International Journal of Computer Vision 70(1), 77–90, 2006
© 2006 Springer Science + Business Media, LLC. Manufactured in The Netherlands.
DOI: 10.1007/s11263-005-4939-z

Unsupervised Learning of Image Manifolds by Semidefinite Programming

KILIAN Q. WEINBERGER AND LAWRENCE K. SAUL
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104-6389
kilianw@cis.upenn.edu
lsaul@cis.upenn.edu

Received April 5, 2005; Revised August 12, 2005; Accepted September 9, 2005

First online version published in May, 2006

Abstract. Can we detect low dimensional structure in high dimensional data sets of images? In this paper, we propose an algorithm for unsupervised learning of image manifolds by semidefinite programming. Given a data set of images, our algorithm computes a low dimensional representation of each image with the property that distances between nearby images are preserved. More generally, it can be used to analyze high dimensional data that lies on or near a low dimensional manifold. 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.

Keywords: manifold learning, dimensionality reduction, semidefinite programming, kernel methods, data analysis, image manifolds, semidefinite embedding

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 (Burges, 2005) 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. The problem arises often in computer vision and pattern recognition (Lee et al., 2003; Pless, 2004; Elgammal and Lee, 2004; Souvenir and Pless, 2005). It is also of great interest to researchers in biological vision and computational neuroscience (Seung and Lee, 2000).

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 (Turk and Pentland, 1991; Beymer and Poggio, 1996). 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 (Lu et al., 1998). Though one can imagine other types of hidden structure in ensembles of images, such as clusters (Gordon et al., 2003) or parts (Lee and Seung, 1999), in this paper, we shall focus solely on the continuous structure that arises from 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 simply stated: given high dimensional data sampled from a low dimensional manifold, what can we infer about the structure of the manifold? In the last few years, researchers have uncovered a large family of graph-based algorithms for computing faithful low dimensional representations of high dimensional data. These so-called spectral methods compute low dimensional embeddings from the top or bottom eigenvectors of an appropriately constructed matrix. Algorithms in this family—including Isomap (Tenenbaum et al., 2000), locally linear (Roweis and Saul, 2000), Laplacian eigenmaps (Belkin and Niyogi, 2003), and others (Brand, 2003; Donoho and Grimes, 2003; Zhang and Zha, 2004)—can reveal low dimensional manifolds that are not detected by classical linear methods, such as principal component analysis (Jolliffe, 1986).

Our main contribution in this paper is a new algorithm for manifold learning based on semidefinite programming. Like other spectral methods for dimensionality reduction, it relies on efficient and tractable 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 well-known limitations of previous work. Our algorithm also reveals an interesting and unexpected connection to recent work on kernel methods in pattern recognition (Schölkopf and Smola, 2002).

The organization of this paper is as follows. In Section 2, we review classical methods for linear dimensionality reduction, then introduce the nonlinear transformations that we consider for unsupervised learning of image manifolds. In Section 3, we show how to formulate manifold learning as a highly tractable problem in semidef-inite programming; this leads to a simple algorithm for computing low dimensional mappings that preserve the distances between nearby data points. In Section 4, we present experimental results on several data sets, including easily visualized examples of curves and surfaces, as well as images of faces, handwritten digits, and solid objects. In Section 5, we show how to relax the distance-preserving constraints in the original formulation of the algorithm; these relaxations can lead to improved solutions for sparsely sampled or noisy data. Finally, in Section 6, we compare our algorithm to previous approaches in manifold learning and conclude by describing several directions for future work.

2. Dimensionality Reduction

We study dimensionality reduction as a problem in unsupervised learning. Given n high dimensional inputs $\mathbf{x}_i \in \mathcal{R}^p$ where $i \in \{1, 2, \ldots, n\}$, how can we compute outputs $\mathbf{y}_i \in \mathcal{R}^d$ in one-to-one correspondence with the inputs that provide a faithful representation in $d < p$ 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 $\mathbf{x}_i$.

Our algorithm for manifold learning builds on classical methods for dimensionality reduction (Jolliffe, 1986; Cox and Cox, 1994). We therefore begin by briefly reviewing the linear methods of principal component analysis (PCA) and metric multidimensional scaling (MDS). 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 their projected variance; the basis vectors of this subspace are given by the top eigenvectors of the $p \times p$ covariance matrix, $C = \frac{1}{n}\sum_i \mathbf{x}_i \mathbf{x}_i^T$. (Here and in what follows, we assume without loss of generality that the inputs are centered on the origin: $\sum_i \mathbf{x}_i = 0$.) In MDS with classical scaling, the inputs are projected into the subspace that best preserves their pairwise squared distances $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ or, more precisely, their dot products $\mathbf{x}_i \cdot \mathbf{x}_j$. The outputs of MDS are computed from the top eigenvectors of the $n \times n$ inner product matrix, with elements $G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$. Note that a set of vectors is determined up to rotation by its inner product matrix.

Though based on somewhat 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 inner product matrix of MDS have the same rank and eigenvalues up to a constant factor. Both matrices are positive semidefinite, and gaps in their eigenvalue spectra indicate that the high dimensional inputs $\mathbf{x}_i \in \mathcal{R}^p$ 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 projections 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*

If PCA and MDS are spectral methods for linear dimensionality reduction, 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 (Tenenbaum et al., 2000; Donoho and Grimes, 2003) to Section 6. 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 (Donoho and Grimes, 2002).

Isometry is a relation between manifolds, but we can extend the notion in a natural way to data sets. Consider two data sets $\{\mathbf{x}_i\}_{i=1}^n$ and $\{\mathbf{y}_i\}_{i=1}^n$ that are in one-to-one correspondence. We will say that the data sets are k-locally isometric if for every point $\mathbf{x}_i$, there exists a rotation and translation that maps $\mathbf{x}_i$ and its k nearest neighbors $\{\mathbf{x}_{j1}, \mathbf{x}_{j2}, \ldots, \mathbf{x}_{jk}\}$ precisely onto the points $\mathbf{y}_i$ and $\{\mathbf{y}_{j1}, \mathbf{y}_{j2}, \ldots\ldots, \mathbf{y}_{jk}\}$. With this definition, we can distinguish between linear methods for dimensionality reduction, such as PCA and MDS, and the nonlinear method proposed in this paper. Whereas the linear methods compute mappings that aim to preserve Euclidean distances between all pairs of data points, our method considers the much larger class of nonlinear transformations that only preserve the geometric properties of local neighborhoods. We formalize this approach as an algorithm in the next section.

We will often refer to the outputs $\mathbf{y}_i$ as providing a low dimensional "embedding" of the high dimensional inputs $\mathbf{x}_i$. By this, we mean that the outputs provide an explicit low dimensional vector representation of the inputs that faithfully preserves local distances. As in previous work on "manifold learning" (Tenenbaum et al., 2000; Roweis and Saul, 2000; Belkin and Niyogi, 2003), it should be emphasized that our algorithm takes as input not the underlying manifold—which may not be known a priori—but a discrete set of high dimensional vectors. From this input, the algorithm constructs a weighted neighborhood graph and produces vector outputs that locally respect the properties of this graph. In the literature on manifold learning, it is common to refer to the outputs as an "embedding" rather than an "immersion", even though the latter better reflects the mathematical usage of these terms for manifolds. In particular, the algorithm does not necessarily output an embedding of the underlying manifold in the sense of the Nash embedding theorems (Nash, 1956).

3. Maximum Variance Unfolding

Taking as a starting point the notion of local isometry, we can now formulate the problem of manifold learning more precisely. In particular, given n inputs $\mathbf{x}_i \in \mathcal{R}^p$, can we find n outputs $\mathbf{y}_i \in \mathcal{R}^d$, where $d < p$, such that the inputs and outputs are k-locally isometric, or at least approximately so? In this section, exploiting the observation that the outputs are determined up to rotation by their inner product matrix $K_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$, we shall show how to express this problem as a constrained optimization over the cone of positive semidefinite matrices.

Like PCA and MDS, the algorithm we propose for manifold learning is based on a simple geometric intuition. Imagine each input $\mathbf{x}_i$ 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. Figure 1 shows the unraveling effect of this transformation on inputs sampled from a "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 notion of isometry between discrete point sets in Section 2.2 can be translated into various sets of equality constraints on the inputs $\{\mathbf{x}_i\}_{i=l}^n$ and the outputs $\{\mathbf{y}_i\}_{i=1}^n$. To begin, note that neighborhoods of inputs and outputs will be related by translation and rotation if and only if all the distances and angles between points and their neighbors are preserved. Thus, whenever both $\mathbf{x}_j$ and $\mathbf{x}_k$ are neighbors of $\mathbf{x}_i$, for local isometry we must have that:

$$(\mathbf{y}_i - \mathbf{y}_j) \cdot (\mathbf{y}_i - \mathbf{y}_k) = (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_i - \mathbf{x}_k). \quad (1)$$

Equation (1) is sufficient for local isometry because the triangle formed by any point and its neighbors is

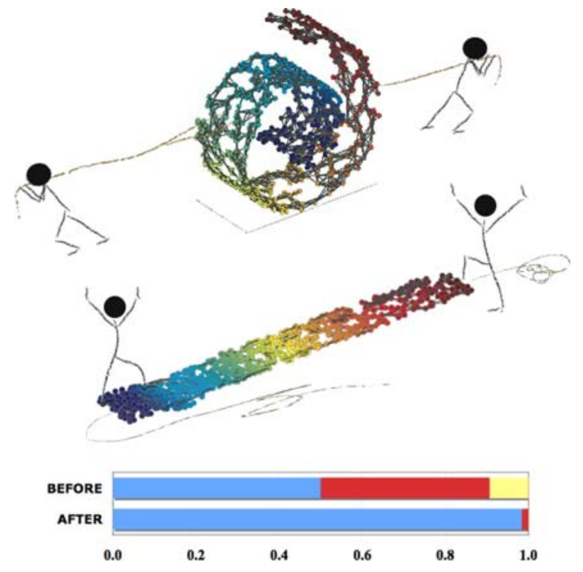

Figure 1. Maximum variance unfolding of $n = 800$ data points sampled with noise from a "Swiss roll". *Top*: A discretized manifold is revealed by forming the graph that pairwise connects each data point and its $k = 6$ nearest neighbors. *Middle*: The data set is unraveled by maximizing its variance subject to the constraint that distances along edges in the graph are preserved. *Bottom*: Eigenvalues of the data set's Gram matrix, before and after unfolding, shown as a fraction of the trace. The number of appreciable eigenvalues reveals the dimensionality of the subspace in which most of the data's variance is concentrated.

determined up to rotation 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 $\{\mathbf{x}_i\}_{i=1}^n$ and $\{\mathbf{y}_i\}_{i=1}^n$ are locally isometric if whenever $\mathbf{x}_i$ and $\mathbf{x}_j$ are themselves neighbors or common neighbors of another point in the data set, we have:

$$\|\mathbf{y}_i - \mathbf{y}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2 . \quad (2)$$

This is an equivalent characterization of local isometry as Eq. (1), but expressed only in terms of pairwise distances.

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 simpler to deal only with constraints on distances, as opposed to angles. To this end, we can further connect the graph by adding edges between the neighbors of each node (if they do not already exist). In other words, we create a fully connected clique of size $k + 1$ out of every input $\mathbf{x}_i$ and its k nearest neighbors. By preserving the distances along edges in this new graph (see Fig. 2), we preserve both the distances along edges and the angles between edges in the original graph—because if all sides of a triangle are preserved, so are its angles.

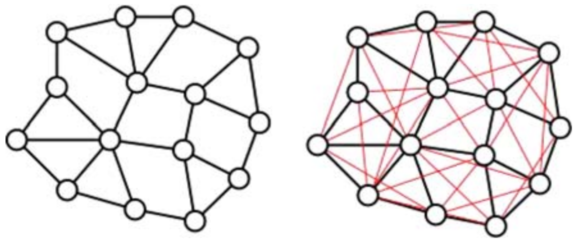

Figure 2. In the left graph, each node is connected to its $k = 3$ nearest neighbors; in the right graph, additional edges (in light gray) also directly connect the neighbors. Preserving the distances along edges in the right graph is equivalent to preserving the distances along edges and the angles between edges in the left graph.

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

$$\sum_i \mathbf{y}_i = \mathbf{0} . \quad (3)$$

Eq. (3) simply removes a translational degree of freedom from the final solution.

3.2. *Optimization*

What objective function 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 $\mathbf{y}_i$ that unfold a manifold sampled by inputs $\mathbf{x}_i$. In particular, we propose to maximize the sum of pairwise squared distances between outputs:

$$\Phi = \frac{1}{2n} \sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 . \tag{4}$$

By maximizing Eq. (4), we pull the outputs as far apart as possible, *subject to the constraints in the previous section.*

We can verify that this objective function 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, for any input $\mathbf{x}_i$, let $\mathcal{N}_i$ be the set of indices of its k nearest neighbors. Let τ be the maximal distance between any two neighbors:

$$\tau = \max_{i,\, j\, \in\, N_1} \|\mathbf{x}_i - \mathbf{x}_j\|. \tag{5}$$

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 $\mathbf{y}_i$ and $\mathbf{y}_j$, we must have $||\mathbf{y}_i - \mathbf{y}_j|| \leq n\tau$. Using this to provide an upper bound on the objective function in Eq. (4), we obtain:

$$\Phi \leq \frac{1}{2n} \sum_{ij} (n\tau)^2 = \frac{n^3\tau^2}{2} . \tag{6}$$

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

Let us now collect the costs and constraints of the optimization to maximize the variance of the outputs $\{\mathbf{y}_i\}_{i=1}^n$ subject to the constraints that they are centered on the origin and locally isometric to the inputs $\{\mathbf{x}_i\}_{i=1}^n$. Let $\eta_{ij} \in \{0, 1\}$ indicate whether there is an edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the graph formed by pairwise connecting all k-nearest neighbors. Then, in terms of the squared distance matrix $D_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2$, the optimization can be written as:

Maximize $\sum_{ij} \|\mathbf{y}_i - \mathbf{y}_j||^2$ **subject to**:
(1) $\sum_i \mathbf{y}_i = 0$.
(2) $||\mathbf{y}_i - \mathbf{y}_j||^2 = D_{ij}$ **for all** (i, j) **with** $\eta_{ij} = 1$.

The optimization as stated above is not convex, as it involves maximizing a quadratic form subject to quadratic equality constraints.

We can obtain a simpler optimization by reformulating the problem in terms of the elements of the inner product matrix, $K_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$. As mentioned previously, the matrix K_{ij} determines the outputs up to rotation. The distance and centering constraints in Eqs. (2) and (3) are easily expressed in terms of these matrix elements. For example, expanding the square in Eq. (2) and substituting $K_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$, we obtain:

$$K_{ii} - 2K_{ij} + K_{jj} = D_{ij} \tag{7}$$

Likewise, the centering constraint in Eq. (3) can be expressed in terms of these inner products as:

$$0 = \left\| \sum_i \mathbf{y}_i \right\|^2 = \sum_{ij} \mathbf{y}_i \cdot \mathbf{y}_j = \sum_{ij} K_{ij} . \tag{8}$$

Note that both Eqs. (7) and (8) are linear equality constraints on the elements of K_{ij}. Writing the constraints in this way, and noting that the outputs are determined up to rotation by their inner products, we may view our original problem as an optimization over inner product matrices K_{ij} rather than vectors $\mathbf{y}_i$. Not all matrices, however, can be interpreted as inner product matrices: only symmetric matrices with nonnegative eigenvalues can be interpreted in this way. Thus, we must further constrain the optimization to the cone of symmetric positive semidefinite matrices (Vandenberghe and Boyd, 1996). We can write this constraint as

$$K \succeq 0. \tag{9}$$

In sum, there are three types of constraints on the inner product matrix K_{ij}, arising from local isometry, centering, and positive semide finiteness. 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 original inner product matrix $G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$ (assuming that the inputs $\mathbf{x}_i$ are centered on the origin).

Finally, we need to express the objective function in Eq. (4) directly in terms of the inner product

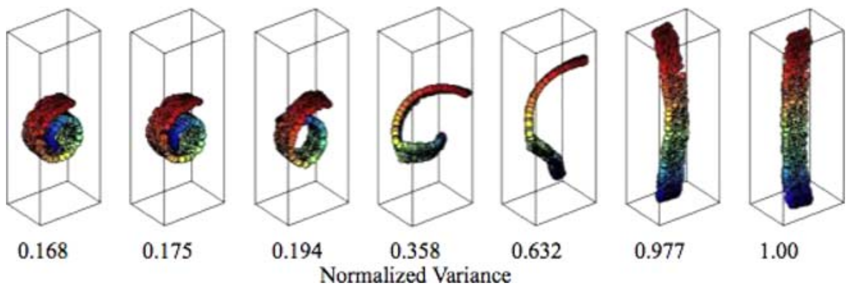

Figure 3. Inputs sampled from a Swiss roll are "unfolded" by maximizing their variance subject to constraints that preserve local distances and angles. The middle snapshots show various feasible (but non-optimal) solutions obtained during intermediate iterations of the optimization described in Section 3.2.

matrix K_{ij} of the outputs $\mathbf{y}_i$. Expanding the terms on the right hand side, and enforcing the constraint that the outputs are centered on the origin, we obtain:

$$\Phi = \frac{1}{2n}\sum_{ij}(\|\mathbf{y}_i\|^2 + \|\mathbf{y}_j\|^2 + 2\mathbf{y}_i\cdot\mathbf{y}_j)\,, \tag{10}$$

$$= \sum_i \|\mathbf{y}_i\|^2\,, \tag{11}$$

$$= \sum_i K_{ii}\,, \tag{12}$$

$$= \mathrm{Tr}\,(K). \tag{13}$$

Thus, we can interpret the objective function for the outputs in several ways: as a sum over pairwise distances in Eq. (4), as a measure of variance in Eq. (11), or as the trace of their inner product matrix in Eq. (13). The second interpretation is reminiscent of PCA, but whereas in PCA we compute the linear projection that maximizes variance, here we compute the k-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. (13), we are maximizing the trace, not minimizing it. While a standard relaxation to minimizing the rank (Fazel et al., 2001) 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 inner product matrix. Figure 3 illustrates the connection between increasing variance and reducing dimensionality. The images in this figure were obtained from the intermediate solutions of a variance-maximizing optimization subject to centering and local distance constraints.

Collecting the costs and constraints of the above optimization in terms of the inner product matrix K_{ij}, we can write the problem as follows:

Maximize trace (K) subject to:
(1) $K \succeq 0$.
(2) $\sum_{ij} K_{ij} = 0$.
(3) $K_{ii} - 2K_{ij} + K_{jj} = D_{ij}$ **for all** (i, j) **with** $\eta_{ij} = 1$.

This problem is an instance of semidefinite programming (SDP) (Vandenberghe and Boyd, 1996): the domain is the cone of positive 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. (6); it is also convex, thus eliminating the possibility of spurious local maxima. The problem is guaranteed to be feasible because the constraints are trivially satisfied by the Gram matrix $G_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j$ of the inputs (assuming that the inputs are centered). 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 CSDP v4.9 toolbox (Borchers, 1999) in MATLAB.

3.3. *Spectral Decomposition*

From the inner product matrix learned by semidefinite programming, we can recover the outputs $\mathbf{y}_i$ by matrix diagonalization. Let $V_{\alpha i}$ denote the ith element of the αth eigenvector, with eigenvalue λ_α. Then the inner product matrix can be written as:

$$K_{ij} = \sum_{\alpha=1}^{n} \lambda_\alpha V_{\alpha i} V_{\alpha j}. \tag{14}$$

An n-dimensional embedding that is k-locally isometric to the inputs $\mathbf{x}_i$ is obtained by identifying the αth element of the output $\mathbf{y}_i$ as:

$$Y_{\alpha i} = \sqrt{\lambda_\alpha V_{\alpha i}}. \tag{15}$$

The eigenvalues of K are guaranteed to be nonnegative. Thus, from Eq. (15), a large gap in the eigenvalue spectrum between the dth and $(d+1)$th eigenvalues indicates that the outputs lie in or near a subspace of dimensionality d. In this case, a low dimensional embedding that is *approximately* locally isometric is given by truncating the elements of $\mathbf{y}_i$. 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 dimensionality of an underlying manifold, as opposed to merely a subspace.

The three steps of the algorithm 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 (though one can imagine more elaborate schemes for determining neighborhoods). The second step of the algorithm, involving semidefinite programming, is the most computationally intensive.

Table 1. The three steps of maximum variance unfolding (MVU), involving nearest neighbor search, semidefinite programming, and matrix diagonalization.

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

The first and third steps resemble those of other algorithms for manifold learning, discussed in Section 6; our algorithm has rather different properties, however, due to the particular nature of its second step. In earlier work (Weinberger and Saul, 2004; Weinberger et al., 2004), we referred to this algorithm as "semidefinite embedding", but here we will adopt the name "maximum variance unfolding" (MVU), coined by others (Sun et al., 2004), as it is both more descriptive and less likely to be confused with other work in graph embedding that makes use of semidefinite programming (Biswas and Ye, 2003; Sha and Saul, 2005).

4. Experimental Results

We evaluated the algorithm in Table 1 on several high dimensional data sets whose inputs were either explicitly sampled or believed to have been sampled from a low dimensional manifold.

Figure 1 shows the maximum variance unfolding of $n = 800$ inputs sampled from a "Swiss roll". The inputs had $p = 8$ dimensions, consisting of the three dimensions shown in the top panel of the figure, plus five extra dimensions filled with low variance Gaussian noise. The middle panel of the figure shows the unfolded Swiss roll computed by semidefinite programming: the solution preserves distances between $k = 6$ nearest neighbors. The eigenvalues of the inner product matrix (before and after unfolding) are shown in the bottom panel, normalized by their sum. There are 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 the algorithm has discovered the correct dimensionality $(d = 2)$ of the underlying manifold.

Figure 4 shows another easily visualized example. The top left panel shows $n = 1617$ inputs sampled from a trefoil knot in $p = 3$ dimensions; the top right panel shows the maximum variance unfolding that preserves distances between $k = 4$ nearest neighbors. The color coding reveals that local neighborhoods have been preserved. The eigenvalue spectrum in the bottom panel reveals two dominant eigenvalues; the rest are essentially zero, again indicating the correct dimensionality $(d = 2)$ of the underlying manifold—in this case, an annulus.

The bottom section of Fig. 5 shows the maximum variance unfolding of color images of a three

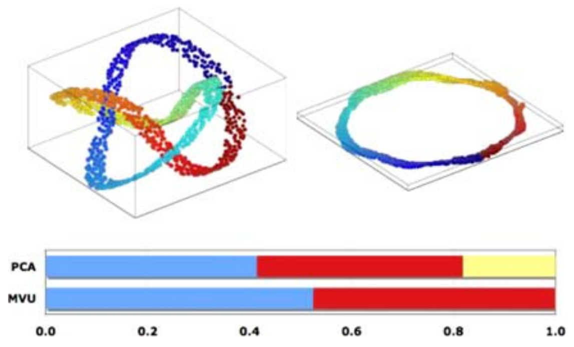

Figure 4. *Top left*: $n = 1617$ inputs sampled from a trefoil knot in $p = 3$ dimensions. *Top right*: two dimensional embedding computed by MVU with $k = 5$ nearest neighbors. The color coding shows that the embedding preserves local neighborhoods. *Bottom*: eigenvalues from the matrices of PCA and MVU, shown as a fraction of the trace.

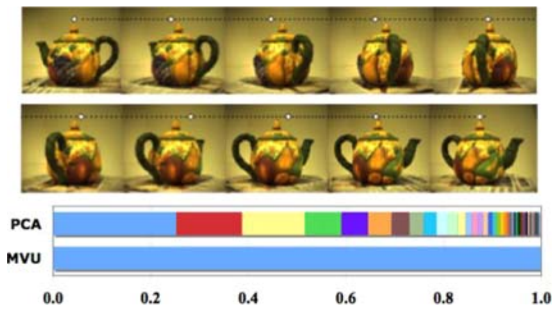

Figure 6. *Top*: one dimensional embedding of $n = 200$ images of a rotating teapot, computed by MVU with $k = 4$ nearest neighbors. For this experiment, the teapot was only rotated 180 degrees. Representative images are shown ordered by their location in the embedding. *Bottom*: eigenvalues from the matrices of PCA and MVU, shown as a fraction of the trace.

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 $p = 23028$ dimensions. Though very high dimensional, the images in this data set are effectively parameterized by one degree of freedom—the angle of rotation. MVU 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 MVU represents the rotating object as a circle—an intuitive result analogous to the embedding discovered for the trefoil knot. The eigenvalue spectrum of the inner product matrix learned by semidefinite programming is shown in the bottom panel; all but the first two eigenvalues are practically zero, indicating the global dimensionality ($d = 2$) of the underlying circle.

Figure 6 was generated from the same data set of images, but using only $n = 200$ images that were sampled over 180 degrees of rotation. In this case, the eigenvalue spectrum from MVU detects that the images lie on a one dimensional curve, and the one dimensional embedding orders the images by their angle of rotation.

Figure 7 shows the maximum variance unfolding of another data set of images. In this experiment, the images were a subset of $n = 953$ handwritten TWOS from the USPS data set of handwritten digits (Hull, 1994). The images have 16×16 grayscale pixels, giving rise to inputs with $p = 256$ dimensions. Intuitively, one would expect these images to lie on a low dimensional manifold parameterized by such features as size, slant, and line thickness. The top panel of Fig. 7 shows the first two dimensions of the embedding computed by MVU with $k = 4$ nearest neighbors. The eigenvalue spectrum in the bottom panel indicates a latent dimensionality significantly larger than two, but still much smaller than the number of significant principal components.

Finally, Fig. 8 shows the two dimensional embedding of a data set of images of rotating globes. The embedding was computed by MVU with $k = 4$ nearest neighbors. The inputs consisted of $n = 900$ color images at 47×47 pixel resolution, corrupted by white

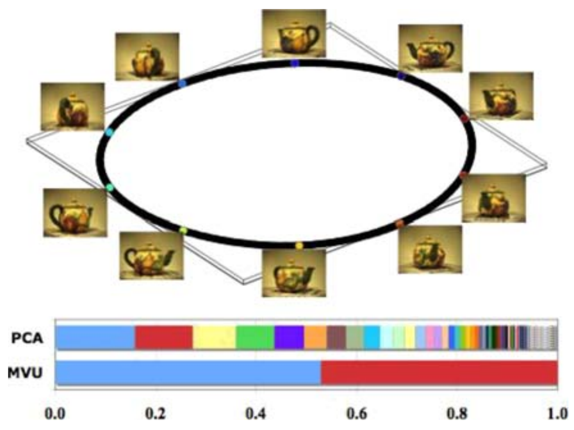

Figure 5. *Top*: two dimensional embedding of $n = 400$ images of a rotating teapot, computed by MVU with $k = 4$ nearest neighbors. For this experiment, the teapot was rotated 360 degrees; the low dimensional embedding is a full circle. A representative sample of images are superimposed on top of the embedding. *Bottom*: eigenvalues from the matrices in PCA and MVU, shown as a fraction of the trace.

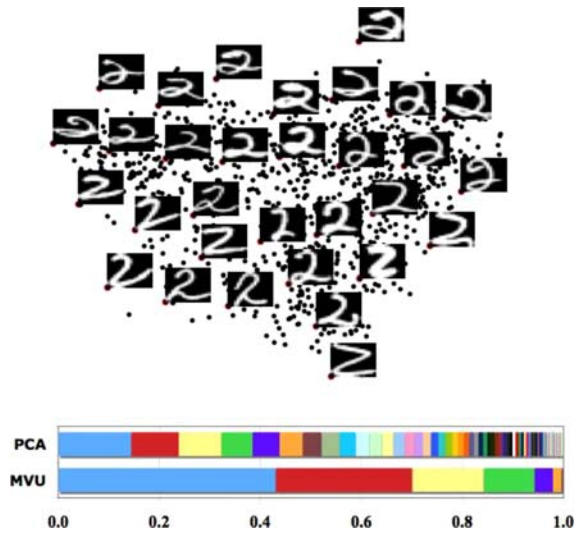

Figure 7. *Top:* Maximum variance unfolding of $n = 953$ images of handwritten TWOS with $k = 4$ nearest neighbors. Representative images are shown next to circled points. *Bottom:* eigenvalues from the matrices of PCA and MVU, shown as a fraction of the trace.

Gaussian noise. The standard deviation of the noise was equal to 30% of the pixel intensity range; for comparison, clean and noisy images are shown to the left and right of the embedding. The viewpoints were evenly sampled over 30 degrees longitude and latitude.

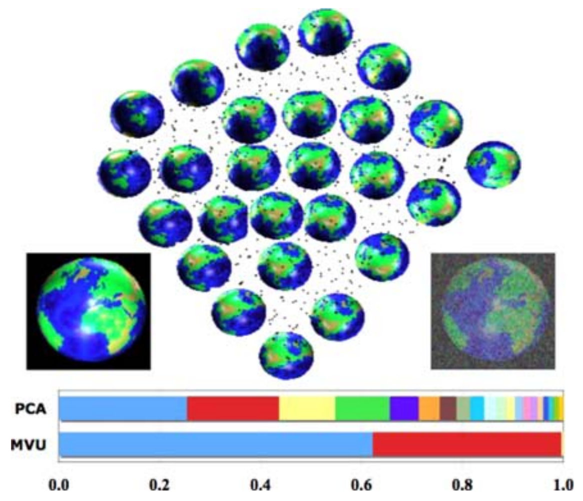

Figure 8. *Top*: Two dimensional embedding of $n = 900$ noisy images of a rotating globe, computed by MVU with $k = 4$ nearest neighbors. The viewpoints were evenly spaced over 30 degrees of longitude and latitude. The color images were corrupted by white Gaussian noise before unfolding the data set to test the algorithm's robustness. Clean and noisy images are shown (enlarged) to the left and right of the embedding. *Bottom*: eigenvalues from the matrices of PCA and MVU, shown as a fraction of the trace.

The diamond-shaped embedding, with representative (clean) images superimposed, reveals the two degrees of freedom corresponding to longitude and latitude. The eigenvalue spectrum in the bottom panel also indicates the correct dimensionality ($d = 2$) of the underlying manifold.

5. Relaxing the Constraints

It is often desirable to relax the constraints in Eq. (2) such that local distances are not strictly preserved. In some applications, for example, these distances only provide a rough estimate of proximity relations. Relaxing the constraints always leads to solutions that have equal or greater variance; the resulting inner product matrices also tend to have fewer numbers of appreciable eigenvalues. Thus, the relaxed optimizations can be viewed as an option for more aggressive forms of dimensionality reduction.

One simple way to relax the constraints is to replace the strict equalities in Eq. (2) by inequalities. This allows the distances between nearest neighbors to shrink, but not to grow. As the optimization in Eq. (4) is based on maximizing variance, it acts to minimize the slack in the inequalities even when these constraints are not enforced as strict equalities. The optimization in this case is given by:

Maximize trace (K) subject to:

(1) $K \succeq 0$.

(2) $\sum_{ij} K_{ij} = 0$.

(3) $K_{ii} - 2K_{ij} + K_{jj} \le D_{ij}$ **for all** (i, j) **with** $\eta_{ij} = 1$.

Interestingly, the dual of this semidefinite program (with inequalities rather than equalities) arises in the calculation of fastest mixing Markov processes on weighted graphs (Sun et al., 2004).

Figure 9 shows the maximum variance unfolding of a data set of $n = 1960$ images of faces, with inequality constraints from $k = 4$ nearest neighbors. The images contain different views and expressions of the same face. The images have 28×20 grayscale pixels, giving rise to inputs with $p = 560$ dimensions. The top panel in the figure shows a three dimensional embedding of these images. The bottom panel shows the eigenvalue spectra of the matrices from PCA and MVU with both strict equality and relaxed inequality constraints. Note

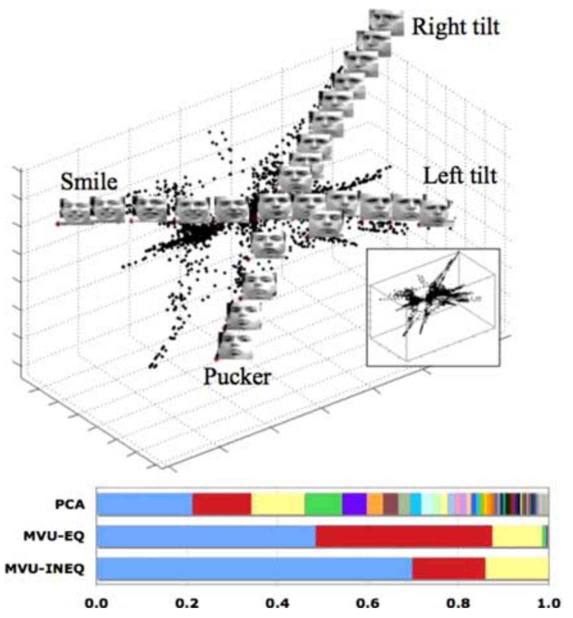

Figure 9. *Top*: Three dimensional embedding of $n = 1960$ images of faces, obtained by MVU using $k = 4$ nearest neighbors (and inequality constraints). Representative faces are shown next to circled points. *Bottom*: eigenvalues from the matrices of PCA and MVU with strict equality and relaxed inequality constraints, shown as a fraction of the trace. The relaxed solution concentrates more of its variance in the leading three dimensions.

how the relaxed optimization concentrates more variance in the leading three dimensions.

Another way to relax the constraints in maximum variance unfolding is to allow the distances to change slightly, but to add a term to the objective function that penalizes this slack. This can be done by introducing slack variables ξ_{ij}, one for each of the constraints in Eq. (2). The relaxed optimization is given by:

Maximize Tr $(K) - \omega \sum_{ij} \eta_{ij} \|\xi_{ij}\|$ **subject to**:

(1) $K \succeq 0$.

(2) $\sum_{ij} K_{ij} = 0$.

(3) $K_{ii} - 2K_{ij} + K_{jj} = D_{ij} + \xi_{ij}$ **for all** (i, j) **with** $\eta_{ij} = 1$.

The constant $\omega > 0$ balances the objectives of maximizing variance and penalizing slack. In practice, choosing $\omega \gg 1$ seems to work well for balancing these two objectives.

Figure 10 shows an example where this use of slack allows the algorithm to return an improved result. In this figure, the algorithm was applied with and without slack variables to $n = 800$ inputs sampled *without noise* from a Swiss roll. When distances are strictly preserved between $k = 6$ nearest neighbors, the outputs "lock up" before the data set is completely unfolded; when slack is allowed, however, we obtain the expected result. (Note that the earlier result in Fig. 1 was obtained from inputs with five extra dimensions filled with Gaussian noise; the noise makes the problem less rigid.) The main disadvantage of slack variables is the extra computation required to optimize an objective function with $O(nk^2)$ additional variables. The resulting optimization can be noticeably slower.

The above relaxations may prove particularly useful in applications when the distances D_{ij} are not computed from Euclidean distances but are specified in some other way. In this case, the optimization with strict distance-preserving constraints may not be feasible. No matter what the distances D_{ij}, the relaxed versions of maximum variance unfolding always have non-empty feasible regions containing the trivial solution $K_{ij} = 0$. In such problems, the optimization can be used to return the variance-maximizing embedding in Euclidean space that best preserves local distances.

Finally, we mention one other type of relaxation that has proved useful in an application of maximum variance unfolding to language data (Blitzer et al., 2005). One obtains a simpler problem in semidefinite programming, with many fewer constraints, by only preserving distances to k-nearest neighbors as opposed to preserving distances and angles. The resulting optimization can be used to unfold larger data sets; it also tends to lead to more aggressive forms of dimensionality reduction.

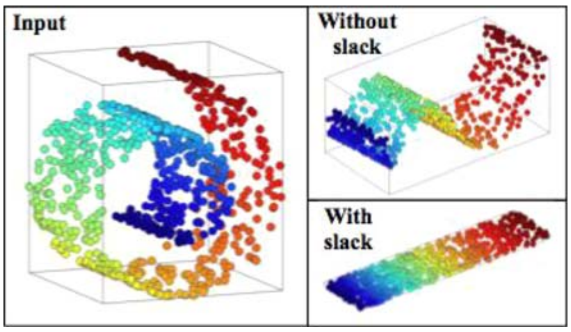

Figure 10. Maximum variance unfolding of $n = 800$ inputs sampled without noise from a Swiss roll. Strictly preserving distances between $k = 6$ nearest neighbors causes the outputs to "lock up" before the data set is completely unfolded. Allowing slack in these constraints leads to the desired solution.

6. Discussion

In this paper we have proposed an algorithm for unsuper-vised learning of image manifolds by semidefinite programming. Our work can be viewed as bridging two ongoing lines of research in machine learning, one on spectral methods for dimensionality reduction, the other on kernel methods for pattern recognition. After placing our work in the context of these two lines of research, we conclude by describing some open questions and plans for future work.

6.1. Spectral Methods

The last few years have witnessed a number of developments in spectral methods for dimensionality reduction and manifold learning. Recently proposed algorithms include Isomap (Tenenbaum et al., 2000), locally linear embedding (LLE) (Roweis and Saul, 2000), hessian LLE (hLLE) (Donoho and Grimes, 2003), and Laplacian eigen-maps (Belkin and Niyogi, 2003); there are also related algorithms for clustering (Shi and Malik, 2000; Ng et al., 2002). Maximum variance unfolding is based on a different geometric intuition than these other algorithms, however, and as a result, it has somewhat different properties. The rest of this section highlights the similarities and differences with previous work.

Most spectral methods for dimensionality reduction all share the same basic structure: (i) computing neighborhoods in the input space, (ii) constructing a square matrix with as many rows as inputs, and (iii) deriving a low dimensional embedding from the top or bottom eigenvectors of this matrix. Algorithms differ in the geometric signatures of manifolds that they attempt to estimate and preserve. For example, Isomap is based on geodesic distances, LLE on the coefficients of local linear reconstructions, and Laplacian eigenmaps on the discrete graph Laplacian. Maximum variance unfolding is based on estimating and preserving local distances and angles. It is most easily compared to Isomap and hLLE, since both these algorithms also attempt to learn isometric mappings and thus seek the same solution, up to a global rotation.

On many problems, Isomap and maximum variance unfolding return generally similar results, with eigenvalue spectra that provide a reliable estimate of the data set's intrinsic dimensionality. Figure 11 illustrates an example where the algorithms return quite different results. The top panel shows the results of Isomap applied to the same data set of teapot images as in Fig. 5;

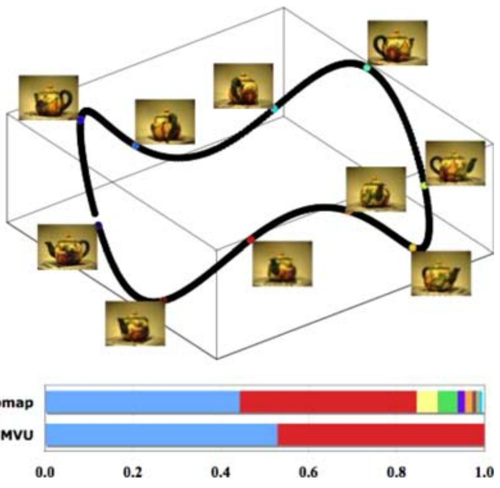

Figure 11. *Top*: The three dimensional embedding of $n = 400$ images of a rotating teapot, computed by Isomap with $k = 10$ nearest neighbors. *Bottom*: The normalized eigenvalues of Isomap in comparison to maximum variance unfolding. Both return two dominant eigenvalues. However, as the data set is not isometric to a convex subset of Euclidean space, Isomap returns more than two nonzero eigenvalues. This is reflected through the artificial wave in the third dimension.

the bottom panel compares its eigenvalue spectrum to that of maximum variance unfolding. Note that Isomap has more than two appreciable eigenvalues (which is manifested by the waves in its three dimensional embedding). This somewhat artificial result is due to the fact that the sampled manifold in this example 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 (Donoho and Grimes, 2002).

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 (Saul and Roweis, 2003), as do Isomap and maximum variance unfolding. There exist convergence proofs for Isomap (Tenenbaum et al., 2000; Donoho and Grimes, 2002; Zha and Zhang, 2003) and hLLE (Donoho and Grimes, 2003), but not for the other algorithms. On the other hand, maximum variance unfolding 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. Finally, while the different algorithms have different computational bottlenecks, it is fair to say that the approach in this paper, based on semidefinite programming, is the most computationally demanding.

6.2. Kernel Methods

Along with the growing interest in manifold learning, the last few years have also witnessed an explosion of interest in kernel methods for pattern recognition (Schölkopf and Smola, 2002). Kernel methods rely on an implicit mapping of inputs to a higher (and potentially infinite) dimensional feature space. The kernel function specifies the dot product between the feature vectors formed in this way from the original inputs. The "kernel trick" is to replace the dot products $\mathbf{x}_i \cdot \mathbf{x}_j$ that appear in linear algorithms for pattern recognition by the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$. Support vector machines (Vapnik, 1998) for classification and kernel PCA (Schölkopf et al., 1998) for nonlinear dimensionality reduction are examples of algorithms that were conceived in this way, with a kernel matrix that stores the pairwise dot products between inputs in feature space. In most kernel machines, the kernel function is simply specified a priori; the most popular choices involve polynomial and Gaussian kernels. Recent work in supervised learning, however, has investigated the possibility of learning kernel matrices by semidefinite programming (Lanckriet et al., 2002).

The inner product matrix K_{ij} computed by maximum variance unfolding can be viewed as a kernel matrix between inputs. While there have been attempts to interpret the matrices constructed by Isomap and LLE as kernels (Schölkopf and Smola, 2002; Bengio et al., 2004; Ham et al., 2003), their interpretation is less straightforward (Weinberger et al., 2004). The kernel matrix in maximum variance unfolding is interesting in several respects. First, it is based on variance maximization, as opposed to margin maximization (Schölkopf and Smola, 2002; Lanckriet et al., 2002); the former applies to unsupervised learning, whereas the latter requires (at least some) labeled examples. Second, whereas most kernel functions are chosen to map the inputs into a higher dimensional feature space, the kernel matrix in maximum variance unfolding does just the opposite, typically mapping the inputs into a lower dimensional space. Finally, maximum variance unfolding may be viewed as a special version of kernel PCA (Schölkopf et al., 1998)—ideally suited for manifold discovery—in which the kernel matrix itself is learned (in a completely unsupervised manner) from unlabeled examples. Attempts to use the kernel matrix from maximum variance unfolding in support vector machines have not met with much success (Weinberger et al., 2004); standard polynomial and radial basis function kernels almost always yield better performance. Maximizing variance subject to neighborhood-preserving constraints does not seem to be the best intuition for problems in classification.

6.3. Conclusion

Our initial results for unsupervised learning of image manifolds seem promising. The algorithm in this paper has different properties than previous algorithms for manifold learning, and many of these properties can be construed as advantages. Like Isomap (and unlike LLE), its eigenvalue spectrum provides an estimate of the underlying dimensionality of sampled manifolds; unlike Isomap, however, it does not assume that the inputs are isometric to a convex subset of Euclidean space. The distance-preserving constraints in maximum variance unfolding can also be relaxed to encourage more aggressive solutions for dimensionality reduction.

The use of semidefinite programming has both advantages and disadvantages. Its main advantage is the simple manner in which we can express and enforce distance-preserving constraints. Such constraints can be tailored to particular applications of nonlinear dimensionality reduction. For example, a recent extension of maximum variance unfolding (Bowling et al., 2005) focused on images collected by a robot moving through a virtual environment. In this application, the robot moved only according to a finite set of actions, and it was assumed that transitions caused by the same actions should be mapped to jumps of equal distance in the embedding space. These additional constraints, naturally accommodated by the use of semidefinite programming, lead to Action Respecting Embeddings (ARE) that more accurately reflect the robot's motion and provide better support for path planning and localization.

The main disadvantage of semidefinite programming is the required amount of computation, which scales as $O(n^3 + c^3)$, where n is the matrix size and c is the number of constraints (Borchers, 1999). While our current implementation can handle data sets with up to $n \approx 2500$ inputs, many applications in computer vision and pattern recognition involve much larger data sets. Thus our main task for future research is to develop shortcuts and approximations for large-scale implementations.

We are currently investigating a framework that has allowed us to reproduce many of the results in this paper in a small fraction of the original time (Weinberger et al., 2005). The framework is based on the observation that for well-sampled manifolds, the entire output inner product matrix $K_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$ can be very accurately reconstructed from a much smaller submatrix of inner products between randomly chosen *landmarks*. In particular, we can write:

$$K \approx QLQ^T \tag{16}$$

where L is the $m \times m$ submatrix of inner products between landmarks (with $m \ll n$) and Q is an $n \times m$ linear transformation derived from solving a sparse set of linear equations. The factorization in (16) enables us to reformulate the semidefinite program in terms of the much smaller matrix L, yielding order-of-magnitude reductions in computation time. We are also exploring other ideas based on low-rank matrix factorizations (Burer and Monteiro, 2003), distributed methods (Biswas and Ye, 2003), the extrapolation of kernel matrices to out-of-sample inputs (Bengio et al., 2004), and the post-processing of results from faster but less robust methods for manifold learning (Sha and Saul, 2005).

In addition to the challenge of large-scale implementations, there are many other directions for future work. It would be interesting to investigate image manifolds which have spherical or toroidal geometries (Pless and Simon, 2002). It might also be fruitful to study problems where more sophisticated similarity metrics (Simard et al., 1993; Belongie et al., 2002) have been developed using prior knowledge, rather than relying on naive nearest-neighbor computations using Euclidean distance. Finally, to the extent the our approach provides a new connection between work in manifold learning and kernel methods, we hope that it will lead to further advances in both areas.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0238323.

References

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

Belongie, S., Malik, J., and Puzicha, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522.

Bengio, Y., Paiement, J.F., and Vincent, P. 2004. Out-of-sample extensions for LLE, Isomap, MDS, eigenmaps, and spectral clustering. *Advances in Neural Information Processing Systems* 16. Cambridge, MA: MIT Press.

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

Biswas, P. and Ye, Y. 2003. A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization. Stanford University, Department of Electrical Engineering, working paper.

Blitzer, J., Weinberger, K.Q., Saul, L.K., and Pereira, F.C.N. 2005. Hierarchical distributed representations for statistical language modeling. *Advances in Neural and Information Processing Systems*. Cambridge, MA: MIT Press.

Borchers, B. 1999. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623.

Bowling, M., Ghodsi, A., and Wilkinson, D. 2005. Action respecting embedding. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML 2005)*. Bonn, Germany.

Brand, M. 2003. Charting a manifold. *Advances in Neural Information Processing Systems 15*, Cambridge, MA: MIT Press, pp. 985–992.

Burer, S. and Monteiro, R. 2003. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming, Series*, B95, pp. 329–357.

Burges, C.J.C. 2005. Geometric methods for feature extraction and dimensional reduction. In L. Rokach and O. Maimon (eds.), *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic Publishers.

Cox, T. and Cox, M. 1994. *Multidimensional Scaling*. London: Chapman and Hall.

Donoho, D.L. and Grimes, C.E. 2002. *When Does Isomap Recover The Natural Parameterization of Families of Articulated Images*? (Technical Report 2002–27). Department of Statistics. Stanford University.

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

Elgammal, A. and Lee, C. 2004. Separating style and content on a nonlinear manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, Washington DC, pp. 478–485.

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

Gordon, S., Goldberger, J., and Greenspan, H. 2003. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *Proceedings of the Ninth International Conference on Computer Vision (ICCV 2003)*, pp. 370–377.

Ham, J., Lee, D.D., Mika, S., and Schölkopf, B. 2003. *A kernel view of the dimensionality reduction of manifolds* (Technical Report TR-110). Max-Planck-Institut für Biologische Kybernetik, Tübingen.

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

Jolliffe, I.T. 1986. *Principal Component Analysis*. New York: Springer-Verlag.

Lanckriet, G.R.G., Christianini, N., Bartlett, P.L., Ghaoui, L.E., and Jordan, M.I. 2002. Learning the kernel matrix with semidefinite programming. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML 2002)*, pp. 323–330.

Lee, D.D., and Seung, H.S. 1999. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791.

Lee, K., Ho, J., Yang, M.-H., and Kriegman, D. 2003. Video-based face recognition using probabilistic appearance manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, pp. 313–320.

Lu, H., Fainman, Y., and Hecht-Nielsen, R. 1998. Image manifolds. *Applications of Artificial Neural Networks in Image Processing III, Proceedings of SPIE*, Bellingham, WA: SPIE, pp. 52–63.

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

Ng, A.Y., Jordan, M., and Weiss, Y. 2002. On spectral clustering: analysis and an algorithm. *Advances in Neural Information Processing Systems* 14 Cambridge, MA: MIT Press, pp. 849–856.

Pless, R. 2004. Differential structure in non-linear image embedding functions. *Proceedings of the IEEE Workshop on Articulated and Non-Rigid Motion*, Washington, D.C. pp. 10–17.

Pless, R. and Simon, I. 2002. Embedding images in non-flat spaces. In *Proceedings of the Conference on Imaging Science Systems and Technology* pp. 182–188.

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

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

Schölkopf, B. and Smola, A.J. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press.

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

Seung, H.S. and Lee, D.D. 2000. The manifold ways of perception. *Science*, 290:2268–2269.

Sha, F. and Saul, L.K. 2005. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *Proceedings of the Twenty-second International Conference on Machine Learning (ICML 2005)*. Bonn, Germany.

Shi, J. and Malik, J. 2000. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pp. 888–905.

Simard, P.Y., LeCun, Y., and Decker, J. 1993. Efficient pattern recognition using a new transformation distance. *Advances in Neural Information Processing Systems*, San Mateo, CA: Morgan Kaufman, pp. 50–58.

Souvenir, R. and Pless, R. 2005. Isomap and non-parametric models of image deformation. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, Breckenridge, CO., pp. 195–200.

Sun, J., Boyd, S., Xiao, L., and Diaconis, P. 2004. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. Submitted to *SIAM Review*.

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

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

Vandenberghe, L. and Boyd, S.P. 1996. Semidefinite programming. *SIAM Review*, 38(1):49–95.

Vapnik, V. 1998. *Statistical Learning Theory*. N.Y.: Wiley.

Weinberger, K.Q., Packer, B.D., and Saul, L.K. 2005. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*. Barbados, West Indies.

Weinberger, K.Q. and Saul, L.K. 2004. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-04)*, Washington D.C, pp. 988–995.

Weinberger, K.Q., Sha, F., and Saul, L.K. 2004. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, Banff, Canada, pp. 839–846.

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

Zhang, Z. and Zha, H. 2004. Principal manifolds and nonlinear dimensionality reduction by local tangent space alignment. *SIAM Journal of Scientific Computing*, 26:313–338.