and only if $g_{\jmath,n} \to g_{\jmath}$. As such, we can reasonably expect that using $g_{\jmath,n}$ or $g_{\jmath,n}^{-1}$ in (4.9) will lead to similar estimated graph Laplacian parameters. In fact, using the dual Riemannian metric was found to work quite well in practice and often better than using the Riemannian metric. To incorporate this flexibility into Algorithm 4 , we introduced the parameter $\tau \in -1, 1$, which dictates whether (4.3) is computed with respect to $g_{\jmath,n}$ or $g_{\jmath,n}^{-1}$, respectively. The usefulness of relying on $g_{\jmath,n}^{-1}$, i.e. $\tau = 1$, to compute the distortion will be explored in more detail in Section 4.4.

As highlighted above, the cost of computing $g_{\jmath,n}$ depends on the intrinsic dimension d of $\mathcal{M}$. However, in the same way that using $g_{\jmath,n}^{-1}$ is less computationally intensive and performs better than using or $g_{\jmath,n}$, assuming a lower intrinsic dimension $d' \leq d$ can also be beneficial: it reduces computational complexity, and is more robust to boundary effects. We discuss this in further detail in Section 4.4.

Finally, another method for reducing computing time consists of computing the distortion (4.9) on a subsample; that is to say, selecting $\mathcal{P}' \subset \mathcal{P}$ with $|\mathcal{P}'| \ll |\mathcal{P}|$ and computing $\mathcal{D}(\mathcal{P}') = \sum_{x \in \mathcal{P}'} ||g_{\jmath,n}(x) - \delta_r|_{T_x\mathcal{M},n}||_{\delta_r|_{T_x\mathcal{M},n}}$ where $g_{\jmath,n}$ and $\delta_r|_{T_x\mathcal{M},n}$ can computed using all points in $\mathcal{P}$ to increase accuracy.

4.3 Related work

Building on the discussion in the previous section, we touch upon a few additional approaches. We note that, although the problem of estimating the "scale" of the data is pervasive in manifold learning, the work has mainly been focused on presenting asymptotic results, with very few papers proposing estimation methods that can be implemented in practice.

We have already discussed the asymptotic result (4.1) of Singer [2006]. Other work in this area (Giné and Koltchinskii [2006], Hein et al. [2007], Ting et al. [2010]) provides the necessary rates of change for ϵ with respect to n to guarantee convergence, and in the case of Giné and Koltchinskii [2006], gives an actual convergence rate. These studies are all relevant; however, they all depend on manifold parameters that are usually not known *a priori*. Finding a self-consistent method to estimate these parameters in tandem with estimating ϵ remains an open problem.

Some practical work has been done on the *nearest neighbor graphs* where the "scale" parameter to be estimated is k, the number of nearest neighbors to use in the construction of the graph Laplacian. The method of Chen and Buja [2009] is reminiscent of ours, in that it is self-concordant and evaluates a given k by how well an embedding using that k preserves the k' neighborhoods that exist in the original data. The paper offers a statistically principled way to compare this criterion over different k and k' values.

However, it is not known whether and how a method for estimating k can be translated into a method for estimating $\sqrt{\epsilon}$ or vice versa. The two most popular graph construction methods (the weighted graph with heat kernel and the k-nearest neighbors graph) exhibit different asymptotic behaviour precisely because they give rise to different ensembles of neighborhoods (Ting et al. [2010]).

There are other problems with the Chen and Buja [2009] method as well. The method is designed to optimize for a specific embedding algorithm, so the values ob-

tained for k change when one chooses different algorithms (for example, Local MDS or Isomap). In addition, in Chapter 2 we have shown that the embeddings themselves induce significant distortion in the data. Even if we are tempted to choose $\sqrt{\epsilon}$ using the Chen and Buja [2009] method, it is important to remember that neither Eigenmap nor Diffusion Maps, the two embedding algorithms that are constructed from the graph Laplacian - the object of interest - are isometric embeddings of the data. Even in the limit of infinite data, the embeddings will produce some reconstruction error, suggesting that the hypothesis according to which minimal reconstruction error leads to "optimal" embeddings might not hold in this case. For this reason alone, using reconstruction error seems ill-suited to estimating the optimal $\sqrt{\epsilon}$ to use for constructing $\tilde{\mathcal{L}}_{\epsilon,n}$.

It is also appropriate to mention the algorithm proposed in Chen et al. [2011] at this juncture. The goal of that body of work is to obtain an estimate of the intrinsic dimension of the data in a statistically principled way; however, a by-product of the algorithm is a range of scales where the tangent space at a data point p is well aligned with the principal subspace obtained by a local SVD centered at p. As these are scales at which the manifold looks locally linear, one can reasonably expect that they are also the correct scales at which to approximate differential operators, such as the Laplace-Beltrami. Given this possibility, we will implement the method and compare it to our own results.

4.4 Experimental Results

In proposing a new method for estimating the bandwidth of the graph Laplacian used in unsupervised and semi-supervised learning, we aim not only to provide a principled framework for doing so, but also to improve on the performance of previous methods. We use both real and synthetic data to illustrate the application of our method.

4.4.1 SSL Classification Tasks

In order to demonstrate the performance of our method, we first work with the six SSL datasets from Chapelle et al. [2006], which were described in detail in Section 3.5.1 in the previous chapter.

Methods Tested

Our first - and primary - task is to test the validity of our approach for estimating the bandwidth ($\sqrt{\epsilon}$) in all six SSL tasks. In addition to comparing the performance of Algorithm 4 to learning-based methods, we compare it to several variations of our method which correspond to alternative choices we might have made in designing Algorithm 4. The methods we evaluate are the following:

`CV` *Train using cross-validation* We use the standard procedure of learning $\sqrt{\epsilon}$ through cross-validation to set a benchmark against which to compare our performance. We split the training set (consisting of 100 points in all data sets) into two equal groups;[3] then, we use simulated annealing to minimize the highly non-linear cross-validation classification error. Since the data sets we use are designed to allow for 12 different test splits, we have 12 different values for $\sqrt{\epsilon}$

[3]To verify the sensitivity of this choice, we also attempted to subdivide the training set in groups of 5 and 20 depending on the data set. However, these alternate subdivisions did not materially affect the estimates of the hyperparameters, and were discarded to allow for faster computing time.

per data set. In Table 4.1 below, we report the mean, standard deviation and range of these 12 values for each data set.

`Rec` *Minimize the reconstruction error using the heat kernel* - A key characteristic of the manifold hypothesis is that the data is locally linear, and the scale at which the data is locally linear is that at which most manifold learning is performed (Aswani et al. [2011], Chen et al. [2011], Lee and Verleysen [2007]). A simple proxy can be used to find this scale without resorting to estimating the intrinsic dimension. Indeed, one can simply use the scale at which the data points are a weighted linear combination of their neighbors. In our case, since we are constructing the Laplacian from the heat kernel, we use that heat kernel as the basis for the weights. Specifically, we minimize the following distortion for ϵ:

$$\mathcal{R}(\epsilon) = \sum_{i=1}^{n} \left|\left| x_i - \sum_{j \neq i} \frac{w_\epsilon(x_i, x_j)}{\sum_{l \neq i} w_\epsilon(x_i, x_l)} x_j \right|\right|^2 ,$$

with w_ϵ again given by the heat kernel (2.1). The estimator we describe here is also used in Methods `Dist2` and `DualDist2` below to serve as the bandwith of the wlPCA.

`Dist` *Minimize distortion as in Algorithm 4* - For this - and all other methods based on Algorithm 4 - we only use one intrinsic dimension to compute the distortion. We discuss the impact of this choice in our results below.

`DualDist` *Minimize distortion with respect to the dual metric* - In Section 4.2.4, we noted that computing time could be reduced by skipping the last step of Algorithm 6, which requires inverting the dual metric, if the ambient space is large. We apply this method to our datasets in order assess its performance.

`Dist2` *Use Algorithm 4, but allow the Laplacian and the wlPCA to encode two different geometries* - In Section 4.2.3, we discussed the problem of defining the appropriate neighborhood size when performing the wlPCA to estimate the tangent

bundle $T\mathcal{M}$. Our solution was to equate the notion of neighborhood in the wlPCA with that embodied in the heat kernel, thus equating the two ϵ and ensuring that the geometry we encode is common to all the transformations we perform. In order to assess the impact of this solution, we try to apply Algorithm 4 *without* equating the two bandwidth; rather, we use `Rec` to estimate the ϵ for wlPCA.

`DualDist2` *Minimize distortion with respect to the dual metric and allow the Lapalcian and the wlPCA to encode two different geometries* - For the sake of completeness, we also combine Methods `DualDist` and `Dist2`.

`TE` *Minimize test error* - Of course, the "true" value of the bandwidth is unknowable for these data sets. In fact, one could argue that there might be more than one applicable "true" bandwidth: one for the unsupervised task of Laplacian-based dimensionality reduction, and one for the semi-supervised task of Laplacian-based regularization described in Chapter 3. Since we are using these data sets and associated classification tasks to evaluate our geometrically motivated method's performance in a semi-supervised learning setting, we take the "optimal" ϵ to refer to the one that leads to the best classification: that obtained by minimizing the test error. Just as in `CV`, since there are 12 different training sets, we will have what are effectively 12 overfitted estimates of ϵ, allowing us to report confidence regions for the "optimal" ϵ.

Results

Our estimates of the bandwidth $\sqrt{\epsilon}$ for each of the six data sets using each of the methods above are provided in Table 4.1 below. The first column of the table reports the "optimal" value of the bandwidth obtained by learning on the test errors.

Our performance in predicting the data following the application of the model

	TE	CV	Rec	Dist	DualDist	Dist2	DualDist2
Digit1	0.6715±0.0758 [0.5700, 0.7817]	0.8015±0.4493 [0.4745, 1.9919]	0.6428	0.7425	0.7440	0.8962	0.8329
USPS	1.2369±0.1483 [1.0406, 1.5854]	1.2370±0.8613 [0.5041, 3.1981]	1.6842	2.4236	1.0968	2.9069	2.4658
COIL	49.79±6.61 [42.82, 60.36]	69.65±31.16 [50.55, 148.96]	78.374	216.95	116.38	180.58	137.35
BCI	3.4734±3.1270 [1.2559, 8.9476]	3.2008±2.5025 [1.1986, 8.1909]	3.3150	3.1892	5.6060	5.5018	4.5092
g241c	8.3296± 2.4833 [6.3040, 14.6331]	8.7752±3.3651 [4.4217, 14.8979]	3.7883	7.3720	7.3823	7.1580	7.0569
g241d	5.7527± 0.2440 [5.6482, 6.2751]	6.4638±1.1489 [4.2934, 8.1885]	3.7709	7.3522	7.3579	7.1511	7.0396

Table 4.1: Estimates of $\sqrt{\epsilon}$ obtained using the six methods presented for the six SSL data sets used, as well as TE. For TE and CV, which relied on the use of 12 training sets, we report the average value along with its standard error, as well as the range (in brackets below).

	CV	Rec	Dist	DualDist	Dist2	DualDist2
Digit1	3.32	2.16	(2.11)	2.11	3.13	2.58
USPS	5.18	4.83	12.00	3.89	19.62	12.39
COIL	7.02	8.03	16.31	8.81	14.64	11.24
BCI	49.22	49.17	50.25	48.67	47.89	49.06
g241c	13.31	23.93	(12.77)	12.77	12.88	12.85
g241d	8.67	18.39	8.76	8.76	8.66	8.52

Table 4.2: Percent classification error for the six SSL datasets using the seven $\sqrt{\epsilon}$ estimation methods. The best results for each data set are highligted in red. Parentheses indicate an extrapolated result: if the value of the $\sqrt{\epsilon}$ did not differ by more than 1%, we did not re-run the experiment.

(3.35) with κ^2 set to zero[4] is reported in Table 4.2, where the values represent percent classification error.

The first observation that can be drawn from the results is that, across all methods and data sets, the estimate of the bandwidth that was furthest away from the "optimal" value led to the highest classification error. In data set Digit1, this "honor" went to methods CV and Dist2; in USPS, it went to Dist2; in COIL, it went to Dist and Dist2; and in g241c and g241d, it went to method Rec. This confirms that performance in classification when using a Laplacian-based regularizer is quite sensitive to the estimate of the bandwidth of the Laplacian and lends legitimacy to our attempt at finding a better, more principled method for doing so.

Across five of the six data sets, cross-validation (CV) did not perform as well as the geometry-based methods. Further, the estimates of $\sqrt{\epsilon}$ in each of the 12 training

[4]The parameters that were found to contribute most to reducing classification error in our six data sets were μ and α. Therefore, to limit the risk of overfitting while learning ϵ via cross-validation, we set κ^2 to zero.

sets within a data set were highly variable, with standard errors often of the same order as the estimated values themselves. This confirms that cross-validation tends to overfit rather than find values that generalize well.

The one exception was the `COIL` data set, where, despite their variability, cross-validation estimates still outperformed the geometrically-based methods. It is interesting to note that this is the only data set constructed from a collection of manifolds - in this case, 24 one-dimensional image rotations. As such, one would expect that there would be more than one natural length scale. As the top two panels of Figure 4.1 demonstrate, that is indeed the case. The top left panel shows the relationship between the distortion D at each point and the $\hat{\epsilon}$, and, when observed in contrast with the same graph for `Digit1`, makes it clear that there is more than one relevant length scale in the data. Indeed, the top right panel shows the four clusters we have been able to identify. It is interesting to note that one of the scales (the cluster depicted in blue) happens to match the length scale of the process to be learned; however, it is drowned by at least three other length scales,[5] which lead to functions that are more difficult to minimize and yield the wrong value. In red, we portray the average of all clusters, showing the distortion function our method is likely working with. The minimum of this average function is not very well defined, and leads to the wrong value. In practice, to deal with a multiscale problem would require us to perform cross-validation or some other method of model selection over the four potential length scales observed; however, that remains significantly simpler than performing an exhaustive search, and is also less likely to lead to overfitting.

It is quite possible that the geometrical approach finds an unhappy middle ground, while cross-validation has an easier time identifying the length scale appropriate for the classification problem, even if there is evidence of some overfitting due to the

[5]We did not conduct an analysis of the number of clusters present, as this would be beyond the scope of our work. Rather, we considered the number of clusters that appear to produce clearly distinguishable landscales.

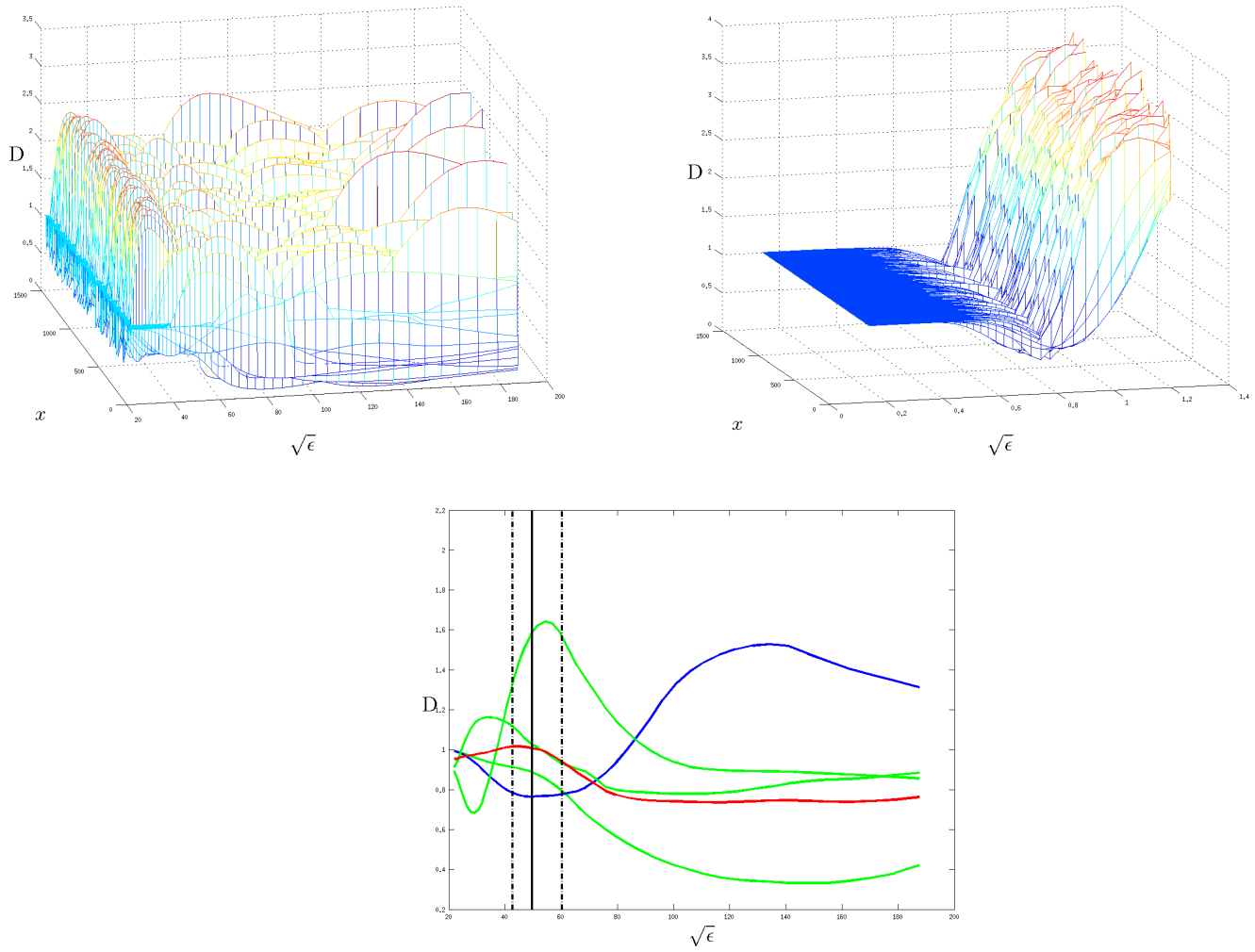


Figure 4.1: Top panels: distortion D plotted against $\sqrt{\epsilon}$ for each point x_i in the `COIL` (left) and `Digit1` (right) data sets. In the bottom, we observe the length scale clusters in the `COIL` data, with the blue being closest to the length scale of the process to be learned, the green being other observed clusters, and the red being the average of all observed clusters. The black vertical line indicates the average "optimal" value of $\sqrt{\epsilon}$, and the dotted black vertical lines indicate its range. In all cases, d is known to be 1.

variability of the estimator. It is worth pointing out that this is the only data set where the classifier trained on the test error is an order of magnitude better than the best "fair" classifier: it outperforms it by 0.73%. For all the other data sets, training on the test error leads to an improvement of between 15% and 30%.

Performing cross-validation was found to take between 2 to 6 times longer than using a pre-set value for the bandwidth learned via another method[6]. This limits the usefulness of cross-validation, particularly in large data sets.

Perhaps not surprisingly since it was not designed specifically for the Laplacian, the reconstruction error method (`Rec`) was found to perform poorly overall. Even in the one case where it found a value closest to the "optimal" value[7], it did not lead to the lowest classification error.

The geometry-based methods that aimed to minimize the distortion (4.9) led to varying degrees of success in estimating $\sqrt{\epsilon}$ and in the classification task. The method that performed best overall was `DualDist`, which used the dual metric and that equated the wlPCA neighborhood with that of the Laplacian itself, especially when used with only one intrinsic dimension. The reason why using the dual metric outperformed the straightforward application of Algorithm 4 is that it is easier to optimize and less likely to settle on a local minimum and yield the wrong metric, as explained in further detail in paragraph 4.4.1.

Overall, our results show that estimating the bandwidth parameter in a principled, geometry-based way, using Algorithm 4, shows improvement in performance over cross-validation and is relatively straightforward to use.

Special Case: Learning Task on `BCI` *data*

As is easily apparent from Table 4.2, the largest classification errors by far were obtained when working with the `BCI` data. We are not the first to experience this

[6]This includes the time required to compute $\hat{\epsilon}$.

[7]The bandwidth learned on the test error.

difficulty. For example, Zhou and Belkin [2011] report a rather disappointing classification error of 44% when using the Iterated Laplacian method presented in their paper, which relies on a squared error loss function rather than the usual hinge or 0-1 loss. Though 44% is not a stellar figure to strive for, we were nevertheless unable to reproduce these results. Indeed, applying Algorithm 4 to estimate the ϵ or using cross-validation as suggested in Zhou and Belkin [2011] and then using the Iterated Laplacian method with a squared error loss, we achieved a classification error of 48%, which suggests that we did little better than a coin toss.

Our lackluster performance and our inability to reproduce the results of Zhou and Belkin [2011] led us to examine exactly how much *could* be learned through their method on this data set. To do so, we randomized the class labels attached to the data, ensuring that any existing correlation between the predictors and the response is removed. We then learned the data through both cross-validation and through minimizing test errors. Through cross-validation, we obtained a classification error of 48%; and when minimizing test errors, we obtained a classification error of 42%. Having obtained this result on the non-correlated data, we are lead to conclude that learning on the test errors led to overfitting, and that there was actually nothing of value to be learned through the Zhou and Belkin [2011] method on this data set. When we employed a GP prior on logistic regression, a method we described in the previous Chapter, we obtained a slightly better result (see Table 3.1). This suggests that the main problem in the application of our method and that of Zhou and Belkin [2011] is the choice of squared error loss in a classification context.

Effect of Dimension

One of the inputs required for computing the distortion of 4.9 is the intrinsic dimension of the data manifold, d. In most cases, this is not known, and we do not offer a new method for estimating it. Rather, we argue that using one or a very few dimensions yields very good results. We have already discussed how using fewer dimensions can

help render the estimate more robust to the boundary effect. In this Section, we report on how changing the intrinsic dimension used alters our estimate of ϵ based on the same data sets discussed above.

Our results are summarized in Table 4.3. We note that the estimates of ϵ are quite similar, regardless of the number of dimensions used. In the case of `Digit1`, we know that the "true" number of dimensions is 5. However, the difference between the estimate of ϵ obtained using one dimension and that using all five dimensions is only 0.419. It is also worth noting that even for `g241c` and `g241d`, which were constructed so as to not satisfy the manifold hypothesis, our method does reasonably well at estimating ϵ. That is, our method finds the ϵ for which the Laplacian encodes the geometry of the data set irrespective of whether or not that geometry is lower-dimensional.

Overall, we have found that using only one dimension is most stable, and that adding more dimensions introduces more numerical problems: it becomes more difficult to optimize the distortion as in 4.7, as the minimum becomes shallower. Figure 4.2 illustrates this point nicely, for the two methods DualDist and Dist with data set `Digit1`. In our experience, this is due to the increase in variance associated with adding more dimensions. Using one dimension probably works well because the wlPCA selects the dimension that explains the most variance and hence is the closest to linear over the scale considered. Subsequently, the wlPCA moves to incrementally "shorter" or less linear dimensions, leading to more variance in the estimate of the tangent plane.

Figure 4.2 also illustrates the reason why using the dual metric works better in practice than inverting the dual metric: it leads to a distortion function that is much easier to optimize. This behavior is largely due to what happens at large values of $\sqrt{\epsilon}$. At these values, the geometry converges to the degenerate case of the single point, for which $||g|| \to 0$ (there are no longer any distances to be measured). This means that, when $\delta_r|_T\mathcal{M}$ is compared to g, the result is simply the 0 matrix minus the identity

		d=1	d=2	d=3	d=5
Digit1	Dist	0.743	0.293^8	0.305^8	1.162
	DualDist	0.744	0.767	0.781	0.797
USPS	Dist	2.424	2.306	3.877	–
	DualDist	1.097	1.156	1.177	–
COIL	Dist	7.372	9.878	9.373	–
	DualDist	7.382	2.846^8	2.846^8	–
BCI	Dist	7.372	9.878	9.373	–
	DualDist	7.382	2.846^8	2.846^8	–
g241c	Dist	7.372	9.878	9.373	–
	DualDist	7.382	2.846^8	2.846^8	–
g241d	Dist	7.352	9.326	9.983	–
	DualDist	7.358	2.833^8	2.833^8	–

Table 4.3: $\sqrt{\epsilon}$ obtained using various d values with the Dist and DualDist methods on all six data sets. $\sqrt{\epsilon}$ was computed for d=5 for the Digit1 dataset, as it is known to have an intrinsic dimension of 5.

matrix, which is just the norm of the identity matrix in d dimensions. Therefore, the distortion converges to a small value as $\epsilon \to \infty$, and for finite samples, this value may be even smaller than the one resulting from the use of the optimal ϵ. In contrast, when $||g|| \to 0$, the dual metric $||g^*|| \to \infty$, so the computed distortion from $\delta_r|T\mathcal{M}$ is going to be very high even for finite samples.

In Figure 4.2, we see that when inverting the dual metric for $d > 1$, there exists the additional problem of the local minimum, which is located between $\sqrt{\epsilon} = 0.2$ and $\sqrt{\epsilon} = 0.3$ in the `Digit1` dataset. Indeed, $\sqrt{\epsilon}$ estimates using `Dist` with $d = 2$ and $d = 3$ never move on from that minimum, and produce suboptimal values for $\sqrt{\epsilon}$. All occasions in which this happens in other data sets are highlighted with footnote [8] in Table 4.3.

4.4.2 Synthetic Data with Noise

We also experimented with estimating ϵ on synthetic data, sampled from known manifolds with noise. We considered the two-dimensional `hourglass` manifold of Figure 4.3. We sampled uniformly from this manifold, adding 10 "noise" dimensions and adding Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the resulting 13 dimensions. We also considered a second manifold, the `dome` (see Figure 4.3).

The range of ϵ values was delimited by ϵ_{min} and ϵ_{max}. We set ϵ_{max} to the average of $||x_i - x_j||^2$ over all point pairs, a scale that is surely larger than the local neighborhood for any manifold with $n >> d$. The lower limit ϵ_{min} is the limit in which the heat kernel K becomes approximately equal to the unit matrix; this is tested by $\max_j(\sum_i K_{ij}) - 1 < \zeta$[9] for $\zeta \approx 10^{-4}$, where ζ is the tolerance parameter. This range spans about two orders of magnitude in the data we considered, and was searched by a logarithmic grid with approximately 20 points. This is because the method in Chen et al. [2011] -

[8] Local minimum.

[9] This implies that all eigenvalues of K are less than ζ away from 1 by the Gershgorin circle theorem.

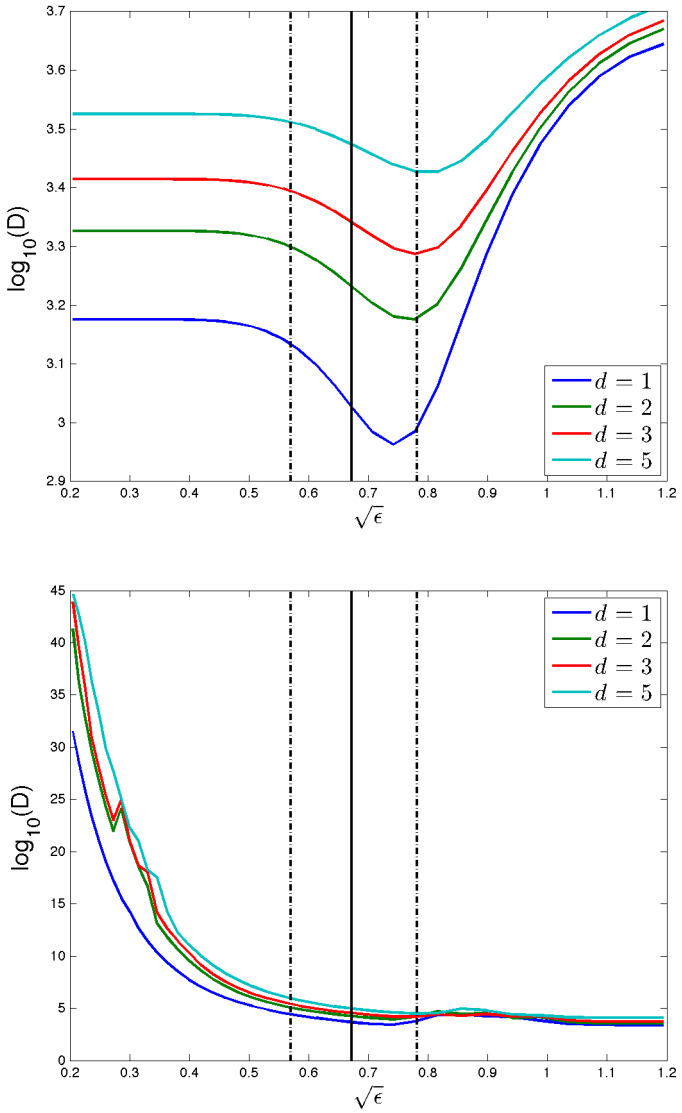


Figure 4.2: Empirical distortion (logarithmic scale) and associated estimates of $\sqrt{\epsilon}$ produced with the DualDist method (top) and the Dist method (bottom) in data set `Digit1`, with varying intrinsic dimensions (d=1 to d=5). The vertical black lines represent the average estimate of $\sqrt{\epsilon}$ obtained using TE (the "optimal" value for $\sqrt{\epsilon}$) (full line) and its range (dotted lines).

which we explain below and to which we compare our results - requires searching the range exhaustively. For our method, we observed that a line search without derivatives starting from ϵ_{min} takes fewer evaluations and returns results that are more precise than using the grid search. We also decreased our computation time by evaluating all pointwise quantities ($\hat{\mathcal{D}}$, local singular value decomposition) on a random sample of size $n' = 200$ of each data set. We replicated each experiment on 10 independent samples.

Effects of d', noise and n.

The estimated values for $\hat{\epsilon}$, with error bars over 10 replications are presented in Figure 4.3. As mentioned before, one could choose to optimize the distortion in any number of dimensions d' not exceeding the intrinsic dimension d. Let $\epsilon_{d'}$ denote the estimate obtained from a d' dimensional metric matching. We note a few interesting things. First, when $d_1 < d_2$, typically $\epsilon_{d_1} > \epsilon_{d_2}$, but the values are of the same order (a ratio of about 2 in the synthetic experiments). The explanation is that, at ϵ values near the optimal one, choosing $d' < d$ directions in the tangent plane will select a subspace aligned with the "least curvature" directions of the manifold by virtue of employing wlPCA to estimate the sub tangent plane, if any exist, or with the "least noise" in the random sample. In these directions, which by construction are close to linear, the data will generally tolerate slightly more smoothing, which results in larger $\hat{\epsilon}$.

The variance of $\hat{\epsilon}$ observed is due to randomness in the subsample n' used to evaluate the distortion. The `hourglass` is a manifold with variable curvature, and the distortions will differ depending on the curvature at the location of the n' points. The optimal ϵ decreases with n and grows with the noise levels, reflecting the balance it must find between variance and bias. Note that for the `hourglass` data, the highest noise level of $\sigma = 0.1$ is an extreme case, where the original manifold is almost drowned in the 13-dimensional noise. Hence, ϵ is not only commensurately larger, but also stable between the two dimensions and runs. This reflects the fact that ϵ captures the

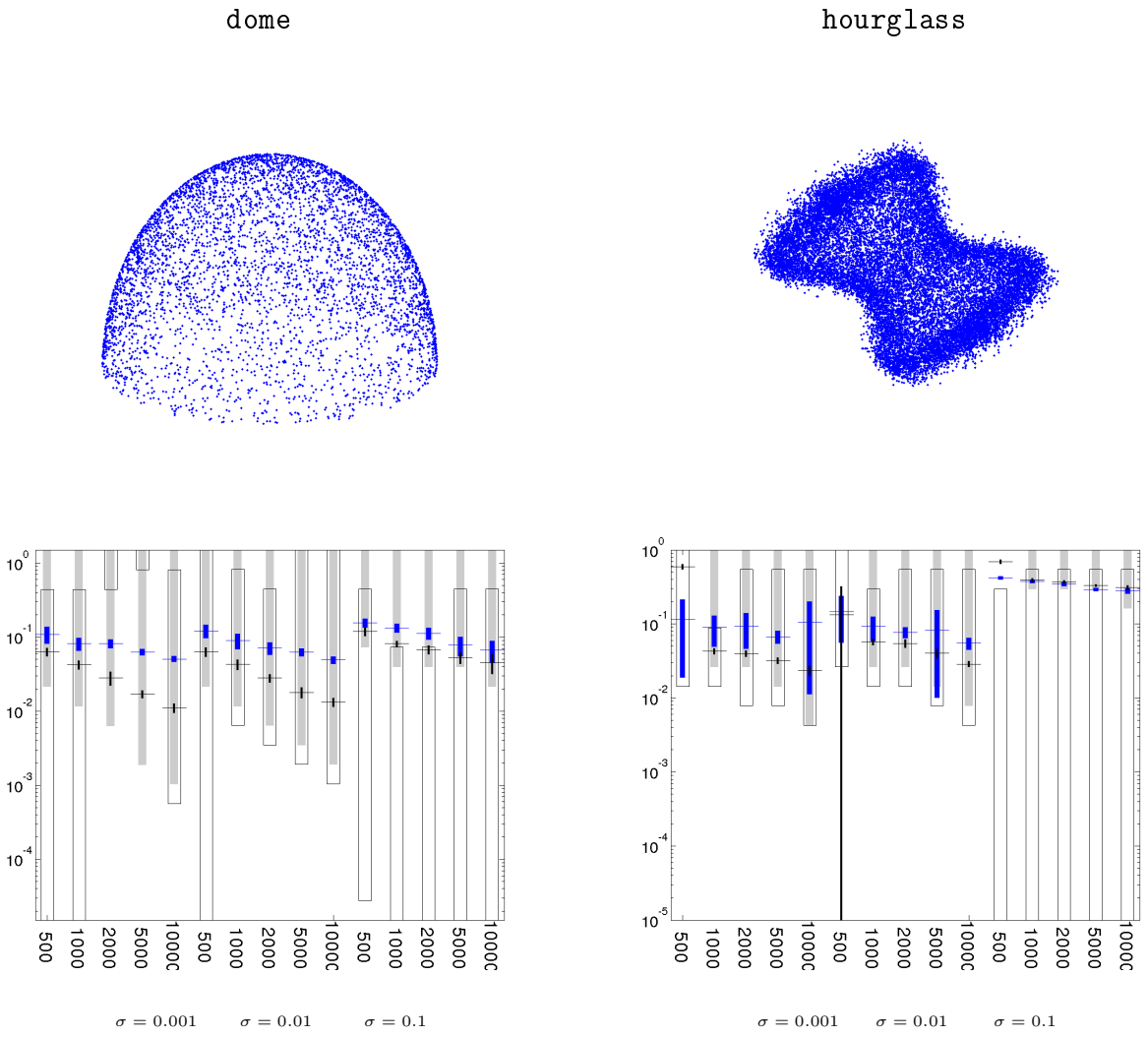


Figure 4.3: Estimates $\hat{\epsilon}$ (vertical axis, marked by $-$, blue for $d = 1$ and black for $d = 2$, with $\pm$ standard deviations as small vertical bars) on the `dome` and `hourglass` data, for various sample sizes n and noise levels σ (shown on the horizontal axis). We also show, for each n, σ, the intervals in ϵ where the eigengaps of local SVD indicate the true dimension (see Chen et al. [2011]) as gray rectangles, and the estimates proposed by Chen et al. [2011] for these intervals as unfilled rectangles.

noise dimensions, and its values are indeed just below the noise amplitude of $0.1\sqrt{13}$. The `dome` data set exhibits the same properties discussed previously, showing that our method is effective even for manifolds with boundary effects.

Boundary effects and scaling.

We consider the scaling of $\hat{\epsilon}$ with n for the two manifolds and compare it to equation (4.1), which, we remind the reader, obtains the optimal rate of decay of $\hat{\epsilon}$ with sample size, such that the variance-bias trade-off is optimized. It predicts that $\log n$ regressed on $\log \frac{1}{\epsilon}$ is a line with slope $\alpha = 3 + \frac{d}{2}$.

We calculated these slopes[10], obtaining the following d estimates

Noise level σ	0	0.01	0.01	0.1	
`hourglass`	2.1	1.9	0.6	12.9	
`dome` $d' = 2$	-2.3	-2.5	-2.2	0.4	(border effect)
`dome` $d' = 1$	1.4	2.3	1.2	0.8	(better results for 1D projection)

If one looks at the low noise regime, the d estimates are fairly noisy[11] for the `hourglass` data, giving a qualitative but beautiful confirmation of the agreement of our method with theory. When the noise is large, the "estimated" dimension grows toward the dimension of the noise, which is 13. However, on the `dome` data the estimates behave very differently. This is the boundary effect at play, which we discuss in more detail below.

$\hat{\epsilon}$ and estimating d using Chen et al. [2011]

Could $\hat{\epsilon}$ be used to improve the estimation of the intrinsic dimension d as suggested in Chen et al. [2011]? The method of estimating the intrinsic dimension d in Chen et al. [2011] has two components: first, it performs local SVD around each data point

[10] For `hourglass` excluding the high-variance $n = 500$.

[11] Estimates of slope are known to be sensitive to outliers, and d itself is very sensitive to the error in the slope. Thus, we do not propose this as a method for estimating d.

at a variety of scales ϵ (this is akin to our weighted tangent plane projections); then, it finds a range of scales in ϵ space, which we shall call the CMLR range (Chen et al. [2011]), where the largest eigengap is the d-th eigengap. The d is estimated by finding the largest eigengap somewhere in the CMLR range.

We computed the CMLR ranges both using the method of Chen et al. [2011] (unfilled rectangles in Figure 4.3) and the ground truth ranges (grey rectangles). As can be seen, our $\hat{\epsilon}$ estimate *always* lies within the true ranges, meaning that if we computed the eigengaps at $\hat{\epsilon}$, we would find the true d, provided that such a range exists. See Chen et al. [2011] for a more detailed discussion of the limitations of this method in e.g. high-dimensional noise. In contrast, the CMLR ranges tend to underestimate the true ranges, and sometimes they do not overlap with them. We also found that the method in Chen et al. [2011], which is based on finding the "first descents" of the singular values, can be unreliable in that it may not find an upper or a lower limit to the interval. Note that the method in Chen et al. [2011] depends on a parameter K to be set by the user, and we gave it the optimal K for these data.

4.4.3 Large Sample Synthetic Data without Noise

To complete the analysis on the effect of the dimension used in computing the empirical distortion, we also compare the convergence rate of $\hat{\epsilon}$ with 1 or 2 dimensions, without noise or subsampling, for large sample sizes.

Figure 4.4 shows our estimates $\hat{\epsilon}$ as the sample size n increases from 500 to 15,000 on the two manifolds `hourglass` and `dome`. The average and standard deviation of $\hat{\epsilon}$ using $d = 1$ is shown in blue, while the average and standard deviation of those using $d = 2$ is shown in black. The red line shows the Singer optimal rate of decline. Note that while the values of $\hat{\epsilon}$ using $d = 1$ and $d = 2$ should be compared to each other, the value of $\hat{\epsilon}$ given by the Singer line is only correct up to an unknown constant which depends on the manifold used, and so should not be used as an absolute reference. Rather, the Singer line only serves as a reference for the optimal rate of convergence.

In all cases, we see that the $\hat{\epsilon}$ computed using $d = 1$ follow the Singer line more closely than the estimates computed using $d = 2$. In the case of the `hourglass`, we observe the effect of curvature discussed previously: $\hat{\epsilon}$ using $d = 2$ is mostly smaller than that using $d = 1$.[12]

In the case of the `hourglass` and the `dome`, we observe that using $d = 2$ produces lower estimates of ϵ and converges faster than the Singer optimal rate. The explanation for this lies in part in Proposition 10 of Coifman and Lafon [2006], who note that, while the bias of the graph Laplacian is usually proportional to ϵ, it is proportional to ϵ^{γ} at the boundary, where $\gamma \in [0, 0.5]$. Thus, to reduce the bias due to the boundary, our method selects a smaller ϵ, and strikes a different variance-bias trade-off. Indeed, since the Singer rate is computed without taking into account the presence of a boundary, the variance-bias trade-off should be optimized by a different optimal bandwidth in that case.

However, $\hat{\epsilon}$ does not appear to be nearly as affected by the presence of a boundary when using $d = 1$. This is because, with $d = 1$, wlPCA will find a one-dimensional subspace that is tangential to the boundary. On that one-dimensional subspace, the graph Laplacian should have the usual ϵ bias and lead to the rate of convergence expected by Singer [2006].

The behavior of the $\hat{\epsilon}$ estimates with $d = 1$ and $d = 2$ in the `hourglass` and the `dome` cases can be contrasted to their behavior in the `sphere` case, which presents no boundary or curvature effects. In the case of the `sphere`, the dimensions used seem to have almost no impact on the rate of convergence of the $\hat{\epsilon}$ estimate, which is only slightly faster than the Singer rate.

Our comparison of the behavior of $\hat{\epsilon}$ when using $d = 1$ and $d = 2$ in the presence of a boundary should naturally generalize to higher-dimensional manifolds (i.e. d-dimensional manifolds with $d - 1$ dimensional boundaries). As such, using fewer

[12] $\hat{\epsilon}$ using $d = 2$ appears to be higher than that using $d = 1$ at small sample sizes. However, their confidence regions overlap.

dimensions to compute the distortion should be more robust to boundary effect; in fact, one could use the number of dimensions in play to determine whether the estimated Laplacian should be dominated by the boundary or the interior of the manifold, where the most interesting "information" is expected to lie.

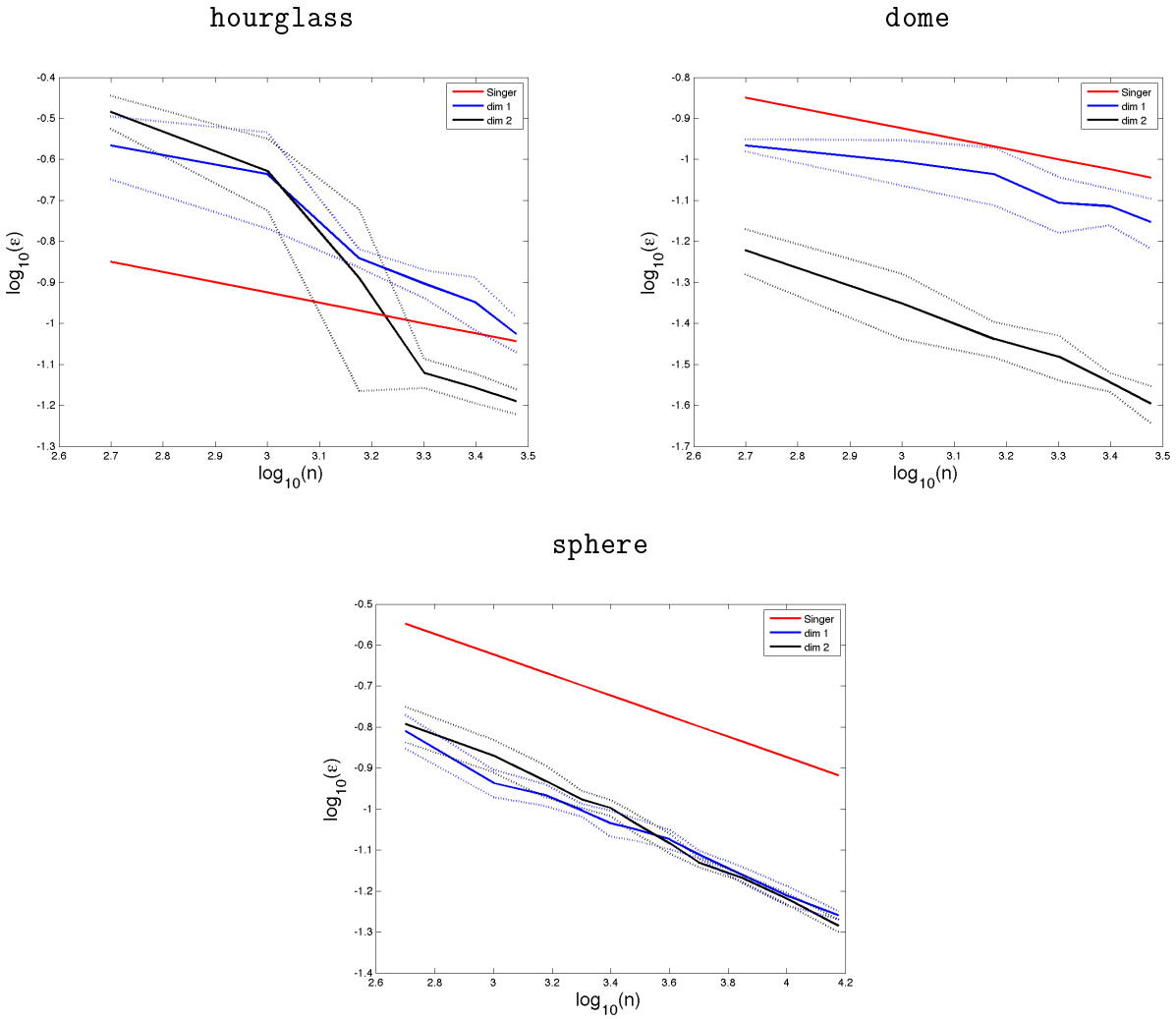


Figure 4.4: $\hat{\epsilon}$ (vertical axis, logarithmic scale) obtained using 1 (blue) and 2 (black) d on the `hourglass` (top left), the `dome` (top right), and the `sphere` (bottom) manifold, compared to the Singer optimal ϵ decay rate (red) over sample sizes ranging from $n = 500$ to $n = 15,000$ (horizontal axis, logarithmic scale). Each experiment was repeated 5 times.

4.5 Discussion

In manifold learning, semi-supervised and unsupervised, estimating the graph Laplacian is a common problem. We have presented a novel and principled method of selecting the bandwidth $\sqrt{\epsilon}$ of graph Laplacians constructed from weighted kernel graphs in order to produce an improved estimate of the Laplace-Beltrami operator on the data manifold $\mathcal{M}$. Aside from producing a graph Laplacian that is geometrically faithful to the original data, we also observed that out method leads to better classification and regression accuracy on benchmark data.

Our method is based on imposing geometrical self-consistency of the graph Laplacian. As such, the method is fully unsupervised, aiming to optimize a geometrical representation of the data, rather than performance in a particular task.

We verified experimentally that our $\sqrt{\epsilon}$ estimates agree with asymptotic manifold theory when noise and border assumptions are not violated. However, the interest of our method lies in the fact that asymptotic results are generally only defined up to an unknown constant; even then, for finite n, a small departure from the asymptotic prescriptions may be beneficial. Indeed, our experiments demonstrate that our algorithm is performant in variety of possible tasks, including embedding of noisy data (non-linear dimension reduction proper), and semi-supervised learning.

Possible Extensions

Although we do not provide a proof the convergence of the empirical distortion, we expect that, as $n \to \infty$, it will converge to

$$\hat{D}\left(g_{\jmath,n}, \delta_r|_{T_x\mathcal{M},n}\right) \;\to\; \int_{\mathcal{M}} ||g_\jmath(x) - \delta_r|_{T_x\mathcal{M}}||^2_{\delta_r|_{T_x\mathcal{M}}}\, p(x)\sqrt{|\delta_r|_{T_x\mathcal{M}}|}dx\,, \quad (4.11)$$

where $\sqrt{|\delta_r|_{T_x\mathcal{M}}|}dx$ is the volume element of the inclusion map of $\mathcal{M}$ in $\mathbb{R}^r$ at point x, and $p(x)$ is the sampling density on $\mathcal{M}$. Formalizing this proof is a natural first extension of the work presented here.

We conclude by referring to the the problem of estimating the intrinsic dimension d (Levina and Bickel [2005], Carter et al. [2007]). We do not explicitly deal with this estimation problem; rather, we note that, when it comes to finding the optimal ϵ using the method we propose, knowing d does not appear to be a prerequisite. In fact, empirically, we have found that using one or just a few dimensions is sufficient, if not beneficial. This also reduces the computational cost of Algorithm 6 and makes the optimization task easier. Once again, however, there is an opportunity to expand upon our method by addressing the complex task of estimating the intrinsic dimension d.

Chapter 5

CONCLUSIONS

In this work, we have shown - first - that there is a equivalence between the Laplace-Beltrami operator and the data geometry, and - second - that exploiting this relationship can improve our understanding and our handling of data manifolds in many settings: dimensionality reduction, parameter estimation, and classification and regression tasks. Throughout, we demonstrate the advantages of preserving the information contained in the geometry of the data and of ensuring that any tasks performed on the data are consistent with their original geometry.

In Chapter 2, we focused on geometrical consistency in unsupervised learning - specifically, in the task of embedding a high-dimensional set of data into lower dimensions. We proposed an algorithm that, in a novel step, estimates the embedding metric h, which establishes a direct correspondence between geometric computations performed using the embedding $(f(\mathcal{M}), h)$ and those performed directly on the original geometry $(\mathcal{M}, g)$ for any embedding f.

This new perspective allows a user to 1) visualize how each embedding algorithm distorts the original geometry at every point of the manifold; 2) measure the local distortion caused by the embedding algorithm; and 3) estimate geometric quantities of interest (distances, areas, volumes) on the output of any embedding algorithm while accounting for the distortions the algorithm imposed on the original geometry.

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, our method has the capacity to fundamentally change the way non-linear dimension reduction is carried out in practice.

In Chapter 3, we exploited the fact that SPDE's based on the Laplace-Beltrami operator can be used to define the popular Matérn class Gaussian Processes on the manifold to introduce geometrically motivated kernel regularization. In so doing, we

unify some of the techniques that have been developed independently in the arenas of semi-supervised and supervised learning, and introduce the Matérn covariance class to semi-supervised learning, showing that it allows for a much more elegant and useful understanding and interpretation of the parameters involved. In other words, our main contribution in this chapter is to define a covariance function over a manifold of unknown geometry from a finite sample that is both interpretable and flexible.

Finally, in Chapter 4, we exploit the correspondence between the Laplace-Beltrami operator and the geometry of the manifold to find a new, principled method for estimating the bandwidth of the graph Laplacian - one of the key hyperparameters needed to ensure the graph Laplacian is a good approximation of the Laplace-Beltrami operator that are not easy to learn by cross-validation or other model selection approaches. Improving on the current technique of simply "tuning" the bandwidth of the graph Laplacian, we presented a new approach that - in keeping with our overall theme in this work - is based on the faithfulness to the geometry of the original data. Our approach consists of choosing the bandwidth parameter that yields the graph Laplacian that is *closest* to the geometry of the data. The experiments we conduct show that our estimate of the bandwidth yields to improvements in the performance of regression and classification methods that consume the parameter, all the while reducing running time.

The results and discussion we present here lend themselves to opportunities for further study in three main areas. First, there is an opportunity to explore the effect of noise on the estimated Riemannian metric. Our approach does not account for situations where the data consists of a noisy representation of the manifold and does not lie directly on it. In such cases, we can only identify the noise component in the data if the embedding algorithm we use explicitly accounts for it. However, even if the embedding algorithm successfully identifies the noise component, this information does not permeate through the graph Laplacian, since its construction is independent of the embedding algorithm. By extension, the estimated Riemannian metric will

also be affected by any noise in the data. Thus, the challenge is to find a method for constructing the graph Laplacian that is more robust to noise. One possibility is to estimate the graph Laplacian using weighted local PCA using the method described in Section 4.2.3.

Second, Chapter 3 could be extended by devising a method for constructing new covariance functions directly from the Riemanninan metric instead of relying on the Laplace-Beltrami operator. Indeed, one could exploit the fact that the Riemannian metric can be used to integrate functions on oriented manifolds to expand the set of reproducing kernel Hilbert Spaces that we can construct on the data manifold.

A final topic for further exploration is the extension of the geometrical consistency approach developed in Chapter 4 beyond its current use in estimating the bandwidth parameter of the graph Laplacian. Indeed, this approach could be used to estimate other parameters or geometrical features of the data manifold in a purely unsupervised fashion.

BIBLIOGRAPHY

R. J. Adler. *The Geometry of Random Fields*. Wiler, 1981.

R. Andersen, F. R. K. Chung, and K. J. Lang. Local partitioning for directed graphs using pagerank. In *Lecture Notes in Computer Science*, volume 4863, pages 166–178, 2007.

N. Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68:337–404, 1950.

A. Aswani, P. Bickel, and C. Tomlin. Regression on manifolds: Estimation of the exterior derivative. *Annals of Statistics*, 39(1):48–81, 2011.

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

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

M. Belkin and P. Niyogi. Convergence of laplacians eigenmaps. In *Advances in Neural Information Processing Systems (NIPS)*, 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, 2006.

M. Belkin, J. Sun, and Y. Wang. Constructing laplace operator from point clouds in rd. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1031–1040, 2009.

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

P. Bérard, G. Besson, and S. Gallot. Embedding riemannian manifolds by their heat kernel. *Geometric and Functional Analysis*, 4(4), 1994.

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

I. Borg and P. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. 2nd edition, 2005.

O. Bousquet, S. Boucheron, and G. Lugosi. Introduction to statistical learning theory. In *Lecture Notes in Computer Science*, volume 3176, pages 169–207, 2004.

M. 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.

K. Carter, A. Hero, and R. Raich. De-biasing for intrinsic dimension estimation. pages 601–605, 2007.

O. Chapelle, B. Schölkopf, A. Zien, and editors. *Semi-Supervised Learning*. the MIT Press, 2006. URL `http://www.kyb.tuebingen.mpg.de/ssl-book`.

G. Chen, A. Little, M. Maggioni, and L. Rosasco. Some recent advances in multiscale geometric analysis of point clouds. In *Wavelets and Multiscale Analysis: Theory and Applications*, Applied and Numerical Harmonic Analysis, pages 199–225. Springer, 2011.

L. Chen and A. Buja. Local Multidimensional Scaling for nonlinear dimension reduction, graph drawing and proximity analysis. *Journal of the American Statistical Association*, 104(485):209–219, 2009.

W. Cheney. *Analysis for Applied Mathematics*. Springer, 2001.

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

R. R. Coifman, S. Lafon, A. Lee, Maggioni, Warner, and Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. In *Proceedings of the National Academy of Sciences*, pages 7426–7431, 2005.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118:69 – 113, 2000.

D. L. Donoho and C. Grimes. Hessian eigenmaps: New locally-linear embedding techniques for high-dimensional data. Technical report, Stanford University Department of Statistics, 2003a.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. In *Proceedings of the National Academy of Sciences*, volume 100, pages 5591–5596, 2003b.

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, 2007, retrieved June 2010. URL `http://www.math.colostate.edu/~thompson/whit_embed.pdf`.

L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 2nd edition, 2010.

D. Gibson, J. M. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, pages 225–234, 1998.

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

T. Gneiting. Simple tests for the validity of correlation function models on the circle. *Statistics and Probability Letters*, 38:119–122, 1998.

Y. Goldberg and Y. Ritov. Local procrustes for manifold embedding: a measure of embedding quality and embedding algorithms. *Machine Learning*, 77(1):1–25, 2009.

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

A. G. Gray and A. W. Moore. N-body problems in statistical learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

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

M. Hein and J.-Y. Audibert. Intrinsic dimensionality estimation of submanifolds in $\mathbb{R}^d$. In *International Conference on Machine Learning*, 2005.

M. Hein and O. Bousquet. Kernels, associated structures and generalization. Technical report, Max Planck Institute for Biological Cybernetics, 2004.

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.

G. Henry and D. Rodriguez. Robust nonparametric regression on riemannian manifolds. *Journal of Nonparametric Statistics*, 21(5):611–628, 2009.

P. Hoff, A. Raftery, and M. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098, 2002.

A. Jafari and F. Almasganj. Using laplacian eigenmaps latent variable model and manifold learning to improve speech recognition accuracy. *Speech Communication*, 52:725–735, 2010.

T. M. Lal, M. Schröder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, and B. Schölkopf. Support vector channel selection in bci. *IEEE Transactions on Biomedical Engineering*, 51(6):1003–1010, 2004.

J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer Publishing Company, Incorporated, 1st edition, 2007.

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

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

E. Levina and P. Bickel. Maximum likelihood estimation of intrinsic dimension. 2005.

B. Lin, C. Zhang, and X. He. Semi-supervised regression via parallel field regularization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

T. Lin and H. Hongbin Zha. Riemannian manifold learning. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 30(5):796–809, 2008.

F. Lindgren, H. Rue, and Johan Lindström. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4), 2011.

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

M. Meilă and W. Pentney. Clustering by weighted cuts in directed graphs. In *Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.

H.Q. Minh, P. Niyogi, and Y. Yao. Mercer's theorem, feature maps, and smoothing. In *Proceedings of the 19th Annual Conference on Learning Theory*, 2006.

B. Nadler, S. Lafon, and R. R. Coifman. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. In *Advances in Neural Information Processing Systems (NIPS)*, 2006a.

B. Nadler, S. Lafon, R. R. Coifman, and Kevrekidis. Diffusion maps, spectral clustering and reaction coordiantes of dynamical systems. *Applied and Computational Harmonic Analysis*, 21:113–127, 2006b.

B. Nadler, N. Srebro, and X. Zhou. Statistical analysis of semi-supervised learning: The limit of infnite unlabelled data. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

J. Nash. The imbedding problem for Riemannian manifolds. *Annals of Mathematics*, 63:20–63, 1956.

A. Ng. Preventing "overfitting" of cross-validation data. In *International Conference on Machine Learning*, 1997.

B. Pelletier. Kernel density estimation on riemannian manifolds. *Statistics and Probability Letters*, 73:297–304, 2005.

B. Pelletier. Nonparametric regression estimation on closed riemannian manifolds. *Journal of Nonparametric Statistics*, 18:57–67, 2006.

W. Pentney and M. Meilă. Spectral clustering of biological sequence data. In *Proceedings of the National Conference on Artificial Intelligence*, 2005.

D. Perrault-Joncas, M. Meilă, and M. Scott. Career paths in the job landscape - spectral embedding of directional data (in preparation). 2012.

P. Ram, D. Lee, W. March, and A. G. Gray. Linear-time algorithms for pairwise statistical problems. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. the MIT Press, 2006. URL `http://www.gaussianprocess.org/gpml/`.

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

P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary covariance structure. *Journal of the American Statistical Association*, 87:108–119, 1992.

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

B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Conference on Computational Learning Theory*, pages 416–426, 2001.

SCOP. Structural classification of proteins database. URL `http://scop.berkeley.edu`.

M. Scott. Affinity models for career sequences. *Journal of the Royal Statistical Society - Series C*, (forthcoming).

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

J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions onPattern Analysis and Machine Intelligence*, 2000.

V. Sindhwani, W. Chu, and S. S. Keerthi. Semi-supervised gaussian process classifiers. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2007.

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

T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

A. J. Smola and I.R. Kondor. Kernels and regularization on graphs. In *Proceedings of the Annual Conference on Computational Learning Theory*, 2003.

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

M. L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging.* Springer-Verlag, New York, 1999.

F. Steinke and B. Schölkopf. Kernels, regularization and differential equations. *Pattern Recognition*, 41:3271–3286, 2008.

G. W. Stewart. On the early history of the singular value decomposition. *Society for Industrial and Applied Mathematics Review*, 35:551–556, 1993.

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 *International Conference on Machine Learning*, pages 1079–1086, 2010.

H. Trevor and S. Werner. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

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

K. Weinberger and K. 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, 2004.

K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70:77–90, 2006.

P. Whittle. On stationary processes in the plane. *Biometrika*, 41:434–449, 1954.

P. Whittle. Stochastic processes in several dimensions. *Bulletin of the International Statistical Institute*, 40:974–994, 1963.

T. Wittman. Manifold learning matlab demo, 2005, retreived 2010. URL `http://www.math.umn.edu/~wittman/mani/`.

H. Yue, M. Tomoyasu, and N. Yamanashi. Weighted principal component analysis and its applications to improve fdc performance. In *43rd IEEE Conference on Decision and Control*, 2004.

H. Zha and Z. Zhang. Isometric embedding and continuum isomap. In *International Conference on Machine Learning*, pages 864–871, 2003.

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *Society for Industrial and Applied Mathematics Journal of Scientific Computing*, 26(1):313–338, 2004.

D. Zhou, J. Huang, and B. Scholkopf. Learning from labeled and unlabeled data on a directed graph. In *International Conference on Machine Learning*, pages 1041–1048, 2005a.

D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems (NIPS)*, 2005b.

X. Zhou and M. Belkin. Semi-supervised learning by higher order regularization. In *The 14th International Conference on Artificial Intelligence and Statistics*, 2011.

X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical report, School of Computer Science, Carnegie Mellon University, 2003.

Appendix A

CONSISTENCY OF THE RIEMANNIAN METRIC

Lemma 25 *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. ■