Non-Isometric Manifold Learning: Analysis and an Algorithm

Piotr Dollár PDOLLAR@CS.UCSD.EDU
Vincent Rabaud VRABAUD@CS.UCSD.EDU
Serge Belongie SJB@CS.UCSD.EDU
Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093, USA

Abstract

In this work we take a novel view of nonlinear manifold learning. Usually, manifold learning is formulated in terms of finding an embedding or 'unrolling' of a manifold into a lower dimensional space. Instead, we treat it as the problem of learning a representation of a nonlinear, possibly non-isometric manifold that allows for the manipulation of novel points. Central to this view of manifold learning is the concept of generalization beyond the training data. Drawing on concepts from supervised learning, we establish a framework for studying the problems of model assessment, model complexity, and model selection for manifold learning. We present an extension of a recent algorithm, Locally Smooth Manifold Learning (LSML), and show it has good generalization properties. LSML learns a representation of a manifold or family of related manifolds and can be used for computing geodesic distances, finding the projection of a point onto a manifold, recovering a manifold from points corrupted by noise, generating novel points on a manifold, and more.

1. Introduction

A number of methods have been developed for dealing with high dimensional data sets that fall on or near a smooth low dimensional nonlinear manifold. Such data sets arise whenever the number of modes of variability of the data is much smaller than the dimension of the input space, as is the case for image sequences. Unsupervised manifold learning refers to the problem of recovering the structure of a manifold from a set of unordered sample points. Usually, the problem is formulated in terms of finding an embedding or 'unrolling' of a manifold into a lower dimensional space such that certain geometric relationships between the original points are preserved, as in the seminal works of ISOMAP (Tenenbaum et al., 2000) and LLE (Roweis & Saul, 2000).

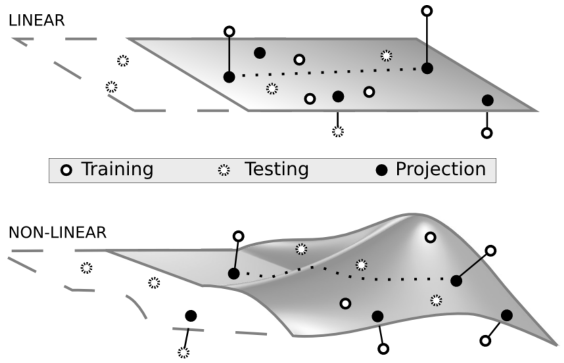

Figure 1. Given the linear subspace (manifold) found by PCA, it is straightforward to project points onto the manifold, measure the distance between a point and the manifold, measure distance between points after projection, and to predict the structure of the manifold in regions with little or no data. In this work we present an extension of LSML that treats manifold learning as a problem of generalizing to unseen portions of a manifold and can be used much like PCA but for nonlinear manifolds. Also, unlike nonlinear embedding methods, LSML can be used with non-isometric manifolds like the one above.

Although good embedding results have been obtained for a number of manifolds, embedding methods are limited in two fundamental ways. First, they are by definition well suited only for manifolds that are *isometric* (or conformal) to a subset of Euclidean space. There is little reason to believe that many manifolds of interest have this property, *e.g.* a sphere does not. Secondly, embedding methods are designed to describe a fixed set of data and not to *generalize* to novel data. Although out-of-sample extensions have been proposed (Bengio et al., 2004), these are typically applicable only in regions of a manifold that have already been sampled densely. Such methods cannot be used to predict manifold structure where no samples are given, even if the manifold has a smooth, consistent form.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. Copyright 2007 by the author(s)/owner(s).

Consider classical MDS, used to find a distance preserving embedding from a matrix of Euclidean distances, and PCA, which finds a low dimensional subspace that explains observed data (Hastie et al., 2001). Although both are designed to deal with linear subspaces, classical MDS is used primarily for data visualization and exploration while PCA is used for finding a low dimensional subspace that can be used to analyze novel data. PCA tends to be more versatile than MDS: given the linear subspace (manifold) found by PCA, it is straightforward to project points onto the manifold, measure the distance between a point and the manifold, measure distance between points after projection, and to predict the structure of the manifold in regions with little or no data.

Nonlinear embedding methods are used much like MDS but for nonlinear manifolds; however, they cannot be used for manipulating novel data. In this work we present an extension of LSML (Dollár et al., 2006), and and show that it can be used much like PCA but for nonlinear manifolds (see Figure 1). LSML treats manifold learning as a problem of generalizing to unseen portions of a manifold, and is applicable to non-isometric cases. Instead of only finding an embedding for visualization, LSML learns a representation of a manifold or family of related manifolds that can be used for computing geodesic distances, finding the projection of a point onto a manifold, recovering a manifold from points corrupted by noise, generating novel points on a manifold, and more.

The remainder of the paper is broken down as follows. We begin with an overview of related work, then in Section 2 we provide an overview of the original LSML and give details of the extension proposed in this work. In Section 3 we introduce a methodology that allows us to perform empirical studies of nonlinear manifold learning methods and show how we can study the generalization properties of LSML. Finally, in Section 4 we show a number of uses for the manifold representation learned by LSML, including projection, manifold de-noising, geodesic distance computation, and manifold transfer.

1.1. Related Work

We begin with a brief overview of literature on nonlinear manifold learning. Traditional methods include self organizing maps, principal curves, and variants of multi-dimensional scaling (MDS) among others, see (Hastie et al., 2001) for a brief introduction to these methods. In recent years, the field has seen a number of interesting developments. (Schölkopf et al., 1998) introduced a kernelized version of PCA. A number of related embedding methods have also been introduced, representatives include LLE (Saul & Roweis, 2003), ISOMAP (Tenenbaum et al., 2000), h-LLE (Donoho & Grimes, 2003), and more recently MVU (Weinberger & Saul, 2006). Broadly, such methods can be classified as spectral embedding methods (Weinberger & Saul, 2006); the embeddings they compute are based on an eigenvector decomposition of an $n \times n$ matrix that represents geometrical relationships of some form between the original n points. Out-of-sample extensions have been proposed (Bengio et al., 2004).

Four methods that LSML shares inspiration with are (Brand, 2003; Keysers et al., 2001; Bengio & Monperrus, 2005; Rao & Ruderman, 1999). Brand (2003) employs a novel charting based method to achieve increased robustness to noise and decreased probability of pathological behavior *vs.* LLE and ISOMAP; we exploit similar ideas in the construction of LSML but differ in motivation and potential applicability. Bengio and Monperrus (2005) proposed a method to learn the tangent space of a manifold and demonstrated a preliminary illustration of rotating a small bitmap image by about 1°. Although our approach to the problem differs, the motivation for LSML is based on similar insights. Work by Keysers et al. (2001) is based on the notion of learning a model for class specific variation, the method reduces to computing a linear tangent subspace that models variability of each class. Rao and Ruderman (1999) shares one of our goals as it addresses the problem of learning Lie groups, the infinitesimal generators of certain geometric transformations.

Finally, there is a vast literature on computing distances between objects undergoing known transformations (Miller & Younes, 2001; Simard et al., 1998), which is essentially the problem of computing distances between manifolds with *known* structure.

2. LSML

Here we give details of the extended version of LSML (Dollár et al., 2006). The basic motivation and general structure of the algorithm is similar to previous work, however, the details differ. We define a new error function and motivate the use of a new regularization term. The minimization of the error is made more efficient, although some details are omitted for space. For an introduction to LSML see Figure 2.

2.1. Motivation and Error Function

Let D be the dimension of the input space, and assume the data lies on a smooth d-dimensional manifold ($d \ll D$). For simplicity assume that the manifold is diffeomorphic to a subset of $\mathbb{R}^d$, meaning that it can be endowed with a global coordinate system (this requirement can easily be relaxed) and that there exists a continuous bijective mapping $\mathcal{M}$ that converts coordinates $\mathbf{y} \in \mathbb{R}^d$ to points $\mathbf{x} \in \mathbb{R}^D$ on the manifold. The goal is to learn a warping function

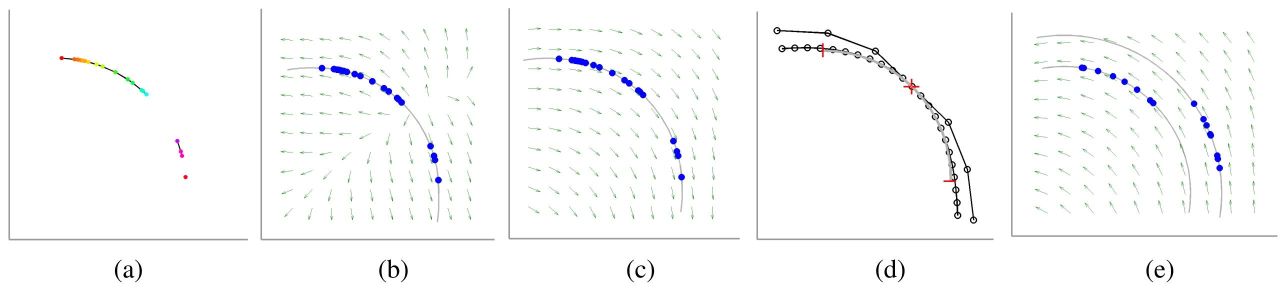

Figure 2. **Overview**. Twenty points (n=20) that lie on 1D curve (d=1) in a 2D space (D=2) are shown in (a). Black lines denote neighbors, in this case the neighborhood graph is not connected. We apply LSML to train $\mathcal{H}$ (with $f = 4$ RBFs). $\mathcal{H}$ maps points in $\mathbb{R}^2$ to tangent vectors; in (b) tangent vectors computed over a regularly spaced grid are displayed, with original points (blue) and curve (gray) overlayed. Tangent vectors near original points align with the curve, but note the seam through the middle. Regularization fixes this problem (c), the resulting tangents roughly align to the curve along its entirety. We can traverse the manifold by taking small steps in the direction of the tangent; (d) shows two such paths, generated starting at the red plus and traversing outward in large steps (outer curve) and finer steps (inner curve). In (e) two parallel curves are shown, with n=8 samples each. Training a common $\mathcal{H}$ results in a vector field that more accurately fits each curve than training a separate $\mathcal{H}$ for each (if their structure was very different this need not be the case).

$\mathcal{W}$ that can take a point on the manifold and return any neighboring point on the manifold, capturing all the modes of variation of the data. Define $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon}) = \mathcal{M}(\mathbf{y} + \boldsymbol{\epsilon})$, where $\mathbf{y} = \mathcal{M}^{-1}(\mathbf{x})$ and $\boldsymbol{\epsilon} \in \mathbb{R}^d$. Taking the first order approximation of the above gives: $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon}) \approx \mathbf{x} + \mathcal{H}(\mathbf{x})\boldsymbol{\epsilon}$, where each column $\mathcal{H}_{\cdot k}(\mathbf{x})$ of the matrix $\mathcal{H}(\mathbf{x})$ is the partial derivative of $\mathcal{M}$ w.r.t. $\mathbf{y}_k$: $\mathcal{H}_{\cdot k}(\mathbf{x}) = \partial/\partial \mathbf{y}_k \mathcal{M}(\mathbf{y})$. This approximation is valid given $\boldsymbol{\epsilon}$ small enough.

The goal of LSML is to learn the function $\mathcal{H}_\theta : \mathbb{R}^D \to \mathbb{R}^{D \times d}$ parameterized by a variable θ. Only data points $\mathbf{x}^i$ sampled from one or several manifolds are given. For each $\mathbf{x}^i$, the set $\mathcal{N}^i$ of neighbors is then computed (*e.g.* using variants of nearest neighbor such as kNN or ϵNN), with the constraint that two points can be neighbors only if they come from the same manifold. The original formulation of LSML was based on the observation that if $\mathbf{x}^j$ is a neighbor of $\mathbf{x}^i$, there then exists an *unknown* $\boldsymbol{\epsilon}^{ij}$ such that $\mathcal{W}(\mathbf{x}^i, \boldsymbol{\epsilon}^{ij}) = \mathbf{x}^j$, or to a good approximation:

$$\mathcal{H}_\theta(\mathbf{x}^i)\boldsymbol{\epsilon}^{ij} \approx \Delta^i_{\cdot j}, \tag{1}$$

where $\Delta^i_{\cdot j} \equiv \mathbf{x}^j - \mathbf{x}^i$. An interpretation of the above is that $\Delta^i_{\cdot j}$ is the *un-centered* estimate of a directional derivative at $\mathbf{x}^i$. However, $\Delta^i_{\cdot j}$ could also serve as the *centered* estimate of the directional derivative at $\overline{\mathbf{x}}^{ij} \equiv \frac{\mathbf{x}^i + \mathbf{x}^j}{2}$:

$$\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} \approx \Delta^i_{\cdot j}. \tag{2}$$

$\overline{\mathbf{x}}^{ij}$ may lie slightly off the manifold, however, as $\mathcal{H}_\theta$ is a smooth mapping over all of $\mathbb{R}^D$, this does not pose a problem. Although the change is subtle, in practice use of (2) provides significant benefit, as the centered approximation of the derivative has no second order error. So, roughly speaking (1) is valid if locally the manifold has a good linear approximation while (2) is valid where a quadratic approximation holds.

To solve for $\mathcal{H}_\theta$, we define the following error:

$$\text{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j} \right\|_2^2. \tag{3}$$

We want to find the θ that minimizes this error. The $\boldsymbol{\epsilon}^{ij}$ are additional free parameters that must be optimized over; they do not affect model complexity.

2.2. Regularization

We can explicitly enforce the mapping $\mathcal{H}_\theta$ to be smooth by adding a regularization term (in addition to implicit smoothness that may come from the form of $\mathcal{H}_\theta$ itself). For each i, the learned tangents at two neighboring locations $\overline{\mathbf{x}}^{ij}$ and $\overline{\mathbf{x}}^{ij'}$ should be similar, *i.e.* $\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'})\|_F^2$ should be small. Note that this may not always be possible, *e.g.* the Hairy Ball Theorem states there is no non-trivial smooth vector field on a sphere. To ensure that the $\mathcal{H}_\theta$'s do not get very small and the $\boldsymbol{\epsilon}$'s very large, $\|\boldsymbol{\epsilon}^{ij}\|_2^2$ must also be constrained. We add the following term to (3):

$$\lambda_E \sum \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 + \lambda_\theta \sum \left\| \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'}) \right\|_F^2. \tag{4}$$

The overall error can be rewritten using a single λ if for any scalar $a > 0$ we treat $\mathcal{H}_\theta$ and $a\mathcal{H}_\theta$ as essentially the same. The error of $\mathcal{H}_\theta$ with regularization parameters $(\lambda_E, \lambda_\theta)$ is the same as the error of $a\mathcal{H}_\theta$ with regularization parameters $(a^2\lambda_E, \frac{1}{a^2}\lambda_\theta)$. Thus there is a single degree of freedom, and we always set $\lambda_E = \lambda_\theta = \lambda$.

2.3. Linear Parametrization

Although potentially any regression technique is applicable, a linear model is particularly easy to work with. We use radial basis functions (RBFs) to define additional features for the data points (Hastie et al., 2001). The number of basis functions, f, is an additional parameter that controls the smoothness of the final mapping $\mathcal{H}_\theta$. Let $\mathbf{f}^{ij}$ be the f features computed from $\overline{\mathbf{x}}^{ij}$. We can then define $\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) = \left[\Theta^1\mathbf{f}^{ij} \cdots \Theta^D\mathbf{f}^{ij}\right]^\top$, where each Θ^k is a $d \times f$ matrix. Re-arranging (3) gives:

$$\mathrm{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j\in\mathcal{N}^i} \sum_{k=1}^{D} \left(\mathbf{f}^{ij\top}\Theta^{k\top}\boldsymbol{\epsilon}^{ij} - \Delta^i_{kj}\right)^2. \quad (5)$$

Solving simultaneously for the $\boldsymbol{\epsilon}$'s and Θ's is complex, but if either the $\boldsymbol{\epsilon}$'s or Θ's are fixed, solving for the remaining variables becomes a least squares problem (details omitted for space). We use an alternating minimization procedure, with random restarts to avoid local minima.

2.4. Radial Basis Functions

For the features we use radial basis functions (RBFs) (Hastie et al., 2001), the number of basis functions, f, being an additional parameter. Each basis function is of the form $f^j(\mathbf{x}) = \exp(-\|\mathbf{x} - \boldsymbol{\mu}^j\|_2^2/2\sigma^2)$ where the centers $\boldsymbol{\mu}^j$ are obtained using K-means clustering on the original data with f clusters and the width parameter σ is set to be twice the average of the minimum distance between each cluster and its nearest neighbor center. The feature vectors are then simply $\mathbf{f}^i = [f^1(\mathbf{x}^i) \cdots f^p(\mathbf{x}^i)]^\top$. The parameter f controls the smoothness of the final mapping $\mathcal{H}_\theta$; larger values result in mappings that better fit local variations of the data, but whose generalization abilities to other points on the manifold may be weaker (see Section 3.2).

3. Analyzing Manifold Learning Methods

In this section we seek to develop a methodology for analyzing nonlinear manifold learning methods without resorting to subjective visual assessment of the quality of an embedding. The motivation is twofold. First, we wish to have an objective criterion by which to compare LSML to existing manifold learning methods. Second, and more importantly, in order to work with non-isometric manifolds (which may not have meaningful embeddings), we need to establish some guidelines for how to properly control the complexity and evaluate the performance of a manifold learning method.

A key issue in supervised learning is the generalization performance of a learning method, defined by its prediction error computed on independent test data (Hastie et al., 2001). Such a notion is of central importance for supervised learning because it provides a principled approach for choosing among learning methods, controlling their complexity, and evaluating their performance. We will show how some of these ideas can be applied in the context of nonlinear manifold learning.

3.1. Model Evaluation

A number of works have established the theoretical relationships between various manifold embedding methods (Bengio et al., 2004; Xiao et al., 2006). Also, asymptotic guarantees exist for ISOMAP, h-LLE and MVU, and conceivably similar bounds could be shown for other methods. Here, however, we are more interested in how these methods perform with a finite sample. We begin by introducing a simple yet intuitive evaluation methodology.

By definition, if a manifold is isometric to a convex subset of a low dimensional Euclidean space, there exists a diffeomorphism that maps the metric of such a manifold, defined by geodesic distance, to the Euclidean metric. A natural measure of the quality of an embedding for this class of manifolds is how closely distance is preserved[1]. Given a sufficient number of samples from an isometric manifold, geodesic distance can be estimated accurately, and this estimate is guaranteed to converge as the number of samples grows (this forms the basis for ISOMAP, we refer the reader to Tenenbaum et al. (2000) for details). We evaluate finite sample performance, using a much larger sample to obtain ground truth distances. This methodology is applicable for manifolds that can be sampled densely, for example for toy data or for image manifolds where we know the underlying transformation or generative model for the manifold.

We assume we are given two sets of samples from the manifold, S_n containing n points, which serves as the training data, and a very large set S_∞, used only during the evaluations stage. The ground truth geodesic distances d_{ij} for each pair of points i, j in S_n are computed using S_∞. Let d'_{ij} denote the Euclidean distance between points i, j in a given embedding. We define the error of an embedding as:

$$\mathrm{err}_{\mathrm{GD}} \equiv \frac{1}{n^2} \sum_{ij} \frac{|d_{ij} - d'_{ij}|}{d_{ij}}. \quad (6)$$

All experiments presented here were averaged over 10 trials. Reasonable effort was made to maximize the performance of each method tested. We compared LSML to three embedding methods: ISOMAP (Tenenbaum et al., 2000), LLE (Roweis & Saul, 2000) and (fast) MVU (Weinberger & Saul, 2006), each of which has code available online. The embedding methods require a fully connected neighborhood graph; we simply discarded data that resulted in disconnected graphs. We sampled at most 1000 pairs of neighboring points to train LSML, under these conditions training takes around two minutes regardless of n or the number of neighbors k. Details of how to use the learned

[1] LLE, MVU and LSML are applicable to larger classes of manifolds. However, any method should work with isometric manifolds. There have also been some results on conformal embeddings (which preserve angles but not distances); a similar methodology could be developed for this case. Finally, note that LLE finds an embedding that can differ by an affine transformation from a distance preserving embedding, and so must be adjusted accordingly.

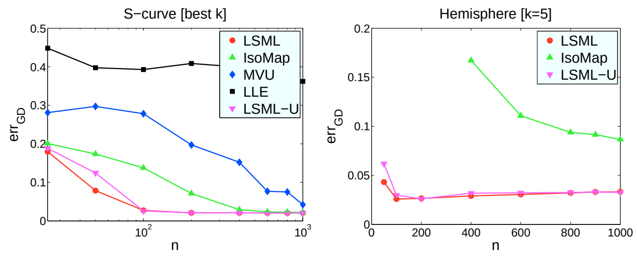

Figure 3. **Finite sample performance**. Performance of various algorithms, measured by $\mathrm{err_{GD}}$ as n is varied. All results shown are averaged over 10 repetitions. *Left*: S-curve error for various methods (k chosen optimally for each data point). ISOMAP, MVU and LSML all performed well for large n. LSML performed quite well even with n=25. Qualitatively similar results were obtained for other manifolds. *Right*: ISOMAP (geodesic distance computation only) and LSML results on a hemisphere. Here $k = 5$ was fixed, preventing ISOMAP from converging to a lower error solution. LSML's performance did not improve after n=100 because of sampling (see text).

model for geodesic distance computation are given in Section 4.3. Since computing pairwise distances is time consuming for large n, we sample 100 landmark points.

The first experiments examine finite sample performance (see Figure 3). The performance of the embedding methods ranked as ISOMAP $\succ$ MVU $\succ$ LLE, with the performance of LLE being quite bad even for large n. To be fair however, LLE can recover embeddings for classes of manifolds where MVU and ISOMAP cannot (and likewise MVU is more general than ISOMAP). LSML performed better than these embedding methods, especially for small n. LSML-U refers to the version of LSML from (Dollár et al., 2006), it generally performs similarly to LSML, except for small n or large k (when curvature is non-negligible).

3.2. Model Complexity

The first step of many embedding methods is to construct a neighborhood graph of the data, using for example the k-nearest neighbors of each point. Thus all these methods have at least one parameter, k, usually set by hand. If k is too small estimates of local geometry may have high error or *variance*, either due to noise or lack of information. If k is too large estimates of local geometry may be *biased* by manifold curvature. For the trivial case of a linear subspace, which has no curvature, increasing k should continuously decrease error since there is no bias term.

In these terms the choice of k is reminiscent of the classic bias-variance tradeoff in supervised learning. Another view of the tradeoff is that enlarging k increases the number of constraints on the embedding, except that additional constraints become increasingly inaccurate. See Figure 4 for the effects of neighborhood size and noise on performance for each method.

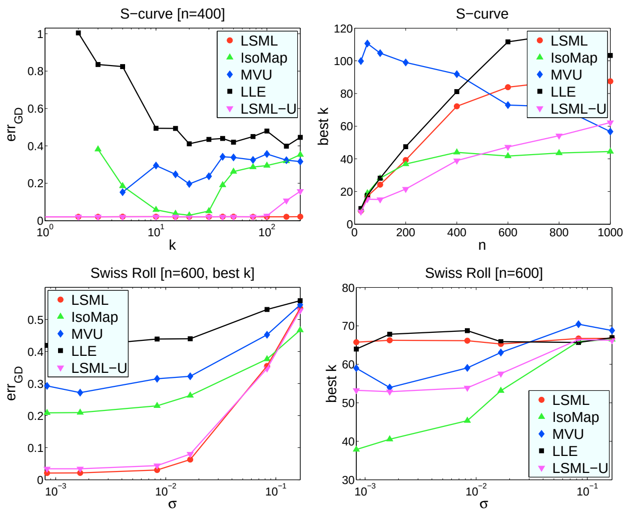

Figure 4. **Bias-Variance Tradeoff**. *Top*: Effects of neighborhood size. *Left*: Error as a function of k for n=400 points from S-curve; note 'U'-shaped curves. Note also the robustness of LSML to very small (k=1) and very large (k=100) neighborhood sizes (eventually LSML does break down, not shown). The reason for this robustness is due to use of the centered error (LSML-U is not quite as robust for large k), see Section 2.1. *Right*: Best value of k for each method as a function of n. For all but MVU k increases sub-linearly with n. *Bottom*: Effects of noise. *Left*: As the amount of noise σ increases, the error of each method increases. LSML performs well, even without the de-noising introduced in Section 4.2. *Right*: Best value of k for each method as a function of σ. For MVU and ISOMAP k increases to compensate for additional noise, LSML always prefers fairly large k given noise.

In addition to k, LSML has two more smoothing parameters: the number of RBF's and the regularization parameter λ. LLE also has a regularization term whose effects can be significant and MVU has a parameter ω that balances between maximizing variance and penalizing slack. Ideally, we would like some way of automatically performing model selection; *i.e.* automatically selecting the value for each of these parameters for a given problem instead of relying on a visual assessment of the final embedding.

3.3. Model Selection

$\mathrm{err_{GD}}$ can be used to analyze model complexity in certain settings, as above. However, in general there is no large sample set available from which to compute ground truth distances, in which case $\mathrm{err_{GD}}$ cannot be used to perform model selection. Also, for the same reasons that in general the quality of an embedding cannot be evaluated, neither can the quality of out-of-sample extension, so this does not provide an answer[2]. The key difficulty stems from the fact

[2]Bengio et al. (2004) evaluated the out-of-sample extension of various methods by seeing how consistent the out-of-sample prediction was compared to simply including the point in the original embedding. However this is only a partial solution since consistency does not imply accuracy (consider a trivial embedding which maps all points to the origin).

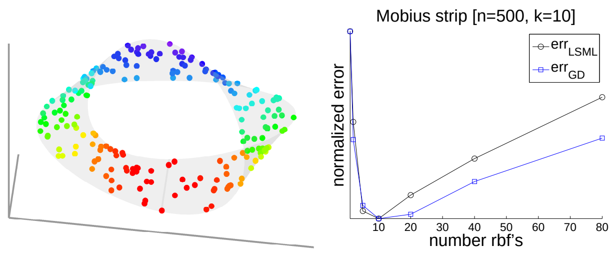

Figure 5. **LSML Test Error**. When err$_{\text{GD}}$ can be computed it is strongly correlated with err$_{\text{LSML}}$, supporting our use of err$_{\text{LSML}}$ to perform model evaluation. *Left*: Points drawn from a Möbius strip, a manifold with $d = 2$, with a small amount of noise added. *Right*: err$_{\text{LSML}}$ and err$_{\text{GD}}$, each normalized to lie between 0 and 1, as the number of RBFs was varied. Of 500 points, $\frac{2}{3}$ were used for training, $\frac{1}{3}$ for testing. According to both error measures approximately 10 to 20 RBFs was best.

that most manifold embedding methods make no *testable* predictions.

If manifold learning can be posed in such a way that it makes testable predictions, the quality of the predictions can potentially be used to perform model selection. For example, if in addition to an embedding the reverse mapping back to $\mathbb{R}^D$ is also learned, error can be measured by how well a test point is encoded. This principle formed the basis for auto-encoding neural nets (DeMers & Cottrell, 1993).

Given a notion of test error, standard techniques from supervised learning, such as cross-validation, can be used for performing model selection and evaluation for manifold learning. However, although any testable prediction gives rise to an error measure, not all error measures are necessarily useful. err$_{\text{GD}}$ gives a way to evaluate performance in certain scenarios; at minimum prediction error should roughly correlate with err$_{\text{GD}}$ when it is available.

3.4. LSML Test Error

LSML predicts the tangent planes $\mathcal{H}$ at all points on a manifold. We can define the test error of LSML according to how well the predicted tangent planes fit unseen data (see Eq. (2)). Given unseen test points $\mathbf{x}^i$, the error is:

$$\text{err}_{\text{LSML}} \equiv \sum_i \min_{\boldsymbol{\epsilon}^{ii'}} \left\| \mathcal{H}_\theta(\overline{\mathbf{x}}^{ii'})\boldsymbol{\epsilon}^{ii'} - (\mathbf{x}^i - \mathbf{x}^{i'}) \right\|_2^2, \qquad (7)$$

where $\mathbf{x}^{i'}$ is the closest neighbor of $\mathbf{x}^i$, either another test point or a training point.

err$_{\text{LSML}}$ is strongly correlated with err$_{\text{GD}}$ both for isometric and non-isometric manifolds. For example, in Figure 5 we show the effect of changing the number of RBFs on both errors for data that lies on a Möbius strip; note that the minima of the errors occur at roughly the same spot. This correlation is important, because it allows us to use err$_{\text{LSML}}$ in place of err$_{\text{GD}}$ to perform model evaluation, select model parameters, *etc*. Although we can only prove the utility of err$_{\text{LSML}}$ for manifolds for which err$_{\text{GD}}$ is defined, err$_{\text{LSML}}$ is most useful when err$_{\text{GD}}$ cannot be computed.

Finally, note that err$_{\text{LSML}}$ cannot be used to choose d, since as d increases err$_{\text{LSML}}$ will decrease. This is the same challenge as choosing k in K-means clustering. A simple rule of thumb is to choose the smallest value of d such that further increasing d results in only a small decrease in error.

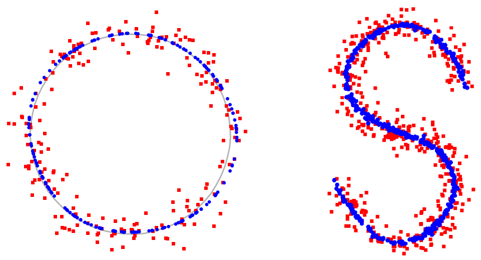

Figure 6. **Manifold De-noising**. Red (square) points are the noisy training data used to learn $\mathcal{H}_\theta$. After, the de-noising procedure from Section 4.2 was applied to recover the blue (circular) points. Original manifolds shown in gray. *Left*: Circle (D=2, d=1, n=200). *Right*: S-curve, side view (D=3, d=2, n=500).

4. Working with $\mathcal{H}_\theta$

We now develop a number of uses for the representation $\mathcal{H}_\theta$ learned by LSML, including projection, manifold de-noising, geodesic distance computation, and manifold transfer. For all experiments we use err$_{\text{LSML}}$ to select model parameters, including the number of RBF's, k, and the regularization parameter λ. Given low test error, we expect geodesic distance computation, de-noising, *etc*. to be accurate (see Section 3.4).

$\mathcal{H}_\theta$ is a function defined everywhere on $\mathbb{R}^D$, not just for points from the original manifold. This allows us to work with points that don't lie precisely on the manifold. Finally, note that in addition to $\mathcal{H}_\theta$ at least one training point is necessary to identify a manifold, and in practice keeping more training points is useful.

4.1. Projection

The projection of a point onto a linear subspace is unique and can be computed in closed form. For nonlinear manifolds this is in general not the case. $\mathbf{x}'$ is an orthogonal projection of a point $\mathbf{x}$ onto a manifold if it minimizes $\|\mathbf{x} - \mathbf{x}'\|_2^2$ in an open neighborhood. We can find such a point by initially setting $\mathbf{x}'$ to be some point from the manifold and then performing gradient descent, being careful not to leave the support of the manifold. $\mathbf{x}'$ is therefore bound to only follow the projection of the gradient on the local tangent space defined by $\mathcal{H}_\theta(\mathbf{x}')$. Let $H' \equiv \text{orth}(\mathcal{H}_\theta(\mathbf{x}'))$ be the $d \times D$ matrix that denotes the

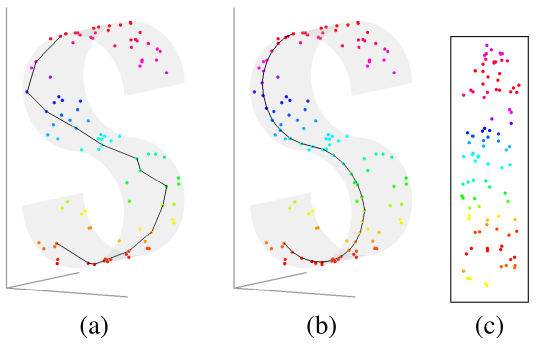

Figure 7. **Geodesic Distance**. We can find a geodesic path between two points by using a snake. The key idea is to start with a discretized path between $\mathbf{x}$ and $\mathbf{x}'$ and perform gradient descent until the length of the path cannot decrease. (a) Initial path between two points on S-curve. (b) Locally shortest path (geodesic) obtained after gradient descent. (c) Embedding computed using classical MDS on geodesic distances (only applicable for isometric manifolds). $\mathcal{H}_\theta$ was trained on the n=100 points shown.

orthonormalized tangent space at $\mathbf{x}'$, and $H'H'^{\top}$ the corresponding projection matrix. The update rule for $\mathbf{x}'$ (with step size α) is:

$$\mathbf{x}' \leftarrow \mathbf{x}' + \alpha H'H'^{\top}(\mathbf{x} - \mathbf{x}'). \quad (8)$$

There may be multiple orthogonal projections but we generally are interested in the closest one. A simple heuristic is to initially set $\mathbf{x}'$ to the nearest point in the training data.

4.2. Manifold De-noising

Assume we have learned or are given the mapping $\mathcal{H}_\theta$ that describes the tangent space of some manifold or family of manifolds. Suppose we are also given samples $\mathbf{x}^i$ from a manifold that have been corrupted by noise so they no longer lie exactly on the manifold (these can be the same samples from which $\mathcal{H}_\theta$ was learned). Here we show how to recover a set of points χ^i that are close to the noisy samples $\mathbf{x}^i$ but lie on a manifold consistent with $\mathcal{H}_\theta$.

The key to the approach is the observation that, as in Eq. (3), if χ^j is a neighbor of χ^i then there exists an unknown $\boldsymbol{\epsilon}^{ij}$ such that $\|\mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j)\|_2^2$ is small. We can thus find a series of points χ^i that are close to the original points $\mathbf{x}^i$ and satisfy the above. The error to minimize is the sum of two terms:

$$\text{err}_{\mathcal{M}}(\chi) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j) \right\|_2^2 \quad (9)$$

$$\text{err}_{\text{orig}}(\chi) = \sum_{i=1}^{n} \left\| \chi^i - \mathbf{x}^i \right\|_2^2 \quad (10)$$

The second term is multiplied by a constant λ_{noise} to weigh the importance of sticking to the original points. We assume that for a small change in χ^i, $\frac{\partial \mathcal{H}_\theta}{\partial \chi^i}$ is negligible with respect to the other quantities. Under such an assumption, the gradient of $\text{err}_{\mathcal{M}}(\chi)$ can easily be rewritten as a product of matrices. We find a minimum energy solution by initially setting $\chi^i = \mathbf{x}^i$ for each i and then performing gradient descent on the χ^i's, this time using only the component of the gradient that is orthonormal to the manifold (we cannot correct the noise in the co-linear direction).

Manifold de-noising has received some attention in the literature (Park et al., 2004); the approach presented here is simpler and does not suffer from the "trim the peak and fill the valley" phenomenon. We show results in Figure 6.

4.3. Geodesic Distance

Given two points on a manifold, $\mathbf{x}$ and $\mathbf{x}'$, we want to compute the geodesic (locally minimal) distance between them. To find such a minimal path we use an active contour model, also known as a 'snake' (Kass et al., 1988; Blake & Isard, 1998), where the length of a discretized path between $\mathbf{x}$ and $\mathbf{x}'$ is optimized by gradient descent.

Let $\chi^1 = \mathbf{x}$ and $\chi^m = \mathbf{x}'$, and $\chi^1, \ldots, \chi^m$ denote a sequence of points on the manifold forming a path between them. The length of the path is given by $\sum_{i=2}^{m} \left\|\chi^i - \chi^{i-1}\right\|_2$. For computational reasons, we use the following instead:

$$\text{err}_{\text{length}}(\chi) = \sum_{i=2}^{m} \left\|\chi^i - \chi^{i-1}\right\|_2^2. \quad (11)$$

We minimize $\text{err}_{\text{length}}$ via gradient descent, keeping the ends of the snake fixed and again being careful not to leave the support of the manifold. The update rule for each χ^i is very similar to the update rule for projection given in Eq. (8).

To get an initial estimate of the snake, we apply Dijkstra's algorithm to the original points (also the first step of ISOMAP). To increase accuracy, additional points are linearly interpolated, and to ensure they lie on the manifold they are 'de-noised' by the procedure from Section (4.2) (with $\lambda_{\text{noise}} = 0$ and neighbors being adjacent points).

Figure 7 shows some example snakes computed over an S-curve. In Figure 8 we show an example snake that plausibly crosses a region where no points were sampled.

4.4. Manifold Transfer

A related problem to learning the structure of a single manifold is to simultaneously learn the representation of multiple manifolds that may share common structure (*cf*. Figure 2e). Such manifolds can arise whenever a common transformation acts on multiple objects, *e.g.* in the image domain this is quite common. One possibility is to relate representations learned separately for each manifold (Elgammal & Lee, 2004), however, learning simultaneously for all manifolds allows sharing of information.

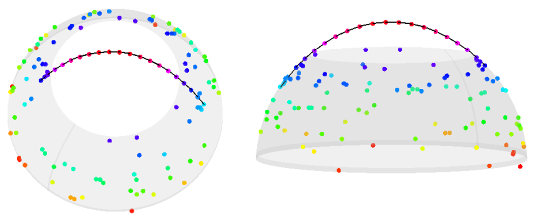

Figure 8. **Missing Data**. $\mathcal{H}_\theta$ was trained on n=100 points sampled from a hemisphere with the top portion removed. $\mathcal{H}_\theta$ plausibly predicted the manifold structure where no samples were given. Shown are two views of a geodesic computed between two points on opposite sides of the hole (see Section 4.3).

An even more interesting problem is generalizing to *novel* manifolds. Given multiple sets of points, each sampled from a single manifold, we can formulate the problem as learning and generalizing to *unseen* manifolds. Once again err_{LSML} serves as the test error, except now it is computed from points on the unseen manifolds. LSML can already be trained on data of this form; two points being defined as neighbors if they are close in Euclidean space and come from the same manifold. Results of manifold transfer were shown in Figure 6 of (Dollár et al., 2006).

5. Conclusion

In this work we presented a novel examination of nonlinear manifold learning, posing it as the problem of learning manifold representations in a manner that allows for the manipulation of novel data. Drawing on concepts from supervised learning, we presented a theoretically sound methodology for quantifying the generalization performance of manifold learning approaches. With this formalism in hand, we presented an extended version of LSML and showed how it can be applied to tasks including nonlinear projection, manifold de-noising, geodesic distance computation and manifold transfer. In ongoing work we are scaling the implementation of LSML to handle large datasets, and tuning it for use in the image domain. In future work we plan to investigate applications that entail generalization to entire families of manifolds.

Acknowledgements

This work was funded by the following grants and organizations: NSF Career Grant #0448615, Alfred P. Sloan Research Fellowship, NSF IGERT Grant DGE-0333451, and UCSD Division of Calit2 and in part by the UCSD FWGrid Project, NSF Research Infrastructure Grant Number EIA-0303622. We would like to thank Lawrence Cayton, Sanjoy Dasgupta and Craig Donner for valuable input and once again Anna Shemorry for her uncanny patience.

References

Bengio, Y., & Monperrus, M. (2005). Non-local manifold tangent learning. *NIPS*.

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Le Roux, N., & Ouimet, M. (2004). Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. *NIPS*.

Blake, A., & Isard, M. (1998). *Active contours*. Berlin Heidelberg New York: Springer.

Brand, M. (2003). Charting a manifold. *NIPS*.

DeMers, D., & Cottrell, G. (1993). Non-linear dimensionality reduction. *NIPS*.

Dollár, P., Rabaud, V., & Belongie, S. (2006). Learning to traverse image manifolds. *NIPS*.

Donoho, D., & Grimes, C. (2003). Hessian eigenmaps: locally linear embedding techniques for high dimensional data. *Proc. of National Academy of Sciences*.

Elgammal, A., & Lee, C. (2004). Separating style and content on a nonlinear manifold. *CVPR*.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Springer.

Kass, M., Witkin, A., & Terzopoulos, D. (1988). Snakes: Active contour models. *IJCV*.

Keysers, D., Macherey, W., Dahmen, J., & Ney, H. (2001). Learning of variability for invariant statistical pattern recognition. *ECML*.

Miller, M., & Younes, L. (2001). Group actions, homeomorphisms, and matching: A general framework. *IJCV*.

Park, J. H., Zhang, Z., Zha, H., & Kasturi, R. (2004). Local smoothing for manifold learning. *CVPR*.

Rao, R., & Ruderman, D. (1999). Learning Lie groups for invariant visual perception. *NIPS*.

Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*.

Saul, L. K., & Roweis, S. T. (2003). Think globally, fit locally: unsupervised learning of low dimensional manifolds. *JMLR*.

Schölkopf, B., Smola, A., & Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neur. Comp.*

Simard, P., LeCun, Y., Denker, J. S., & Victorri, B. (1998). Transformation invariance in pattern recognition-tangent distance and tangent propagation. *Neural Networks: Tricks of the Trade*.

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dim. reduct. *Science*, *290*.

Weinberger, K. Q., & Saul, L. K. (2006). Unsupervised learning of image manifolds by semidefinite programming. *IJCV*.

Xiao, L., Sun, J., & Boyd, S. (2006). A duality view of spectral methods for dimensionality reduction. *ICML*.