Unsupervised Learning of Image Manifolds by Semidefinite Programming

Kilian Q. Weinberger and Lawrence K. Saul
Department of Computer and Information Science
University of Pennsylvania, Levine Hall
3330 Walnut Street, Philadelphia, PA 19104-6389
{*kilianw,lsaul*}*@cis.upenn.edu*

Abstract

Can we detect low dimensional structure in high dimensional data sets of images and video? The problem of dimensionality reduction arises often in computer vision and pattern recognition. In this paper, we propose a new solution to this problem based on semidefinite programming. Our algorithm can be used to analyze high dimensional data that lies on or near a low dimensional manifold. It overcomes certain limitations of previous work in manifold learning, such as Isomap and locally linear embedding. We illustrate the algorithm on easily visualized examples of curves and surfaces, as well as on actual images of faces, handwritten digits, and solid objects.

1. Introduction

Many data sets of images and video are characterized by far fewer degrees of freedom than the actual number of pixels per image. The problem of dimensionality reduction is to understand and analyze these images in terms of their basic modes of variability—for example, the pose and expression of a human face, or the rotation and scaling of a solid object. Mathematically, we can view an image as a point in a high dimensional vector space whose dimensionality is equal to the number of pixels in the image [3, 20]. If the images in a data set are effectively parameterized by a small number of continuous variables, then they will lie on or near a low dimensional *manifold* in this high dimensional space [12]. This paper is concerned with the unsupervised learning of such image manifolds.

Beyond its applications in computer vision, manifold learning is best described as a problem at the intersection of statistics, geometry, and computation. The problem is illustrated in Fig. 1. Given high dimensional data sampled from a low dimensional manifold, how can we efficiently compute a faithful (nonlinear) embedding? In the last few years, researchers have uncovered a large family of algorithms for computing such embeddings from the top or bottom eigenvectors of an appropriately constructed matrix. These algorithms—including Isomap [19], locally linear embedding (LLE) [14, 15], hessian LLE [8], Laplacian eigenmaps [1], and others [5]—can reveal low dimensional manifolds that are not detected by classical linear methods, such as principal component analysis (PCA) [11].

Our main contribution in this paper is a new algorithm for manifold learning based on semidefinite programming. Like Isomap and LLE, it relies on efficient and tractable

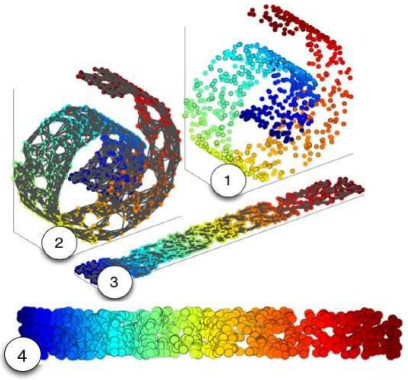

Figure 1. The problem of manifold learning, illustrated for $N = 800$ data points sampled from a "Swiss roll". (1). A discretized manifold is revealed by connecting each data point and its $k = 6$ nearest neighbors (2). An unsupervised learning algorithm unfolds the Swiss roll while preserving the local geometry of nearby data points (3). Finally, the data points are projected onto the two dimensional subspace that maximizes their variance, yielding a faithful embedding of the original manifold (4).

optimizations that are not plagued by spurious local minima. Interestingly, though, our algorithm is based on a completely different geometric intuition (and optimization), and it overcomes certain limitations of previous work.

2. Dimensionality Reduction

We study dimensionality reduction as a problem in unsupervised learning. Given N high dimensional inputs $\vec{X}_i \in \mathcal{R}^D$ (where $i = 1, 2, \ldots, N$), the problem is to compute outputs $\vec{Y}_i \in \mathcal{R}^d$ in one-to-one correspondence with the inputs that provide a faithful embedding in $d < D$ dimensions. By "faithful", we mean that nearby points remain nearby and that distant points remain distant; we shall make this intuition more precise in what follows. Ideally, an unsupervised learning algorithm should also estimate the intrinsic dimensionality d of the manifold sampled by the inputs $\vec{X}_i$.

Our algorithm for manifold learning builds on classical linear methods for dimensionality reduction. We therefore begin by briefly reviewing principal component analysis (PCA) [11] and metric multidimensional scaling (MDS) [6]. The generalization from subspaces to manifolds is then made by introducing the idea of local isometry.

2.1. Linear Methods

PCA and MDS are based on simple geometric intuitions. In PCA, the inputs are projected into the lower dimensional subspace that maximizes the projected variance; the basis vectors of this subspace are given by the top eigenvectors of the $D \times D$ covariance matrix, $C = \frac{1}{N} \sum_i \vec{X}_i \vec{X}_i^T$. (Here and in what follows, we assume without loss of generality that the inputs are centered on the origin: $\sum_i \vec{X}_i = \vec{0}$.)

In MDS with classical scaling, the inputs are projected into the subspace that best preserves their pairwise squared distances $|\vec{X}_i - \vec{X}_j|^2$ or, as done in practice, their dot products $\vec{X}_i \cdot \vec{X}_j$. The outputs of MDS are computed from the top eigenvectors of the $N \times N$ Gram matrix with elements $G_{ij} = \vec{X}_i \cdot \vec{X}_j$. Note that a set of vectors is determined up to rotation by its Gram matrix of dot products.

Though based on different geometric intuitions, PCA and MDS yield the same results—essentially a rotation of the inputs followed by a projection into the subspace with the highest variance. The correlation matrix of PCA and the Gram matrix of MDS have the same rank and nonzero eigenvalues up to a constant factor. Both matrices are semi-positive definite, and gaps in their eigenvalue spectra indicate that the high dimensional inputs $X_i \in \mathcal{R}^D$ lie to a good approximation in a lower dimensional subspace of dimensionality d, where d is the number of appreciably positive eigenvalues. These linear methods for dimensionality reduction generate faithful embeddings when the inputs are mainly confined to a low dimensional subspace; in this case, their eigenvalues also reveal the correct underlying dimensionality. They do not generally succeed, however, in the case that the inputs lie on a low dimensional manifold.

2.2. From Subspaces to Manifolds

We will refer to any method that computes a low dimensional embedding from the eigenvectors of an appropriately constructed matrix as a method in *spectral embedding*. If PCA and MDS are linear methods in spectral embedding, what are their nonlinear counterparts? In fact, there are several, most of them differing in the geometric intuition they take as starting points and in the generalizations of linear transformations that they attempt to discover.

The nonlinear method we propose in this paper is based fundamentally on the notion of *isometry*. (For the sake of exposition, we defer a discussion of competing nonlinear methods based on isometries [8, 19] to section 5.) Formally, two Riemannian manifolds are said to be isometric if there is a diffeomorphism such that the metric on one pulls back to the metric on the other. Informally, an isometry is a smooth invertible mapping that looks locally like a rotation plus translation, thus preserving distances along the manifold. Intuitively, for two dimensional surfaces, the class of isometries includes whatever physical transformations one can perform on a sheet of paper without introducing holes, tears, or self-intersections. Many interesting image manifolds are isometric to connected subsets of Euclidean space [7].

Isometry is a relation between manifolds, but we can extend the notion in a natural way to data sets. Consider two data sets $X = \{\vec{X}_i\}_{i=1}^N$ and $Y = \{\vec{Y}_i\}_{i=1}^N$ that are in one-to-one correspondence. Let the $N \times N$ binary matrix η indicate a neighborhood relation on X and Y, such that we regard $\vec{X}_j$ as a neighbor of $\vec{X}_i$ if and only if $\eta_{ij} = 1$ (and similarly, for $\vec{Y}_j$ and $\vec{Y}_i$). We will say that *the data sets X and Y are locally isometric under the neighborhood relation η if for every point $\vec{X}_i$, there exists a rotation, reflection and/or translation that maps $\vec{X}_i$ and its neighbors precisely onto $\vec{Y}_i$ and its neighbors.*

We can translate the above definition into various sets of equality constraints on X and Y. To begin, note that the local mapping between neighborhoods will exist if and only if the distances and angles between points and their neighbors are preserved. Thus, whenever both $\vec{X}_j$ and $\vec{X}_k$ are neighbors of $\vec{X}_i$ (that is, $\eta_{ij}\eta_{ik} = 1$), for local isometry we must have that:

$$\left(\vec{Y}_i - \vec{Y}_j\right) \cdot \left(\vec{Y}_i - \vec{Y}_k\right) \;=\; \left(\vec{X}_i - \vec{X}_j\right) \cdot \left(\vec{X}_i - \vec{X}_k\right). \quad (1)$$

Eq. (1) is sufficient for local isometry because the triangle formed by any point and its neighbors is determined up to rotation, reflection and translation by specifying the lengths

of two sides and the angle between them. In fact, such a triangle is similarly determined by specifying the lengths of all its sides. Thus, we can also say that X and Y are locally isometric under η if whenever $\vec{X}_i$ and $\vec{X}_j$ are themselves neighbors (that is, $\eta_{ij} = 1$) or are common neighbors of another point in the data set (that is, $[\eta^T \eta]_{ij} > 0$), we have:

$$\left|\vec{Y}_i - \vec{Y}_j\right|^2 \;=\; \left|\vec{X}_i - \vec{X}_j\right|^2. \tag{2}$$

This is an equivalent characterization of local isometry as eq. (1), but expressed only in terms of pairwise distances. Finally, we can express these constraints purely in terms of dot products. Let $G_{ij} = \vec{X}_i \cdot \vec{X}_j$ and $K_{ij} = \vec{Y}_i \cdot \vec{Y}_j$ denote the Gram matrices of the inputs and outputs, respectively. We can rewrite eq. (2) as:

$$K_{ii} + K_{jj} - K_{ij} - K_{ji} \;=\; G_{ii} + G_{jj} - G_{ij} - G_{ji}. \tag{3}$$

Eq. (3) expresses the conditions for local isometry purely in terms of Gram matrices; it is in fact this formulation that will form the basis of our algorithm for manifold learning.

3. Semidefinite Embedding

We can now formulate the problem of manifold learning more precisely, taking as a starting point the notion of local isometry. In particular, given N inputs $\vec{X}_i \in \mathcal{R}^D$ and a prescription for identifying "neighboring" inputs, can we find N outputs $\vec{Y}_i \in \mathcal{R}^d$, where $d < D$, such that the inputs and outputs are locally isometric, or at least approximately so? Alternatively, we can state the problem in terms of Gram matrices: can we find a Gram matrix K_{ij} that satisfies the constraints in eq. (3), and for which the vectors $\vec{Y}_i$ (which are determined up to a rotation by the elements of the Gram matrix) lie in a subspace of dimensionality $d < D$, or at least approximately lie in such a subspace? In this section, we show how this can be done by a constrained optimization over the cone of semidefinite matrices.

Like PCA and MDS, the algorithm we propose for manifold learning is based on a simple geometric intuition. Imagine each input $\vec{X}_i$ as a steel ball that is connected to its k nearest neighbors by rigid rods. The effect of the rigid rods is to fix the distances and angles between nearest neighbors, no matter what other forces are applied to the inputs. Now imagine that the inputs are pulled apart, maximizing their total variance subject to the constraints imposed by the rigid rods. Fig. 1 shows the unraveling effect of this transformation on inputs sampled from the Swiss roll. The goal of this section is to formalize the steps of this transformation—in particular, the constraints that must be satisfied by the final solution, and the nature of the optimization that must be performed.

3.1. Constraints

The constraints that we need to impose for local isometry are naturally represented by a graph with N nodes, one for each input. Consider the graph formed by connecting each input to its k nearest neighbors, where k is a free parameter of the algorithm. For simplicity, we assume that the graph formed in this way is connected; if not, then each connected component should be analyzed separately. The constraints for local isometry under this neighborhood relation are simply to preserve the lengths of the edges in this graph, as well as the angles between edges at the same node. In practice, it is easier to deal only with constraints on distances, as opposed to angles. To this end, let us further connect the graph by adding edges between the neighbors of each node (if they do not already exist). Now by preserving the distances of all edges in this new graph, we preserve both the distances of edges and the angles between edges in the original graph—because if all sides of a triangle are preserved, so are its angles.

In addition to imposing the constraints represented by the "neighborhood graph", we also constrain the outputs $\vec{Y}_i$ to be centered on the origin:

$$\sum\nolimits_i \vec{Y}_i = \vec{0}. \tag{4}$$

Eq. (4) simply removes a translational degree of freedom from the final solution. The centering constraint can be expressed in terms of the Gram matrix K_{ij} as follows:

$$0 = \left|\sum\nolimits_i \vec{Y}_i\right|^2 = \sum_{ij} \vec{Y}_i \cdot \vec{Y}_j = \sum_{ij} K_{ij}. \tag{5}$$

Note that eq. (5) is a linear equality constraint on the elements of the output Gram matrix, just like eq. (3).

Because the geometric constraints on the outputs $\vec{Y}_i$ are so naturally expressed in terms of the Gram matrix K_{ij} (and because the outputs are determined up to rotation by their Gram matrix), we may view manifold learning as an optimization over Gram matrices K_{ij} rather than vectors $\vec{Y}_i$. Not all matrices, however, can be interpreted as Gram matrices: only symmetric matrices with nonnegative eigenvalues can be interpreted in this way. Thus, we must further constrain the optimization to the cone of semidefinite matrices [21].

In sum, there are three types of constraints on the Gram matrix K_{ij}, arising from local isometry, centering, and semidefiniteness. The first two involve linear equality constraints; the last one is not linear, but importantly it is *convex*. We will exploit this property in what follows. Note that there are $O(Nk^2)$ constraints on $O(N^2)$ matrix elements, and that the constraints are not incompatible, since at the very least they are satisfied by the input Gram matrix G_{ij} (assuming, as before, that the inputs $\vec{X}_i$ are centered on the origin).

3.2. Optimization

What function of the Gram matrix can we optimize to "unfold" a manifold, as in Fig. 1? As motivation, consider the ends of a piece of string, or the corners of a flag. Any slack in the string serves to decrease the (Euclidean) distance between its two ends; likewise, any furling of the flag serves to bring its corners closer together. More generally, we observe that any "fold" between two points on a manifold serves to decrease the Euclidean distance between the points. This suggests an optimization that we can perform to compute the outputs $\vec{Y}_i$ that unfold a manifold sampled by inputs $\vec{X}_i$. In particular, we propose to maximize the sum of pairwise squared distances between outputs:

$$\mathcal{T}(Y) = \frac{1}{2N} \sum_{ij} \left| \vec{Y}_i - \vec{Y}_j \right|^2 . \tag{6}$$

By maximizing eq. (6), we pull the outputs as far apart as possible, *subject to the constraints in the previous section*.

Before expressing this objective function in terms of the Gram matrix K_{ij}, let us verify that it is indeed bounded, meaning that we cannot pull the outputs infinitely far apart. Intuitively, the constraints to preserve local distances (and the assumption that the graph is connected) prevent such a divergence. More formally, let $\eta_{ij} = 1$ if $\vec{X}_j$ is one of the k nearest neighbors of $\vec{X}_i$, and zero otherwise, and let τ be the maximal distance between any two such neighbors:

$$\tau = \max_{ij} \left[\eta_{ij} \left| \vec{X}_i - \vec{X}_j \right| \right] . \tag{7}$$

Assuming the graph is connected, then the longest path through the graph has a distance of at most $N\tau$. We observe furthermore that given two nodes, the distance of the path through the graph provides an upper bound on their Euclidean distance. Thus, for all outputs $\vec{Y}_i$ and $\vec{Y}_j$, we must have $|\vec{Y}_i - \vec{Y}_j| < N\tau$. Using this to provide an upper bound on the objective function in eq. (6), we obtain:

$$\mathcal{T}(Y) \;\leq\; \frac{1}{2N} \sum_{ij} (N\tau)^2 \;=\; \frac{N^3 \tau^2}{2}. \tag{8}$$

Thus, the objective function cannot increase without bound if we enforce the constraints to preserve local distances.

We can express the objective function in eq. (6) directly in terms of the Gram matrix K_{ij} of the outputs $\vec{Y}_i$. Expanding the terms on the right hand side, and enforcing the constraint that the outputs are centered on the origin, we obtain:

$$\mathcal{T}(Y) = \sum_i \left| \vec{Y}_i \right|^2 = \sum_i K_{ii} = \mathrm{Tr}(K). \tag{9}$$

Thus, we can interpret the objective function for the outputs in several ways: as a sum over pairwise distances in eq. (6), as a measure of variance in eq. (9), or as the trace of their Gram matrix in eq. (9). The second interpretation is reminiscent of PCA, but whereas in PCA we compute the linear projection that maximizes variance, here we compute the locally isometric embedding. Put another way, the objective function for maximizing variance remains the same; we have merely changed the allowed form of the dimensionality reduction. We also emphasize that in eq. (9), we are maximizing the trace, not minimizing it. While a standard relaxation to minimizing the rank [9] of a semidefinite matrix is to minimize its trace, the intuition here is just the opposite: we will obtain a low dimensional embedding by maximizing the trace of the Gram matrix.

Let us now collect the costs and constraints of this optimization. The problem is to maximize the variance of the outputs $\{\vec{Y}_i\}_{i=1}^N$ subject to the constraints that they are centered on the origin and locally isometric to the inputs $\{\vec{X}_i\}_{i=1}^N$. In terms of the input Gram matrix $G_{ij} = \vec{X}_i \cdot \vec{X}_j$, the output Gram matrix $K_{ij} = \vec{Y}_i \cdot \vec{Y}_j$ and the adjacency matrix η_{ij} indicating nearest neighbors, the optimization can be written as:

Maximize $\mathrm{Tr}(K)$ **subject to** $K \succeq 0$**,** $\sum_{ij} K_{ij} = 0$**,**
and $\forall ij$ **such that** $\eta_{ij} = 1$ **or** $[\eta^T \eta]_{ij} = 1$**,**
$K_{ii} + K_{jj} - K_{ij} - K_{ji} \;=\; G_{ii} + G_{jj} - G_{ij} - G_{ji}.$

This problem is an instance of semidefinite programming (SDP) [21]: the domain is the cone of semidefinite matrices intersected with hyperplanes (represented by equality constraints), and the objective function is linear in the matrix elements. The optimization is bounded above by eq. (8); it is also convex, thus eliminating the possibility of spurious local maxima. There exists a large literature on efficiently solving SDPs, as well as a number of general-purpose toolboxes. The results in this paper were obtained using the SeDuMi and CSDP 4.7 toolboxes [4, 18] in MATLAB.

3.3. Spectral Embedding

From the Gram matrix learned by semidefinite programming, we can recover the outputs $\vec{Y}_i$ by matrix diagonalization. Let $V_{\alpha i}$ denote the i^{th} element of the α^{th} eigenvector, with eigenvalue λ_α. Then the Gram matrix can be written as:

$$K_{ij} = \sum_{\alpha=1}^{N} \lambda_\alpha V_{\alpha i} V_{\alpha j}. \tag{10}$$

An N-dimensional embedding that is locally isometric to the inputs $\vec{X}_i$ is obtained by identifying the α^{th} element of the output $\vec{Y}_i$ as:

$$Y_{\alpha i} = \sqrt{\lambda_\alpha} V_{\alpha i}. \tag{11}$$

The eigenvalues of K are guaranteed to be nonnegative. Thus, from eq. (11), a large gap in the eigenvalue spectrum between the d^{th} and $(d+1)^{\text{th}}$ eigenvalues indicates that the inputs lie on or near a manifold of dimensionality d. In this case, a low dimensional embedding that is *approximately* locally isometric is given by truncating the elements of $\vec{Y}_i$. This amounts to projecting the outputs into the subspace of maximal variance, assuming the eigenvalues are sorted from largest to smallest. The quality of the approximation is determined by the size of the truncated eigenvalues; there is no approximation error for zero eigenvalues. The situation is analogous to PCA and MDS, but here the eigenvalue spectrum reflects the underlying dimensionality of a manifold, as opposed to merely a subspace.

The three steps of the algorithm, which we call Semidefinite Embedding (SDE), are summarized in Table 1. In its simplest formulation, the only free parameter of the algorithm is the number of nearest neighbors in the first step.

(I) Nearest Neighbors	Compute the k nearest neighbors of each input. Form the graph that connects each input to its neighbors and each neighbor to other neighbors of the same input.
(II) Semidefinite Programming	Compute the Gram matrix of the maximum variance embedding, centered on the origin, that preserves the distances of all edges in the neighborhood graph.
(III) Spectral Embedding	Extract a low dimensional embedding from the dominant eigenvectors of the Gram matrix learned by semidefinite programming.

Table 1. Steps of Semidefinite Embedding.

4. Results

We used several data sets of curves, surfaces, and images to evaluate the algorithm in Table 1 for low dimensional embedding of high dimensional inputs.

Fig. 1 shows $N = 800$ inputs sampled off a "Swiss roll" [19]. The inputs to the algorithm had $D=8$ dimensions, consisting of the three dimensions shown in the figure, plus five extra dimensions[1] filled with low variance Gaussian noise. The bottom plot of the figure shows the unfolded Swiss roll extracted from the Gram matrix learned by semidefinite programming. The top three eigenvectors are plotted, but the variance in the third dimension (shown to scale) is negligible. The eigenvalue spectrum in Fig. 7 reveals two dominant eigenvalues—a major eigenvalue, representing the unwrapped length of the Swiss roll, and a minor eigenvalue, representing its width. (The unwrapped Swiss roll is much longer than it is wide.) The other eigenvalues are nearly zero, indicating that SDE has discovered the true underlying dimensionality ($d=2$) of these inputs.

Fig. 2 shows another easily visualized example. The left plot shows $N = 539$ inputs sampled from a trefoil knot in $D = 3$ dimensions; the right plot shows the $d = 2$ embedding discovered by SDE using $k=4$ nearest neighbors. The color coding reveals that local neighborhoods have been preserved. In this case, the underlying manifold is a one-dimensional curve, but due to the cycle, it can only be represented in Euclidean space by a circle. The eigenvalue spectrum in Fig. 7 reveals two dominant eigenvalues; the rest are essentially zero, indicating the underlying (global) dimensionality ($d=2$) of the circle.

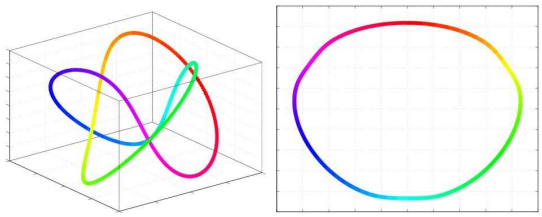

Figure 2. Left: $N=539$ inputs sampled along a trefoil knot in $D=3$ dimensions. Right: $d=2$ embedding computed by SDE using $k = 4$ nearest neighbors. The color coding shows that local neighborhoods are preserved.

Fig. 3 shows the results of SDE applied to color images of a three dimensional solid object. The images were created by viewing a teapot from different angles in the plane. The images have 76×101 pixels, with three byte color depth, giving rise to inputs of $D=23028$ dimensions. Though very high dimensional, the images in this data set are effectively parameterized by one degree of freedom—the angle of rotation. SDE was applied to $N=400$ images spanning 360 degrees of rotation, with $k=4$ nearest neighbors used to generate a connected graph. The two dimensional embedding discovered by SDE represents the rotating object as a circle—an intuitive result analogous to the embedding discovered for the trefoil knot. The eigenvalue spectrum of the Gram matrix learned by semidefinite programming is shown in Fig. 7; all but the first two eigenvalues are practically zero, indicating the underlying (global) dimensionality ($d=2$) of the circle.

1 For $K = 6$ nearest neighbors, the noise in extra dimensions helps to prevent the manifold from "locking up" when it is unfolded subject to the equality constraints in eqs. (1–3). Alternatively, the constraints in the SDP can be slightly relaxed by introducing slack variables.

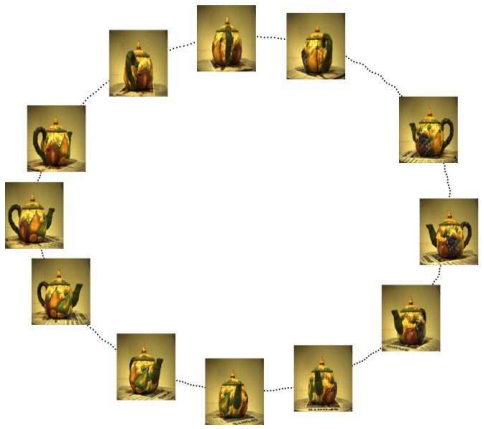

Figure 3. Two dimensional embedding of $N = 400$ images of a rotating teapot, obtained by SDE using $k = 4$ nearest neighbors. For this experiment, the teapot was rotated 360 degrees; the low dimensional embedding is a full circle. A representative sample of images are superimposed on top of the embedding.

Fig. 4 was generated from the same data set of images; however, for this experiment, only $N = 200$ images were used, sampled over 180 degrees of rotation. In this case, the eigenvalue spectrum from SDE detects that the images lie on a one dimensional curve (see Fig. 7), and the $d = 1$ embedding in Fig. 4 orders the images by their angle of rotation.

Fig. 5 shows the results of SDE on a data set of $N = 1000$ images of faces. The images contain different views and expressions of the same face. The images have 28×20 grayscale pixels, giving rise to inputs with $D = 560$ dimensions. The plot in Fig. 5 shows the first two dimensions of the embedding discovered by SDE, using $k = 4$ nearest neighbors. Interestingly, the eigenvalue spectrum in Fig. 7 indicates that most of the variance of the spectral embedding is contained in the first three dimensions.

Fig. 6 shows the results of SDE applied to another data set of images. In this experiment, the images were a subset of $N = 638$ handwritten TWOS from the USPS data set of handwritten digits [10]. The images have 16×16 grayscale pixels, giving rise to inputs with $D = 256$ dimensions. Intuitively, one would expect these images to lie on a low dimensional manifold parameterized by such features as size, slant, and line thickness. Fig. 6 shows the first two dimensions of the embedding obtained from SDE, with $k = 4$ nearest neighbors. The eigenvalue spectrum in Fig. 7 indicates a latent dimensionality significantly larger than two, but still much smaller than the actual number of pixels.

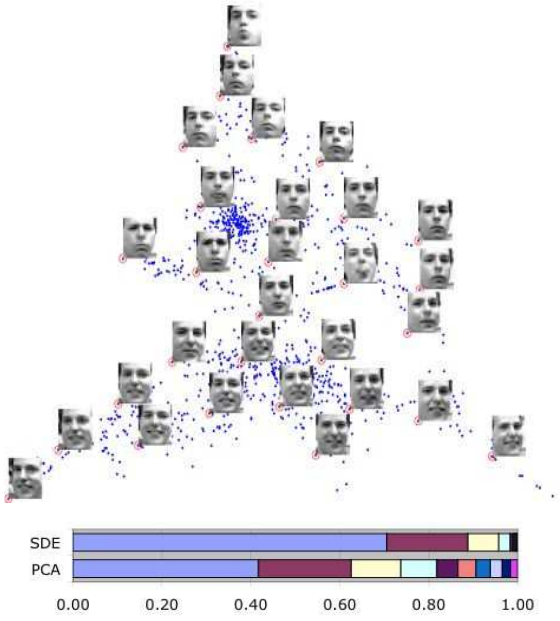

Figure 5. *Top:* **two dimensional embedding of $N = 1000$ images of faces, obtained by SDE using $k = 4$ nearest neighbors. Representative faces are shown next to circled points.** *Bottom:* **eigenvalues of SDE and PCA on this data set, indicating their estimates of the underlying dimensionality. The eigenvalues are shown as a percentage of the trace of the output Gram matrix for SDE and the trace of the input Gram matrix for PCA. The eigenvalue spectra show that most of the variance of the nonlinear embedding is confined to many fewer dimensions than the variance of the linear embedding.**

5. Discussion

The last few years have witnessed a number of developments in manifold learning. Recently proposed algorithms include Isomap [19], locally linear embedding (LLE) [14, 15], hessian LLE (hLLE) [8], and Laplacian eigenmaps [1]; there are also related algorithms for clustering [17]. All these algorithms share the same basic structure as SDE, consisting of three steps: (i) computing neighborhoods in the input space, (ii) constructing a square matrix with as many rows as inputs, and (iii) spectral embedding via the top or bottom eigenvectors of this matrix. SDE is based on a rather different geometric intuition, however, and as a result, it has different properties.

Comparing the algorithms, we find that each one attempts to estimate and preserve a different geometric signa-

Figure 4. One dimensional embedding of $N = 200$ images of a rotating teapot, obtained by SDE using $k = 4$ nearest neighbors. For this experiment, the teapot was only rotated 180 degrees. Representative images are shown ordered by their location in the embedding.

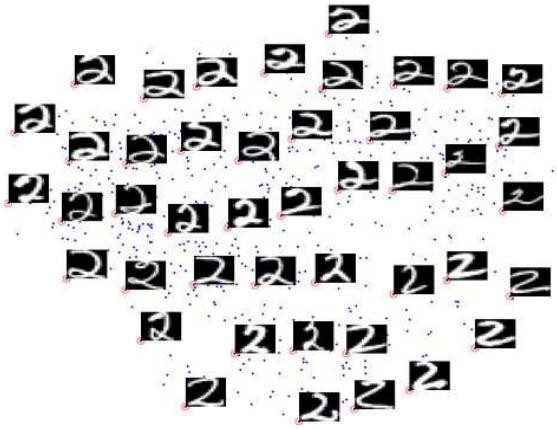

Figure 6. Results of SDE using $k = 4$ nearest neighbors on $N = 638$ images of handwritten TWOS. Representative images are shown next to circled points.

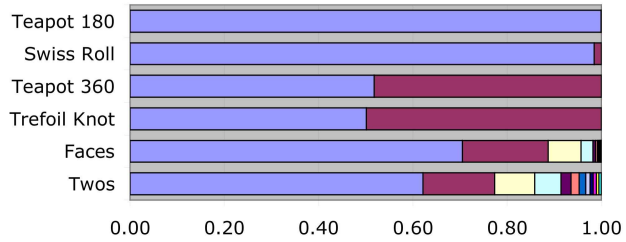

Figure 7. Eigenvalue spectra from SDE on the data sets in this paper. The eigenvalues are shown as a percentage of the trace of the Gram matrix learned by semidefinite programming. SDE identifies the correct underlying dimensionality of the Swiss roll, trefoil knot, and teapot data sets. The images of faces and handwritten digits give rise to many fewer non-zero eigenvalues than the actual number of pixels.

ture of the underlying manifold. Isomap estimates geodesic distances between inputs; LLE estimates the coefficients of local linear reconstructions; hLLE and Laplacian eigenmaps estimate the Hessian and Laplacian on the manifold, respectively; SDE estimates local angles and distances. Of these algorithms, only Isomap, hLLE, and SDE attempt to learn isometric embeddings; they are therefore the easiest to compare (since they seek the same solution, up to rotation and scaling). The results on the data set in Fig. 8 reveal some salient differences between these algorithms. While SDE and hLLE reproduce the original inputs up to isometry, Isomap fails in this example because the sampled manifold is not isometric to a *convex* subset of Euclidean space. (This is a key assumption of Isomap, one that is not satisfied by many image manifolds [7].) Moreover, comparing the eigenvalue spectra of the algorithms, only SDE detects the correct underlying dimensionality of the inputs; Isomap is foiled by non-convexity, while the eigenvalue spectra of LLE and hLLE do not reveal this type of information [8, 15].

Overall, the different algorithms for manifold learning should be viewed as complementary; each has its own advantages and disadvantages. LLE, hLLE, and Laplacian eigenmaps construct sparse matrices, and as a result, they are easier to scale to large data sets. On the other hand, their eigenvalue spectra do not reliably reveal the underlying dimensionality of sampled manifolds, as do Isomap and SDE. There exist rigorous proofs of asymptotic convergence for Isomap [7, 22] and hLLE [8], but not for the other algorithms. On the other hand, SDE by its very nature provides finite-size guarantees that its constraints will lead to locally isometric embeddings. We are not aware of any finite-size guarantees provided by the other algorithms, and indeed, the Hessian estimation in hLLE relies on numerical differencing, which can be problematic for small sample sizes. Finally, while the different algorithms have different computational bottlenecks, the second step in SDE (involving semidefinite programming) is more computationally demanding than the analogous steps in LLE and Isomap.

Our initial results for SDE appear promising. There are many important directions for future work. The most obvious is the investigation of faster methods for solving the semidefinite program in SDE. This study used a generic solver that did not exploit the special structure of the constraints. A data set with $N = 1000$ points (and $k = 4$) required about 30 minutes of computation time on a machine with a 2Ghz Pentium 4 processor. Data sets with up to $N = 1500$ points took several hours. A specialized

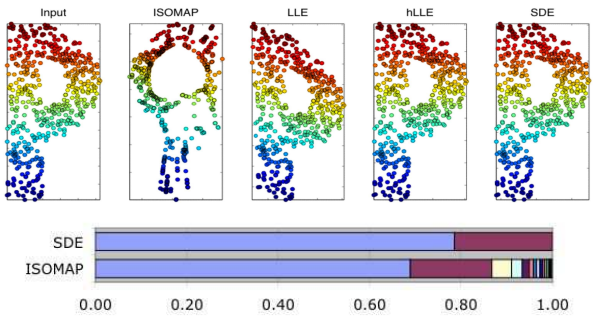

Figure 8. *Top:* **embedding of a non-convex two dimensional data set ($N = 500$) by different algorithms for manifold learning. Isomap, LLE, and hLLE were run with $k = 10$ nearest neighbors; SDE, with $k = 5$ nearest neighbors. Only hLLE and SDE reproduce the original inputs up to isometry.** *Bottom:* **only SDE has an eigenvalue spectrum that indicates the correct dimensionality ($d = 2$).**

solver should allow us to scale SDE up to larger data sets and larger neighborhood sizes. Another direction is relax the constraints in eqs. (1–3) by introducing slack variables. While slack variables do not change the basic structure of the semidefinite program, they may improve the robustness of the algorithm on small or noisy data sets. Other directions for future work include the investigation of image manifolds with different topologies [13] (such as those isometric to low dimensional spheres or torii), the extrapolation of results to out-of-sample inputs [2], and the relation of SDE to kernel methods for nonlinear dimensionality reduction [16]. Finally, as has been done for Isomap [7, 19, 22] and hLLE [8], it would be desirable to formulate SDE in the continuum limit and to construct rigorous proofs of asymptotic convergence. Such theoretical results would likely provide additional insight into the behavior of the algorithm.

References

[1] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[2] Y. Bengio, J.-F. Paiement, and P. Vincent. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2003. MIT Press.

[3] D. Beymer and T. Poggio. Image representation for visual learning. *Science*, 272:1905, 1996.

[4] D. B. Borchers. CSDP, a C library for semidefi nite programming. *Optimization Methods and Software 11(1):613-623*, 1999.

[5] M. Brand. Charting a manifold. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[6] T. Cox and M. Cox. *Multidimensional Scaling*. Chapman & Hall, London, 1994.

[7] D. L. Donoho and C. E. Grimes. When does Isomap recover the natural parameterization of families of articulated images? Technical Report 2002-27, Department of Statistics, Stanford University, August 2002.

[8] D. L. Donoho and C. E. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596, 2003.

[9] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734–4739, June 2001.

[10] J. J. Hull. A database for handwritten text recognition research. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

[11] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[12] H. Lu, Y. Fainman, and R. Hecht-Nielsen. Image manifolds. In N. M. Nasrabadi and A. K. Katsaggelos, editors, *Applications of Artificial Neural Networks in Image Processing III, Proceedings of SPIE*, volume 3307, pages 52–63, Bellingham, WA, 1998. SPIE.

[13] R. Pless and I. Simon. Embedding images in non-flat spaces. Technical Report WU-CS-01-43, Washington University, December 2001.

[14] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[15] L. K. Saul and S. T. Roweis. Think globally, fi t locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

[16] B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 888–905, August 2000.

[18] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization overy symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999.

[19] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[20] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[21] L. Vandenberghe and S. P. Boyd. Semidefi nite programming. *SIAM Review*, 38(1):49–95, March 1996.

[22] H. Zha and Z. Zhang. Isometric embedding and continuum Isomap. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 864–871, 2003.