Journal of Machine Learning Research 4 (2003) 119-155 Submitted 6/02; Published 6/03

Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds

Lawrence K. Saul LSAUL@CIS.UPENN.EDU
Department of Computer and Information Science
University of Pennsylvania
200 South 33rd Street
557 Moore School - GRW
Philadelphia, PA 19104-6389, USA

Sam T. Roweis ROWEIS@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road
Pratt Building 283
Toronto, Ontario M5S 3G4, CANADA

Editor: Yoram Singer

Abstract

The problem of dimensionality reduction arises in many fields of information processing, including machine learning, data compression, scientific visualization, pattern recognition, and neural computation. Here we describe locally linear embedding (LLE), an unsupervised learning algorithm that computes low dimensional, neighborhood preserving embeddings of high dimensional data. The data, assumed to be sampled from an underlying manifold, are mapped into a single global coordinate system of lower dimensionality. The mapping is derived from the symmetries of locally linear reconstructions, and the actual computation of the embedding reduces to a sparse eigenvalue problem. Notably, the optimizations in LLE—though capable of generating highly nonlinear embeddings—are simple to implement, and they do not involve local minima. In this paper, we describe the implementation of the algorithm in detail and discuss several extensions that enhance its performance. We present results of the algorithm applied to data sampled from known manifolds, as well as to collections of images of faces, lips, and handwritten digits. These examples are used to provide extensive illustrations of the algorithm's performance—both successes and failures—and to relate the algorithm to previous and ongoing work in nonlinear dimensionality reduction.

1. Introduction

Many problems in machine learning begin with the preprocessing of raw multidimensional signals, such as images of faces or spectrograms of speech. The goal of preprocessing is to obtain more useful representations of the information in these signals for subsequent operations such as classification, denoising, interpolation, visualization, or outlier detection. In the absence of prior knowledge, such representations must be learned or discovered automatically. Automatic methods which discover hidden structure from the statistical regularities of large data sets can be studied in the general framework of *unsupervised learning* (Hinton and Sejnowski, 1999).

Two main goals have been proposed for algorithms in unsupervised learning: *density estimation* and *dimensionality reduction*. The goal of density estimation is to learn the parameters of a probabilistic model that can be used to predict or assess the novelty of future observations. The goal of dimensionality reduction is to obtain more compact representations of the original data that capture the information necessary for higher-level decision making. A recent trend in machine learning has been to pursue these goals simultaneously, with probabilistic generative models of raw sensory inputs whose hidden variables represent low dimensional degrees of freedom (Attias, 1999; Dayan et al., 1995; Hyvärinen, 1998; Roweis, 1998; Roweis et al., 2002; Tipping and Bishop, 1999). However, the goal of dimensionality reduction can also be pursued in a non-probabilistic and non-parametric setting. This is the approach taken here, leading to an efficient eigenvector method for nonlinear dimensionality reduction of large data sets.

Our work addresses a longstanding problem at the intersection of geometry and statistics: to compute a low dimensional embedding of high dimensional data sampled (with noise) from an underlying manifold. Many types of high dimensional data can be characterized in this way—for example, images generated by different views of the same three dimensional object. Beyond applications in machine learning and pattern recognition, the use of low dimensional manifolds to represent continuous percepts is also a recurring theme in computational neuroscience (Seung and Lee, 2000). The goal of our algorithm is to learn such representations from examples: to discover—in a general setting, without the use of a priori knowledge—the few degrees of freedom that underlie observed modes of continuous variability.

Two canonical forms of dimensionality reduction are the eigenvector methods of principal component analysis (PCA) (Jolliffe, 1986) and multidimensional scaling (MDS) (Cox and Cox, 1994). Most applications of PCA and MDS involve the modeling of linear variabilities in multidimensional data. In PCA, one computes the linear projections of greatest variance from the top eigenvectors of the data covariance matrix. In MDS, one computes the low dimensional embedding that best preserves pairwise distances between data points. If these distances correspond to Euclidean distances — the case of so-called classical scaling — then the results of MDS are equivalent to PCA (up to a linear transformation). Both methods are simple to implement, and their optimizations are well understood and not prone to local minima. These virtues account for the widespread use of PCA and MDS, despite their inherent limitations as linear methods.

Recently, we introduced a more powerful eigenvector method—called locally linear embedding (LLE)—for the problem of *nonlinear dimensionality reduction* (Roweis and Saul, 2000). This problem is illustrated by the manifolds in Figure 1. In these examples, dimensionality reduction by LLE succeeds in recovering the underlying manifolds, whereas linear embeddings by PCA or MDS map faraway data points to nearby points in the plane, creating distortions in both the local and global geometry. Like PCA and MDS, our algorithm is simple to implement, and its optimizations do not involve local minima. Unlike these methods, however, it is capable of generating highly nonlinear embeddings, and its main optimization involves a *sparse* eigenvalue problem that scales well to large, high dimensional data sets.

Note that mixture models for local dimensionality reduction (Fukunaga and Olsen, 1971; Ghahramani and Hinton, 1996; Kambhatla and Leen, 1997), which cluster the data and perform PCA within each cluster, do not address the problem considered here—namely, how to map high dimensional data into a single global coordinate system of lower dimensionality. In particular, while such models can be used to discover clusters in high dimensional data and to model their density,

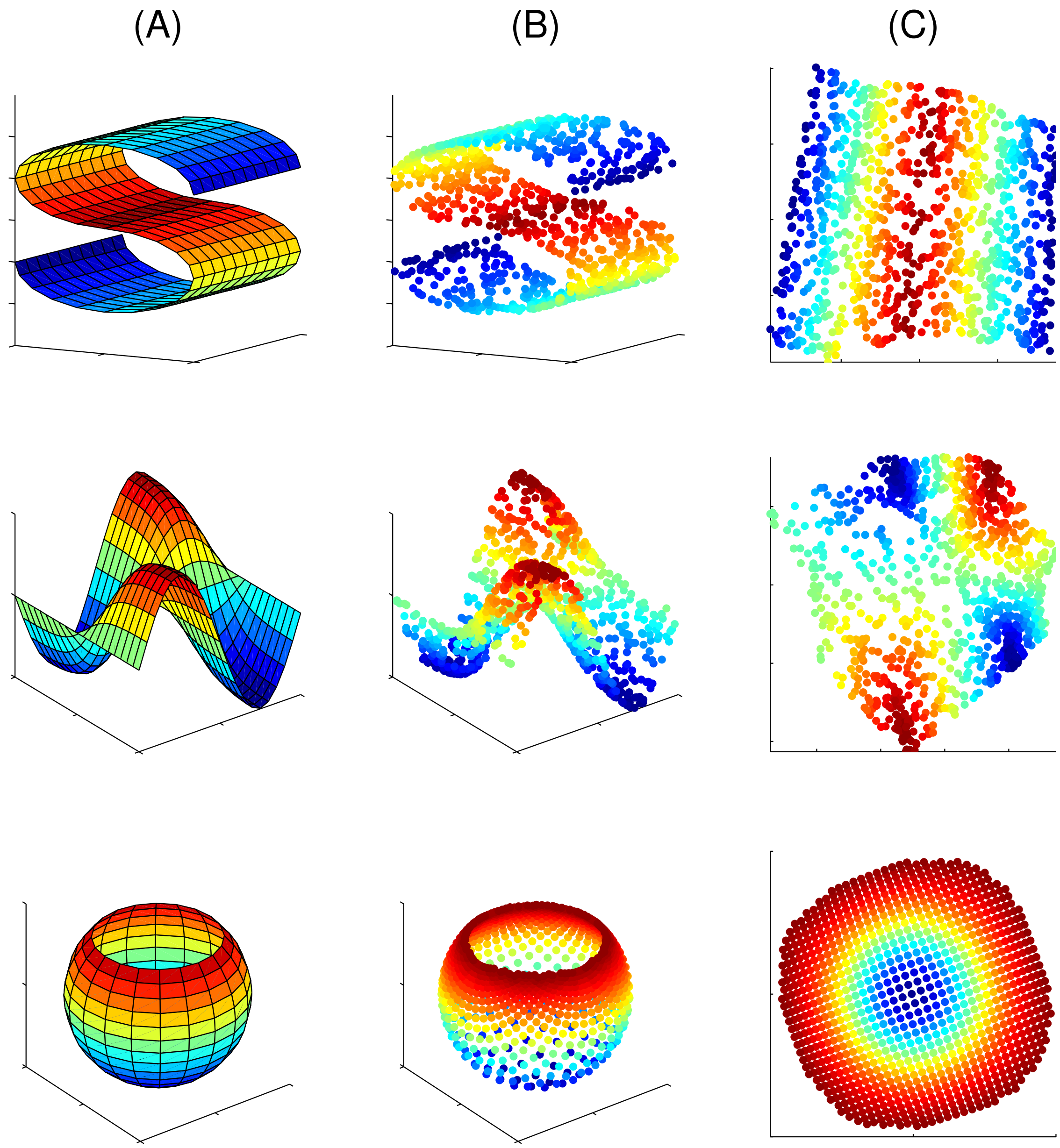

Figure 1: The problem of nonlinear dimensionality reduction, as illustrated for three dimensional data (B) sampled from two dimensional manifolds (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without external signals that suggest how the data should be embedded in two dimensions. The LLE algorithm described in this paper discovers the neighborhood-preserving mappings shown in (C); the color coding reveals how the data is embedded in two dimensions.

they could not be used to compute the two dimensional embeddings in the rightmost panels of Figure 1.

In this paper, we describe the LLE algorithm, providing significantly more examples and details of its implementation than can be found in our earlier work (Roweis and Saul, 2000). We also discuss a number of extensions that enhance its performance. The organization of this paper is as follows: In Section 2, we describe the algorithm in general terms, focusing on its main procedures and geometric intuitions. In Section 3, we illustrate the algorithm's performance on several problems of different size and dimensionality. In Section 4, we discuss the most efficient ways to implement the algorithm and provide further details on the nearest neighbor search, least squares optimization, and eigenvalue problem. In Sections 5 and 6, we describe a number of extensions to LLE, including how to estimate and/or enforce the dimensionality of discovered manifolds, as well as how to derive mappings that generalize the results of LLE to examples outside the training set. Finally, in Section 7, we compare LLE to other eigenvector-based methods for clustering and nonlinear dimensionality reduction (Belkin and Niyogi, 2002; Ng et al., 2002; Schölkopf et al., 1998; Shi and Malik, 2000; Tenenbaum et al., 2000; Weiss, 1999) and mention several directions for future work.

2. Algorithm

The LLE algorithm, summarized in Figure 2, is based on simple geometric intuitions. Essentially, the algorithm attempts to compute a low dimensional embedding with the property that *nearby points in the high dimensional space remain nearby and similarly co-located with respect to one another in the low dimensional space*. Put another way, the embedding is optimized to preserve the local configurations of nearest neighbors. As we shall see, under suitable conditions, it is possible to derive such an embedding solely from the geometric properties of nearest neighbors in the high dimensional space. Indeed, the LLE algorithm operates entirely without recourse to measures of distance or relation between faraway data points.

To begin, suppose the data consist of N real-valued vectors $\vec{X}_i$ (or *inputs*), each of dimensionality D, sampled from a smooth underlying manifold. Provided there is sufficient data (such that the manifold is well-sampled), we expect each data point and its neighbors to lie on or close to a locally linear patch of the manifold. More precisely, by "smooth" and "well-sampled" we mean that for data sampled from a d-dimensional manifold, the curvature and sampling density are such that each data point has on the order of $2d$ neighbors which define a roughly linear patch on the manifold with respect to some metric in the input space. Under such conditions, we can characterize the local geometry in the neighborhood of each data point by linear coefficients that reconstruct the data point from its neighbors. The LLE algorithm derives its name from the nature of these reconstructions: it is *local*, in the sense that only neighbors contribute to each reconstruction, and *linear*, in the sense that reconstructions are confined to linear subspaces.

In the simplest formulation of LLE, one identifies K nearest neighbors per data point, as measured by Euclidean distance. (More sophisticated neighborhood criteria are discussed in Section 4.) Reconstruction errors are then measured by the cost function:

$$\mathcal{E}(W) = \sum_i \left| \vec{X}_i - \sum\nolimits_j W_{ij} \vec{X}_j \right|^2 , \tag{1}$$

1. **Compute the neighbors of each data point, $\vec{X}_i$.**

2. **Compute the weights W_{ij} that best reconstruct each data point $\vec{X}_i$ from its neighbors, minimizing the cost in Equation (1) by constrained linear fits.**

3. **Compute the vectors $\vec{Y}_i$ best reconstructed by the weights W_{ij}, minimizing the quadratic form in Equation (2) by its bottom nonzero eigenvectors.**

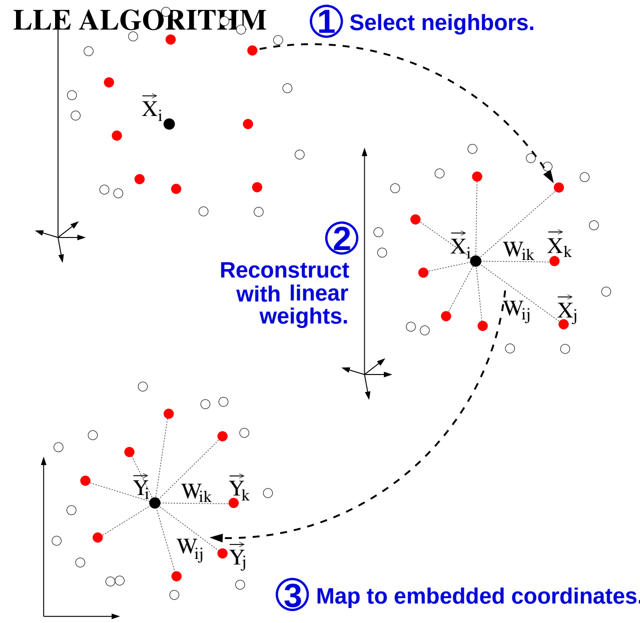

Figure 2: Summary of the LLE algorithm, mapping high dimensional inputs $\vec{X}_i$ to low dimensional outputs $\vec{Y}_i$ via local linear reconstruction weights W_{ij}.

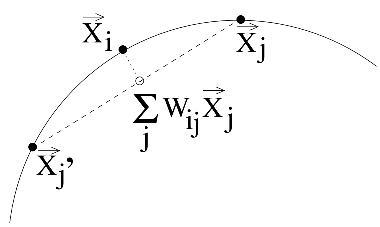

Figure 3: A data point $\vec{X}_i$, its neighbors $\vec{X}_j$, and its locally linear reconstruction $\sum_j W_{ij}\vec{X}_j$. The reconstruction weights are constrained to satisfy $\sum_j W_{ij} = 1$.

which adds up the squared distances between all the data points and their reconstructions. The weight W_{ij} summarizes the contribution of the jth data point to the ith reconstruction. To compute the weights, we minimize the cost function in Equation (1) subject to two constraints: a *sparseness* constraint and an *invariance* constraint. The sparseness constraint is that each data point $\vec{X}_i$ is reconstructed only from its neighbors, enforcing $W_{ij} = 0$ if $\vec{X}_j$ does not belong to this set. The invariance constraint is that the rows of the weight matrix sum to one: $\sum_j W_{ij} = 1$. The reason for this latter constraint will become clear shortly. The optimal weights W_{ij} subject to these constraints are found by solving a set of constrained least squares problems, as discussed further in Section 4.

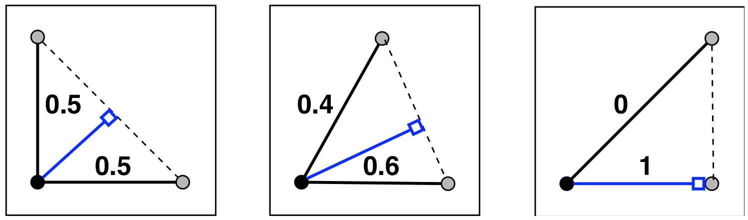

Figure 4: The effects of shearing on reconstruction weights. Shown are the optimal weights that linearly reconstruct the black point in terms of its gray neighbors; the resulting reconstructions are indicated by white squares. Note that the optimal weights and linear reconstructions are not invariant to shear transformations.

Note that the constrained weights that minimize these reconstruction errors obey several important symmetries: for any particular data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors.[1] The invariance to rotations and rescalings follows immediately from the form of Equation (1); the invariance to translations is enforced by the sum-to-one constraint on the rows of the weight matrix. A consequence of these symmetries is that the reconstruction weights characterize geometric properties that do not depend on a particular frame of reference. Note that the optimal weights and linear reconstructions for each data point are *not* in general invariant to local affine transformations, such as shears. (See Figure 4 for a counterexample.)

Suppose the data lie on or near a manifold of dimensionality $d \ll D$. To a good approximation, then, we imagine that there exists a linear mapping—consisting of a translation, rotation, and rescaling—that maps the high dimensional coordinates of each neighborhood to global internal coordinates on the manifold. By design, the reconstruction weights W_{ij} reflect those geometric properties of the data that are invariant to exactly such transformations. We therefore expect their characterization of local geometry in the input space to be equally valid for local patches on the manifold. In particular, the same weights W_{ij} that reconstruct the input $\vec{X}_i$ in D dimensions should also reconstruct its embedded manifold coordinates in d dimensions.

(Informally, imagine taking a pair of scissors, cutting out locally linear patches of the underlying manifold, and arranging them in the low dimensional embedding space. If the transplantation of each patch involves no more than a translation, rotation, and rescaling, then the angles between data points on the patch will be preserved. It follows that when the patch arrives at its low dimensional destination, the same weights will provide the optimal reconstruction of each data point from its neighbors.)

LLE constructs a neighborhood preserving mapping based on the above idea. In the third and final step of the algorithm, each high dimensional input $\vec{X}_i$ is mapped to a low dimensional *output* $\vec{Y}_i$ representing global internal coordinates on the manifold. This is done by choosing the d-dimensional coordinates of each output $\vec{Y}_i$ to minimize the embedding cost function:

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2. \tag{2}$$

1. Naturally, they are also invariant to global rotations, translations and homogeneous rescalings of all the inputs, but the invariance to local transformations has more far-reaching implications.

This cost function—like the previous one—is based on locally linear reconstruction errors, but here we fix the weights W_{ij} while optimizing the outputs $\vec{Y}_i$. Note that the embedding is computed directly from the weight matrix W_{ij}; the original inputs $\vec{X}_i$ are not involved in this step of the algorithm. Thus, the embedding is determined entirely by the geometric information encoded by the weights W_{ij}. Our goal is to find low dimensional outputs $\vec{Y}_i$ that are reconstructed by the *same* weights W_{ij} as the high dimensional inputs $\vec{X}_i$.

The embedding cost function in Equation (2) defines a quadratic form in the outputs $\vec{Y}_i$. Subject to constraints that make the problem well-posed, the cost function has a unique global minimum. This unique solution for the outputs $\vec{Y}_i$ is the result returned by LLE as the low dimensional embedding of the high dimensional inputs $\vec{X}_i$. The embedding cost function can be minimized by solving a sparse $N \times N$ eigenvalue problem. Details of this eigenvalue problem are discussed in Section 4. There, we show that the bottom $d+1$ non-zero eigenvectors of an easily computed *cost matrix* provide an ordered set of d embedding coordinates.

Note that while the reconstruction weights for each data point are computed from its local neighborhood—independent of the weights for other data points—the embedding coordinates are computed by an $N \times N$ eigensolver, a global operation that couples all data points (or more precisely, all data points that lie in the same connected component of the graph defined by the neighbors). This is how the algorithm discovers global structure—by integrating information from overlapping local neighborhoods. Note also that the d^{th} coordinate output by LLE always corresponds to the $(d+1)^{st}$ smallest eigenvector of the cost matrix, regardless of the total number of outputs requested. Thus, the LLE coordinates are ordered or "nested"; as more dimensions are added to the embedding space, the existing ones do not change.

Implementation of the algorithm is straightforward. In the simplest formulation of LLE, there exists only one free parameter: the number of neighbors per data point K (or any equivalent neighborhood-determining parameter, such as the radius of a ball to be drawn around each point). Once neighbors are chosen, the optimal weights W_{ij} and outputs $\vec{Y}_i$ are computed by standard methods in linear algebra, as detailed in Section 4. The algorithm involves a single pass through the three steps in Figure 2 and finds global minima of the reconstruction and embedding costs in Equations (1) and (2). No learning rates or annealing schedules are required during the optimization, and no random initializations or local optima affect the final results.

3. Examples

The embeddings discovered by LLE are easiest to visualize for data sampled from two dimensional manifolds. In Figure 1, for example, the input to LLE consisted of $N = 1000$ data points sampled from the manifolds shown in panel (A). The resulting embeddings show how the algorithm, using $K = 8$ neighbors per data point, faithfully maps these manifolds to the plane.

The example in the bottom row of Figure 1 shows that, under the right conditions, LLE can learn the stereographic mapping from the sphere to the plane. For the algorithm to succeed in this case, a neighborhood of the north pole must be excluded, and the data must be sampled uniformly in manifold (stereographic) coordinates (which corresponds to increasing density as one approaches the north pole in the input space). This example suggests that LLE can recover conformal mappings—mappings which locally preserve angles, but not distances. Such a conjecture is also motivated by the invariance of the reconstruction weights in Equation (1) to translations, rotations, *and scalings* of local neighborhoods. Nevertheless, it remains an open problem to prove whether such manifolds can generally be discovered by LLE, and if so, under what sampling conditions. A survey of known theoretical results for algorithms in nonlinear dimensionality reduction (Tenenbaum et al., 2000; de Silva and Tenenbaum, 2002; Bengio et al., 2003; Brand and Huang, 2003; Donoho and Grimes, 2003) is given in Section 7.

Figure 5 shows another two dimensional manifold, but one living in a much higher dimensional space. Here, we generated examples—shown in the middle panel of the figure—by translating the image of a single face across a larger background of random noise. The noise was independent from one example to the next. Thus, the only consistent structure in the resulting images describes a two dimensional manifold parameterized by the face's center of mass. The input to LLE consisted of $N=961$ grayscale images, with each image containing a 28×20 face superimposed on a 59×51 background of noise. Note that while easy to visualize, the manifold of translated faces is highly nonlinear in the high dimensional vector space ($D=3009$) of pixel coordinates. The bottom portion of Figure 5 shows the first two components discovered by LLE, with $K=4$ neighbors per data point. By contrast, the top portion shows the first two components discovered by PCA. It is clear that the manifold structure in this example is much better modeled by LLE. (The minor edge effects are due to selecting a constant number of neighbors per data point. Thus, the neighbors of boundary points lie further away than the neighbors of interior points.)

We also applied LLE to a data set containing many different images of a single person's face. This data set (consisting of frames from a digital movie) contained $N=1965$ grayscale images at 20×28 resolution ($D=560$). Figure 6 shows the first two components of these images discovered by LLE with $K=12$ nearest neighbors. These components appear correlated with highly nonlinear features of the image, related to pose and expression.

Finally, we applied LLE to images of lips used in the animation of talking heads (Cosatto and Graf, 1998). This data set contained $N=15960$ color (RGB) images of lips at 144×152 resolution ($D=65664$). Figure 7 shows the first two components of these images discovered by LLE with $K=24$ nearest neighbors. These components appear to capture highly nonlinear degrees of freedom associated with opening the mouth, pursing the lips, and clenching the teeth. Figure 8 shows how one particular neighborhood of lip images is embedded in this two dimensional space. Dimensionality reduction of these images is useful for faster and more efficient animation. In particular, the low dimensional outputs of LLE can be used to index the original collection of high dimensional images. Fast and accurate indexing is an essential component of example-based video synthesis from a large library of stored frames.

The data set of lip images is the largest data set to which we have applied LLE. LLE scales relatively well to large data sets because it generates *sparse* intermediate results and eigenproblems. Figure 9 shows the sparsity pattern of a large sub-block of the weight matrix W_{ij} for the data set of lip images. Computing the twenty bottom eigenvectors ($d=20$) for this embedding took only about 2.5 hours on a high end workstation, using specialized routines for finding eigenvectors of sparse, symmetric matrices (Fokkema et al., 1998).

4. Implementation

The algorithm, as described in Figure 2, consists of three steps: nearest neighbor search (to identify the nonzero elements of the weight matrix), constrained least squares fits (to compute the values of these weights), and singular value decomposition (to perform the embedding). We now discuss each of these steps in more detail.

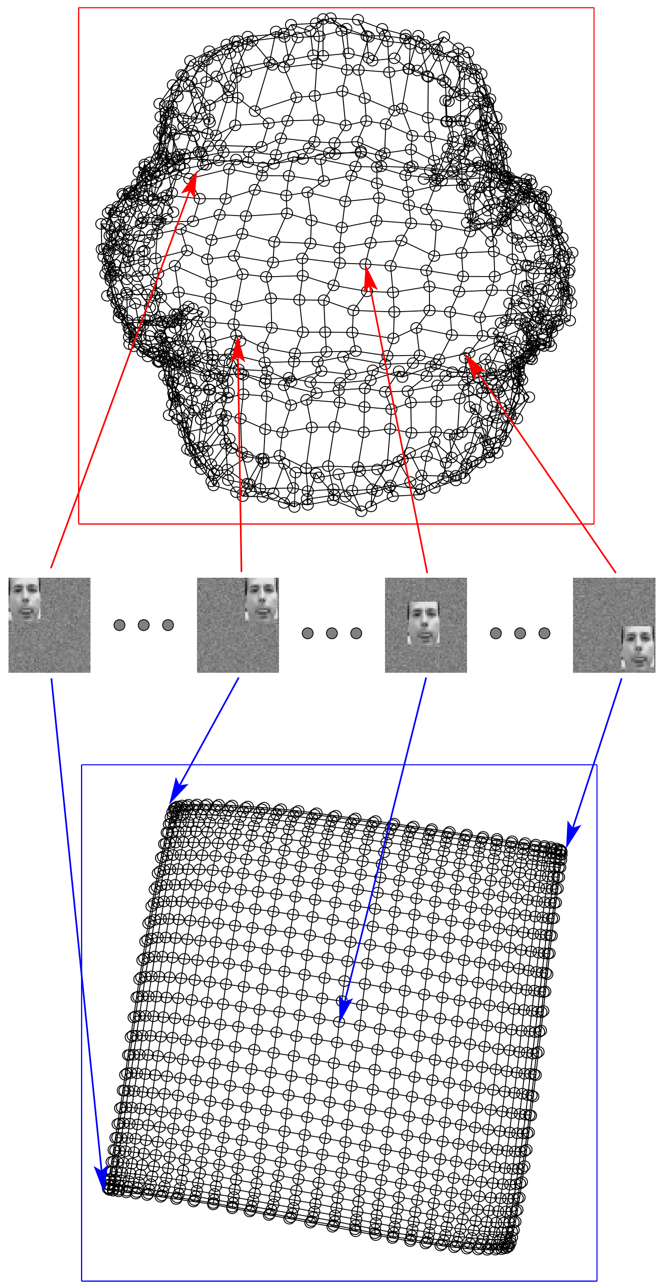

Figure 5: Successful recovery of a manifold of known structure. Shown are the results of PCA (top) and LLE (bottom), applied to $N = 961$ grayscale images of a single face translated across a two dimensional background of noise. Such images lie on an intrinsically two dimensional manifold, but have an extrinsic dimensionality equal to the number of pixels in each image ($D = 3009$). Note how LLE (using $K = 4$ nearest neighbors) maps the images with corner faces to the corners of its two dimensional embedding ($d = 2$), while PCA fails to preserve the neighborhood structure of nearby images.

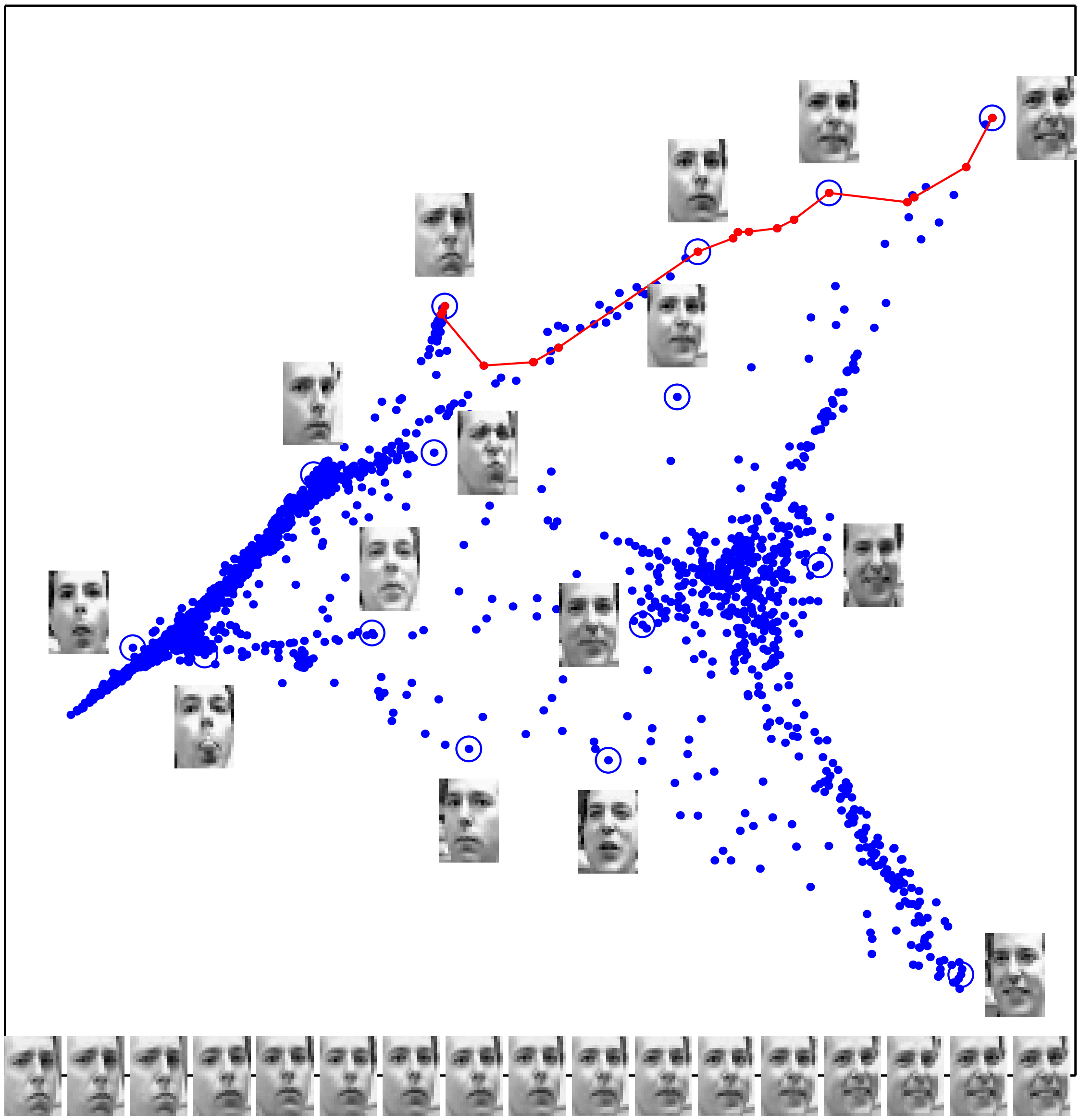

Figure 6: Images of faces mapped into the embedding space described by the first two coordinates of LLE, using $K = 12$ nearest neighbors. Representative faces are shown next to circled points at different points of the space. The bottom images correspond to points along the top-right path, illustrating one particular mode of variability in pose and expression. The data set had a total of $N = 1965$ grayscale images at 20×28 resolution ($D = 560$).

4.1 Step 1: Neighborhood Search

The first step of LLE is to identify the neighborhood of each data point. In the simplest formulation of the algorithm, one identifies a fixed number of nearest neighbors, K, per data point, as measured

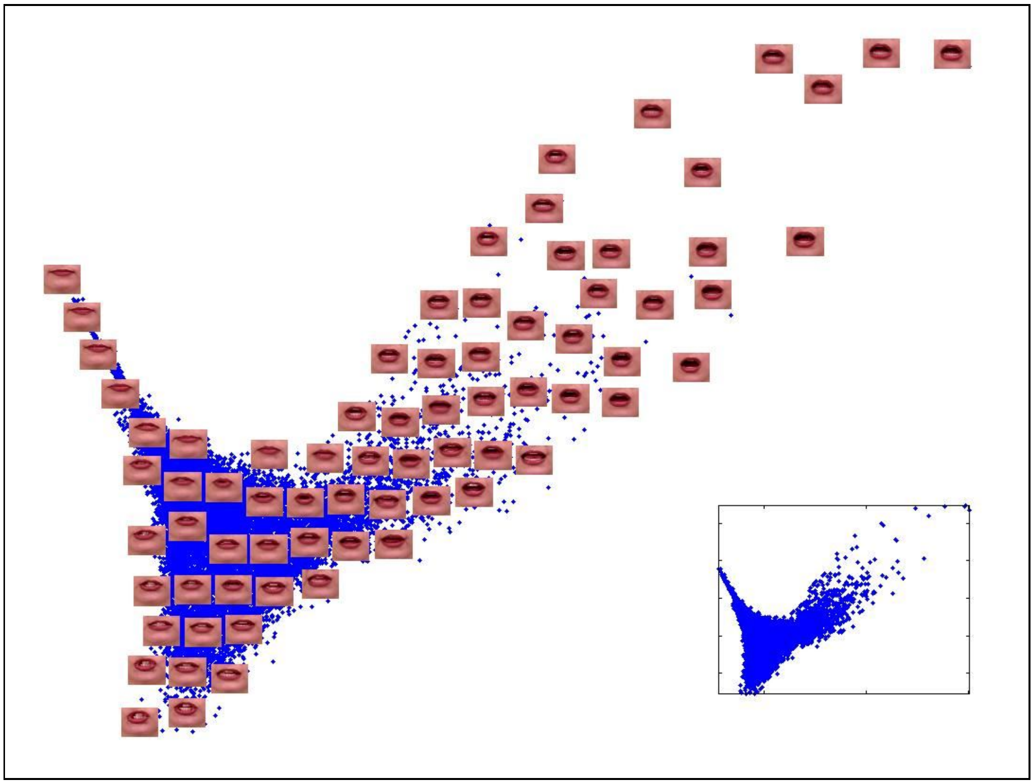

Figure 7: High resolution ($D = 65664$) images of lips, mapped into the embedding space discovered by the first two coordinates of LLE, using $K = 24$ nearest neighbors. Representative lips are shown at different points in the space. The inset shows the first two LLE coordinates for the entire data set ($N = 15960$) without any corresponding images.

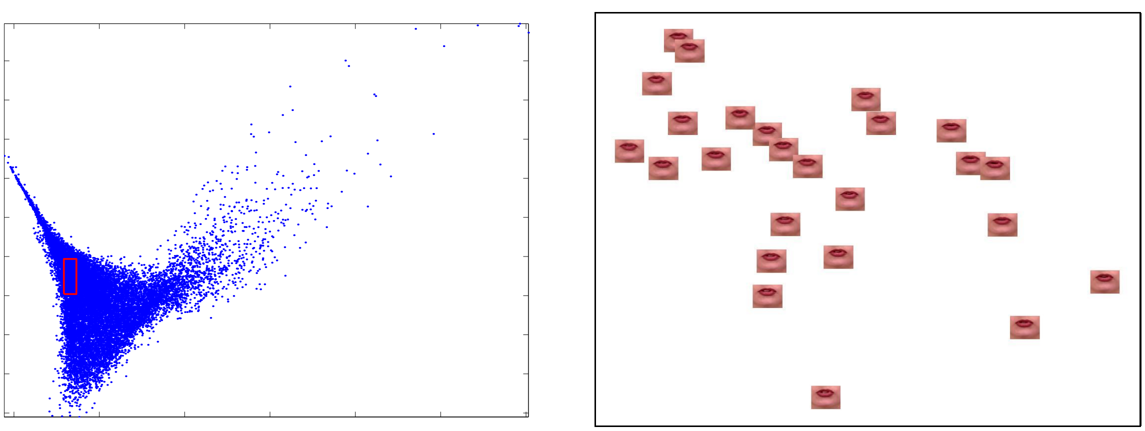

Figure 8: A typical neighborhood of $K = 24$ lip images mapped into the embedding space described by the first two coordinates of LLE. The rectangle in the left plot locates the neighborhood shown on the right in the overall space of lip images.

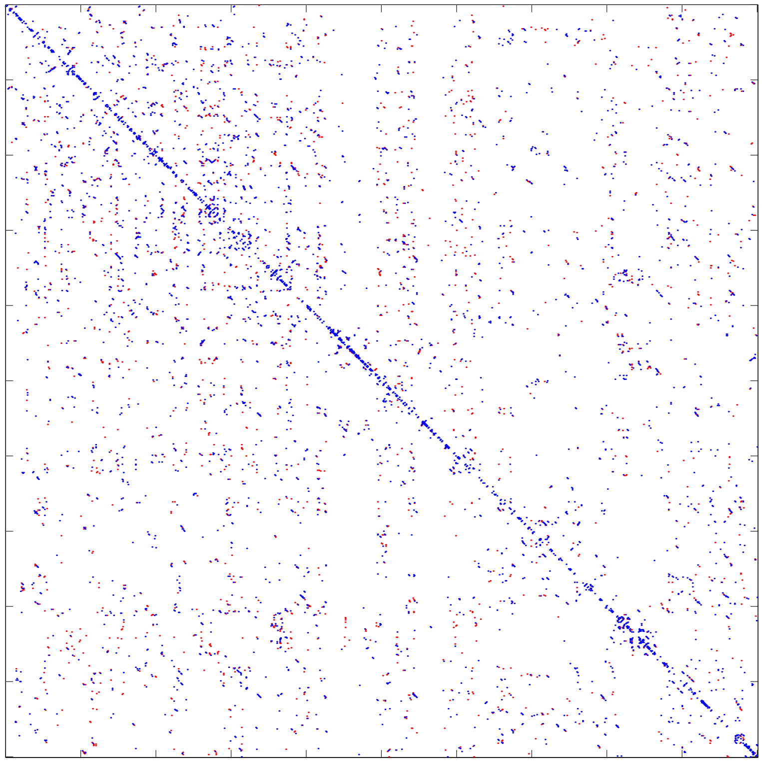

Figure 9: Sparsity pattern of the weight matrix W_{ij} for the data set of lip images. (Only a 1000×1000 sub-block of the matrix is shown to reduce blurring of zero and nonzero elements during printing.) Positive weights are indicated in blue; negative weights, in red. Roughly 99.85% of the elements are zero, although the ink density of the image above does not reflect this accurately.

by Euclidean distance. Other criteria, however, can also be used to choose neighbors. For example, one can identify neighbors by choosing all points within a ball of fixed radius. One can also use locally derived distance metrics (based on a priori knowledge, estimated curvature, pairwise distances, or nonparametric techniques such as box counting) that deviate significantly from a globally Euclidean norm. The number of neighbors does not have to be the same for each data point. In fact, neighborhood selection can be quite sophisticated. For example, we can take all points within a certain radius up to some maximum number, or we can take up to a certain number of neighbors but none outside a maximum radius. In general, specifying the neighborhoods in LLE presents the practitioner with an opportunity to incorporate a priori knowledge about the problem domain.

The results of LLE are typically stable over a range of neighborhood sizes. The size of that range depends on various features of the data, such as the sampling density and the manifold geometry. Several criteria should be kept in mind when choosing the number of neighbors per data point. First, the algorithm can only be expected to recover embeddings whose dimensionality, d, is strictly less[2]

2. The K neighbors span a space of dimensionality at most $K-1$.

than the number of neighbors, K, and we have observed that some margin between d and K is generally necessary to obtain a topology-preserving embedding. (The exact relation between K and the faithfulness of the resulting embedding remains an important open question.) Second, the algorithm is based on the assumption that a data point and its nearest neighbors can be modeled as locally linear; for curved data sets, choosing K too large will in general violate this assumption. Finally in the unusual case where $K > D$ (indicating that the original data is itself low dimensional), each data point can be reconstructed perfectly from its neighbors, and the local reconstruction weights are no longer uniquely defined. In this case, some further regularization must be added to break the degeneracy.[3] Figure 10 shows a range of embeddings discovered by the algorithm, all on the same data set but using different numbers of nearest neighbors, K. The results are stable over a wide range of values but do break down as K becomes too small or large.

The nearest neighbor step in LLE is simple to implement, though it can be time consuming for large data sets ($N \geq 10^4$) if performed without any optimizations. Computing nearest neighbors scales in the worst case as $O(DN^2)$, or linearly in the input dimensionality, D, and quadratically in the number of data points, N. For many distributions of data, however—and especially for those concentrated on a thin submanifold of the input space—constructions such as K-D trees or ball trees can be used to compute the neighbors in $O(N \log N)$ time (Friedman et al., 1977; Gray and Moore, 2001; Moore et al., 2000; Omohundro, 1989, 1991). Recent work by Karger and Ruhl (2002) specifically addresses the problem of computing nearest neighbors for data on low dimensional manifolds. Efficient but approximate methods are also possible, some of which come with various guarantees as to their accuracy (Indyk, 2000).

An implementation of LLE also needs to check that the graph formed by linking each data point to its neighbors is connected. Efficient algorithms (Tarjan, 1972, 1983) exist for this purpose. If the graph is disconnected (or weakly connected), then LLE should be applied separately to the data in each of the graph's (strongly) connected components; or else the neighborhood selection rule should be refined to give a more strongly connected graph. In the disconnected or weakly connected case, data from different connected components should be interpreted as lying on distinct data manifolds. In theory, such situations could be detected after the fact by zeros in the eigenvalue spectrum (Perona and Polito, 2002) of LLE.[4] In practice, though, it seems much more straightforward to first compute connected components and then apply LLE separately to each component. This not only reduces the computational complexity of the algorithm, but also avoids any possible confounding of results from different components.

4.2 Step 2: Constrained Least Squares Fits

The second step of LLE is to reconstruct each data point from its nearest neighbors. The optimal reconstruction weights can be computed in closed form. Consider a particular data point $\vec{x}$ with K nearest neighbors $\vec{\eta}_j$ and reconstruction weights w_j that sum to one. We can write the reconstruction error as:

$$\varepsilon = \left|\vec{x} - \sum_j w_j \vec{\eta}_j\right|^2 = \left|\sum_j w_j (\vec{x} - \vec{\eta}_j)\right|^2 = \sum_{jk} w_j w_k G_{jk}, \tag{3}$$

3. A simple regularizer is to penalize the sum of the squares of the weights $\sum_j W_{ij}^2$, which favors weights that are uniformly distributed in magnitude. This is discussed further in Section 4.2.
4. The corresponding eigenvectors have constant values within each connected component, but different values in different components. This yields a zero embedding cost in Equation (2).

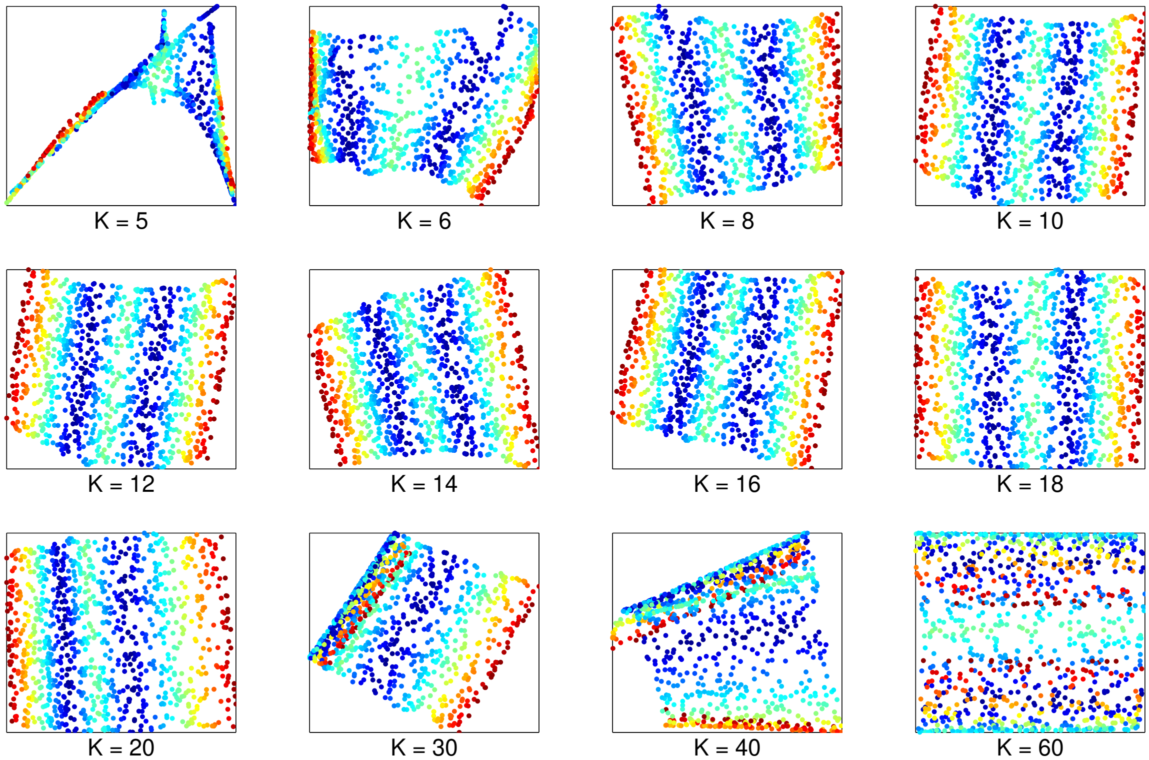

Figure 10: Effect of neighborhood size on LLE. Embeddings of the two dimensional S-manifold in the top panels of Figure 1, computed for different choices of the number of nearest neighbors, K. A reliable embedding from $D=3$ to $d=2$ dimensions is obtained over a wide range of values. If K is too small (top left) or too large (bottom right), however, LLE does not succeed in unraveling the manifold and recovering the two underlying degrees of freedom.

where in the first identity, we have exploited the fact that the weights sum to one, and in the second identity, we have introduced the "local" Gram matrix,

$$G_{jk} = (\vec{x}-\vec{\eta}_j)\cdot(\vec{x}-\vec{\eta}_k). \tag{4}$$

By construction, this Gram matrix is symmetric and semipositive definite. The reconstruction error can be minimized analytically using a Lagrange multiplier to enforce the constraint that $\sum_j w_j = 1$. In terms of the inverse Gram matrix, the optimal weights are given by:

$$w_j = \frac{\sum_k G^{-1}_{jk}}{\sum_{lm} G^{-1}_{lm}}. \tag{5}$$

The solution, as written in Equation (5), appears to require an explicit inversion of the Gram matrix. In practice, a more efficient and numerically stable way to minimize the error (which yields the

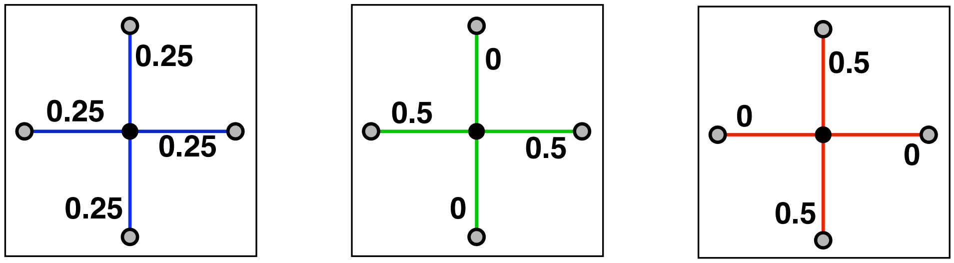

Figure 11: Degeneracy of reconstruction weights. If there are more neighbors (K) than input dimensions (D), then the weights which minimize the reconstruction error in Equation (1) do not have a unique solution. Consider, for example, a two dimensional data point whose four neighbors lie at the corners of a diamond. In this case, many different settings of the weights lead to zero reconstruction error. Three possible settings are shown above. Adding a regularizer that penalizes the squared magnitude of the weights favors the left solution over all others. Note that while this example has particular symmetries (chosen for ease of visualization), the degeneracy arises whenever $K > D$, even for points in general position.

same result as above) is simply to solve the linear system of equations, $\sum_k G_{jk} w_k = 1$, and then to rescale the weights so that they sum to one.

In unusual cases, it can arise that the Gram matrix in Equation (4) is singular or nearly singular—for example, for example, when there are more neighbors than input dimensions ($K > D$), or when the data points are not in general position. In this case, the least squares problem for finding the weights does not have a unique solution (see Figure 11), and the Gram matrix must be conditioned (before solving the linear system) by adding a small multiple of the identity matrix,

$$G_{jk} \leftarrow G_{jk} + \delta_{jk} \left(\frac{\Delta^2}{K} \right) \mathrm{Tr}(G),$$

where δ_{jk} is 1 if $j = k$ and 0 otherwise, $\mathrm{Tr}(G)$ denotes the trace of G, and $\Delta^2 \ll 1$. This amounts to adding a regularization term to the reconstruction cost that measures the summed squared magnitude of the weights. (One can also consider the effect of this term in the limit $\Delta \to 0$.)

The regularization term acts to penalize large weights that exploit correlations beyond some level of precision in the data sampling process. It may also introduce some robustness to noise and outliers. This form of regularization (with $\Delta = 0.1$) was used, for example, to compute all the embeddings in Figure 1. For these synthetic manifolds, the regularization is essential because there are more neighbors ($K = 8$) than input dimensions ($D = 3$). (For most real data sets requiring dimensionality reduction, however, D is much larger than K.)

Computing the reconstruction weights W_{ij} is typically the least expensive step of the LLE algorithm. The computation scales as $O(DNK^3)$; this is the number of operations required to solve a $K \times K$ set of linear equations for each data point. It is linear in both the number of data points and the number of input dimensions. The weight matrix can be stored as a sparse matrix with NK nonzero elements. The inputs $\vec{X}_i$ do not all need to be in memory at once for this step because the algorithm fills in the weights based on a purely local computation.

4.3 Step 3: Eigenvalue Problem

The final step of LLE is to compute a low dimensional embedding based on the reconstruction weights W_{ij} of the high dimensional inputs $\vec{X}_i$. Note that only information captured by the weights W_{ij} is used to construct an embedding; the actual inputs $\vec{X}_i$ do not appear anywhere in the final step of the algorithm (and hence do not need to remain in memory once the weights are computed). The low dimensional outputs $\vec{Y}_i$ are found by minimizing the cost function, Equation (2), for fixed weights W_{ij}. This cost function is minimized when the outputs $\vec{Y}_i$ are reconstructed (or nearly reconstructed) by the *same* weighted linear combinations of neighbors as computed for the inputs. Note that the assignment of neighbors is always based on the locations of the inputs $\vec{X}_i$; the algorithm does not dynamically recompute neighbors based on the locations of the outputs $\vec{Y}_i$.

To optimize the embedding cost function in Equation (2), we rewrite it as the quadratic form:

$$\Phi(Y) = \sum_{ij} M_{ij}(\vec{Y}_i \cdot \vec{Y}_j), \tag{6}$$

involving inner products of the outputs $\vec{Y}_i$. The square $N \times N$ matrix M that appears in Equation (6) is given by:

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}. \tag{7}$$

The matrix M is sparse, symmetric, and semipositive definite.

The optimization of Equation (6) is performed subject to constraints that make the problem well posed. Note that we can translate the outputs $\vec{Y}_i$ by a constant displacement without affecting the cost, $\Phi(Y)$, in Equation (2). We remove this translational degree of freedom by requiring the outputs to be centered on the origin:

$$\sum_i \vec{Y}_i = \vec{0}. \tag{8}$$

We can also rotate the outputs $\vec{Y}_i$ without affecting the cost, $\Phi(Y)$, in Equation (2). To remove this rotational degree of freedom—and to fix the scale—we constrain the $\vec{Y}_i$ to have unit covariance, with outer products that satisfy

$$\frac{1}{N} \sum_i \vec{Y}_i \vec{Y}_i^\top = I, \tag{9}$$

where I is the $d \times d$ identity matrix. This constraint that the output covariance be equal to the identity matrix embodies three assumptions: first, that the different coordinates in the embedding space should be uncorrelated to second-order; second, that reconstruction errors for these coordinates should be measured on the same scale; and third, that this scale should be of order unity. (Note that these assumptions are rather mild. Since we are free to rotate and homogeneously rescale the outputs, we can always make the covariance of $\vec{Y}$ to be diagonal and of order unity. Further restricting the covariance to be the identity matrix only introduces the additional assumption that all the embedding coordinates should be of the same scale.)

Under these restrictions, the optimal embedding—up to a trivial global rotation of the embedding space—is found by minimizing Equation (2) subject to the constraints in Equations (8–9). This can be done in many ways, but the most straightforward is to find the bottom $d+1$ eigenvectors of the cost matrix, M. (The "bottom" or "top" eigenvectors are those corresponding to the smallest or largest eigenvalues.) This equivalence between the optimization of a normalized quadratic form and the computation of largest or smallest eigenvectors is a version of the Rayleitz-Ritz theorem (Horn and Johnson, 1990). The optimization is performed by introducing Lagrange multipliers to enforce the constraints in Equations (8–9). Setting the gradients with respect to the vectors $\vec{Y}_i$ to zero leads to a symmetric eigenvalue problem, exactly as in derivations of principal component analysis (Bishop, 1996). The eigenvector computation in LLE also has the same form as the eigenvector computation for image segmentation by normalized cuts (Shi and Malik, 2000). The bottom eigenvector of the matrix, M, which we discard, is the unit vector with all equal components; it represents the free translation mode of eigenvalue zero. Discarding this eigenvector enforces the constraint in Equation (8) that the outputs have zero mean, since the components of other eigenvectors must sum to zero, by virtue of orthogonality with the bottom one. The remaining d eigenvectors constitute the d embedding coordinates found by LLE.

Note that the bottom $d+1$ eigenvectors of the sparse, symmetric matrix M can be found without performing a full matrix diagonalization (Bai et al., 2000). Operations involving M can also exploit its representation as the product of two sparse matrices,

$$M = (I - W)^\top (I - W),$$

giving substantial computational savings for large values of N. In particular, left multiplication by M (the subroutine required by most sparse eigensolvers) can be performed as:

$$M\vec{v} = (\vec{v} - W\vec{v}) - W^\top(\vec{v} - W\vec{v}),$$

requiring just one multiplication by W and one multiplication by $W^\top$. Thus, the matrix M never needs to be explicitly created or stored; it is sufficient just to store and multiply by the (even sparser) matrix W. An efficient implementation of the multiplication $\vec{v} \leftarrow M\vec{v}$ can be achieved using the update $\vec{v} \leftarrow \vec{v} - W\vec{v}$ followed by the update $\vec{v} \leftarrow \vec{v} - W^\top\vec{v}$.

The final step of LLE is typically the most computationally expensive. Without special optimizations, computing the bottom eigenvectors scales as $O(dN^2)$, linearly in the number of embedding dimensions, d, and quadratically in the number of data points, N. Specialized methods for sparse, symmetric eigenproblems (Bai et al., 2000; Fokkema et al., 1998), however, can be used to reduce the complexity to subquadratic in N. For very large problems, one can consider alternative methods for optimizing the embedding cost function, such as direct descent by conjugate gradient methods (Press et al., 1993), iterative partial minimizations, Lanczos iterations, or stochastic gradient descent (LeCun et al., 1998).

Note that the d^{th} coordinate output by LLE always corresponds to the $(d+1)^{st}$ smallest eigenvector of the matrix M, regardless of the total number of outputs computed or the order in which they are calculated. Thus, for efficiency or convenience, we can compute the bottom eigenvectors of Equation (7) one at a time, yielding a "nested" set of embeddings in successively higher dimensions.

5. Extensions

In this section, we describe several useful extensions to the basic LLE algorithm, including the handling of input in the form of pairwise distances, the use of convex reconstructions, and the estimation of a manifold's underlying dimensionality.

5.1 LLE from Pairwise Distances

The LLE algorithm, as described in Figure 2, takes as input the N high dimensional vectors, $\vec{X}_i$. In many settings, however, the user may not have access to data of this form, but only to measurements of dissimilarity or distance between data points. A simple variation of LLE can be applied to input of this form. In this way, matrices of pairwise distances can be analyzed by LLE just as easily as by MDS (Cox and Cox, 1994) or other distance-based approaches to dimensionality reduction (Tenenbaum et al., 2000).

To derive the reconstruction weights for each data point, we need to compute the Gram matrix G_{jk} between its nearest neighbors, as defined by Equation (4). This matrix can be found by inferring dot products from pairwise distances in exactly the same manner as MDS. In particular, consider a point $\vec{x}$ and its K neighbors η_i, and let S_{ij} denote the symmetric square matrix, of size $(K+1)\times(K+1)$, that records pairwise *squared* distances between these points. (In a slight abuse of notation, here we will allow the indices to range from $i, j = 0, 1, \ldots, K$, where positive values refer to the K neighbors of $\vec{x}$ and the zero index refers to the point $\vec{x}$ itself.) For the purpose of computing the Gram matrix G in Equation (4), we can assume without loss of generality that these $K+1$ points are centered on the origin. In this case, their dot products ρ_{ij} are given exactly in terms of their pairwise squared distances S_{ij} by:

$$\rho_{ij} = \frac{1}{2}\left[\left(\frac{1}{K+1}\right)\sum_{k=0}^{K}(S_{ik}+S_{kj}) - \left(\frac{1}{K+1}\right)^2 \sum_{k,\ell=0}^{K} S_{k\ell} - S_{ij}\right].$$

In terms of our earlier notation, the elements in this matrix store the dot products $\rho_{00} = |\vec{x}|^2$, $\rho_{0j} = \vec{x}\cdot\vec{\eta}_j$ (for $j>0$), and $\rho_{ij} = \vec{\eta}_i\cdot\vec{\eta}_j$ (for $i, j > 0$). The elements of the "local" Gram matrix G_{ij} are given in terms of these dot products ρ_{ij} by:

$$G_{ij} \;=\; (\vec{x}-\vec{\eta}_i)\cdot(\vec{x}-\vec{\eta}_j) \;=\; \rho_{00}-\rho_{i0}-\rho_{0j}+\rho_{ij}.$$

Note that G_{ij} is a $K\times K$ square matrix, whereas ρ_{ij} is one dimension larger. In terms of the Gram matrix G_{ij}, the reconstruction weights for each data point are given by Equation (5). The rest of the algorithm proceeds as usual.

Note that this variant of LLE does not in fact require the complete $N\times N$ matrix of pairwise distances. Instead, for each data point, the user needs only to specify the nearest neighbors, the distances to neighbors, and the pairwise distances between neighbors. This information can be conveniently stored in a sparse matrix. The first step of the algorithm remains to identify nearest neighbors; these can be identified, for example, by the K smallest non-missing elements of each row in the given distance matrix.

It is natural to wonder if this variant of LLE could succeed with the user specifying even fewer elements in the matrix of pairwise distances. Figure 12 shows that just preserving the pairwise distances between nearest neighbors is not in general sufficient to recover an underlying manifold. Consider the three dimensional data set whose points have integer coordinates satisfying $x+y+z=0$; that is, they lie at the sites of a planar square lattice. Suppose that points with even x-coordinates are colored black and those with odd x-coordinates are colored white. The degenerate "two point" embedding that maps all black points to the origin and all white points one unit away *exactly* preserves the distance between each point and its four nearest neighbors. Nevertheless, this embedding completely fails to preserve the structure of the underlying manifold.

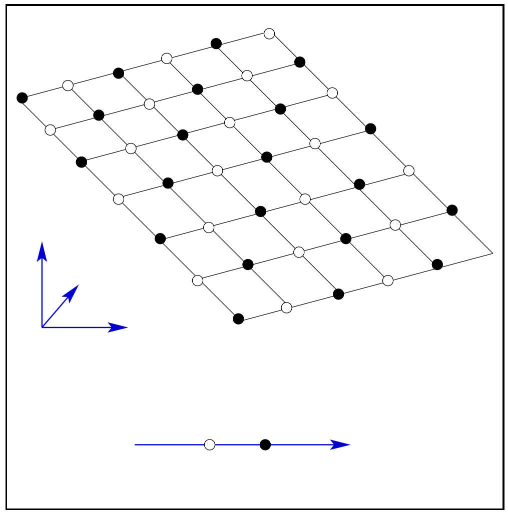

Figure 12: Preserving just the distances to nearest neighbors is not sufficient to recover the underlying manifold. Shown above is a trivial "two-point" embedding that exactly preserves the distances between each point and its four nearest neighbors without revealing the two dimensional structure of the data.

5.2 Convex Reconstructions

The rows of the weight matrix W_{ij} computed by the second step of LLE are constrained to sum to one but may be either positive or negative. In simple geometric terms, this constraint forces the reconstruction of each data point to lie in the subspace spanned by its nearest neighbors; the optimal weights compute the projection of the data point into this subspace. One can additionally constrain these weights to be nonnegative, thus forcing the reconstruction of each data point to lie within the convex hull of its neighbors. It is more expensive to compute the least squares solution with nonnegativity constraints, but the additional cost is usually negligible compared to the other two steps of LLE.[5] In conjunction with the sum-to-one constraint, the constraint of nonnegativity limits the weights strictly to the range [0,1]. Such a constraint has both advantages and disadvantages. On one hand, it tends to increase the robustness of linear fits to outliers. On the other hand, it can degrade the reconstruction of data points that lie on the boundary of a manifold and outside the convex hull of their neighbors. For such points, negative weights may be helpful.

A general prescription for using convex versus linear reconstructions does not exist to cover all applications of LLE. In certain applications, it may be straightforward to certify that neighborhoods are free of outliers, thus minimizing the dangers of unbounded weights; in others, it may be simpler to choose K nearest neighbors everywhere and and require convex reconstructions. A useful heuristic is to inspect a histogram of the reconstruction weights obtained without nonnegativity constraints. If certain data points have very large (positive or negative) reconstruction weights, it may be wise to re-assign their neighbors or to constrain their linear reconstructions to be convex. In our

5. In particular, one must solve a problem in quadratic programming: minimize $\sum_{jk} w_j w_k G_{jk}$ from Equation (3) subject to $\sum_j w_j = 1$ and $w_j \geq 0$. The required optimization is convex, with solutions that often lie on the edge of the constraint region (Judge and Takayama, 1966).

applications of LLE to images and other data sets, these warning signals did not arise. Thus, in our general experience, we have not found the extra constraint of convexity to be especially necessary for well-sampled data.

5.3 Estimating the Intrinsic Dimensionality, d

Given data sampled from an underlying manifold, it is naturally of interest to estimate the manifold's intrinsic dimensionality, d. Recall how PCA solves this problem for linear subspaces: the dimensionality is estimated by the number of eigenvalues of the sample covariance matrix comparable in magnitude to the largest eigenvalue. An analogous strategy for LLE (Perona and Polito, 2002) immediately suggests itself—that is, to estimate d by the number of eigenvalues comparable in magnitude to the smallest nonzero eigenvalue of the cost matrix, M, from Equation (7). In practice, however, we have found this procedure to work only for contrived examples, such as data that lies on an essentially linear manifold, or data that has been sampled in an especially uniform way (so that the lowest nonzero eigenvalues are equal or nearly equal due to symmetry). More generally, we have not found it to be reliable. Figure 13 plots the eigenvalue spectrum of the cost matrix in Equation (7) for several data sets of intrinsic dimensionality $d = 2$. The eigenvalues reveal a distinguishing signature at $d = 2$ in some of these plots, but not in others.

We have found it more useful to rely on classical methods (Pettis et al., 1979) for estimating the intrinsic dimensionality d of a data set. One way to estimate this dimensionality is by examining the eigenvalue spectra of local covariance matrices. Performing in essence a local PCA in the neighborhood of each data point, we can then ask whether these analyses yield a consistent estimate of the intrinsic dimensionality. Yet another estimate can be obtained by box-counting. Suppose we consider two points to be neighbors if they lie within a distance, ε. If the data are uniformly sampled over the manifold, then the number of neighbors should scale for small ε as $K_\varepsilon \propto \varepsilon^d$, where d is the intrinsic dimensionality. Recently, robust variations on this basic idea have been developed to estimate the intrinsic dimensionality of finite data sets with noise and several degrees of freedom (Brand, 2003; Kegl, 2003). Both these methods can be used prior to the final step of LLE to set the number of embedding coordinates computed by the algorithm.

5.4 Enforcing the Intrinsic Dimensionality, d

LLE normally computes an ordered set of embedding coordinates without assuming the particular number that will be used. In some applications, however, a manifold's intrinsic dimensionality may be known a priori, or the user may wish to bias the results of LLE toward an embedding of a particular dimensionality (such as $d = 2$, which is easily visualized). In these circumstances, the second step of LLE can be modified in a simple way to suppress spurious or noisy degrees of freedom and force a desired intrinsic dimensionality, d. For each data point, the idea is to project its neighbors into their d-dimensional subspace of maximal variance before performing the least squares reconstruction. The subspace is computed from the d dominant eigenvectors of the Gram matrix G in Equation (4). The effect of this projection is to limit the rank of G before solving for the reconstruction weights. (Note that this is a far better way to limit the rank of the Gram matrix than simply reducing the number of nearest neighbors.) The reconstruction weights are then computed as before, but from the rank-limited Gram matrix (and using the minimum norm solution to the least squares problem).

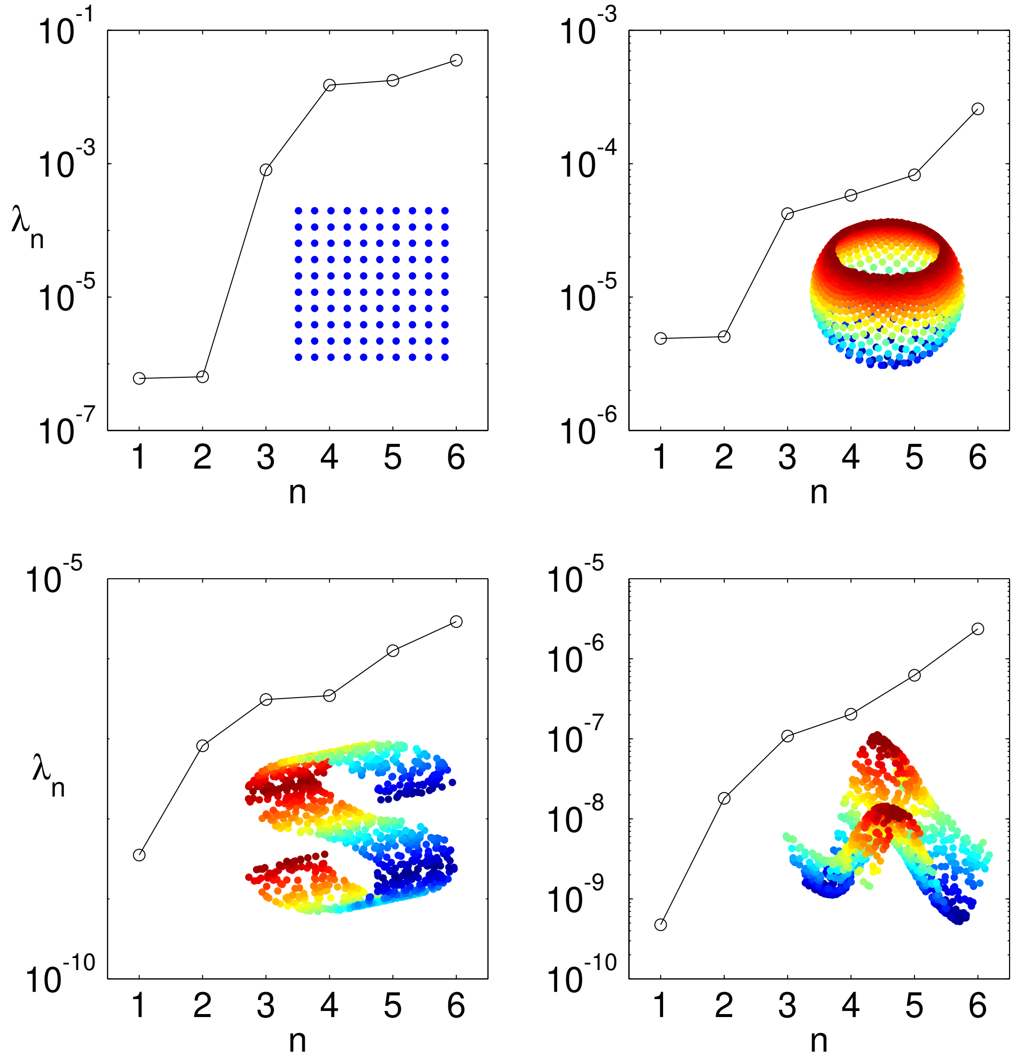

Figure 13: Eigenvalues from the third step of LLE are *not* reliable indicators of intrinsic dimensionality. Plots show the smallest nonzero eigenvalues λ_n of the embedding cost matrix in Equation (7) for several data sets. The gap between the $n=2$ and $n=3$ eigenvalues reveals the true dimensionality ($d=2$) of regularly sampled data in the plane and on the sphere. There is no similar signature, however, for the randomly sampled data on the two bottom manifolds.

As an example of this method, Figure 14 shows the results of applying LLE to scanned images of handwritten digits from the USPS data set (Hull, 1994). The digits (ZERO through NINE) were taken from zip codes on Buffalo postal envelopes. The images were preprocessed by downsampling to 16×16 resolution ($D=256$) and quantizing the grayscale intensity to 256 levels. The inputs $\vec{X}_i$ to LLE were the raw pixel values. For the results in Figure 14, neighbors were projected into an eight

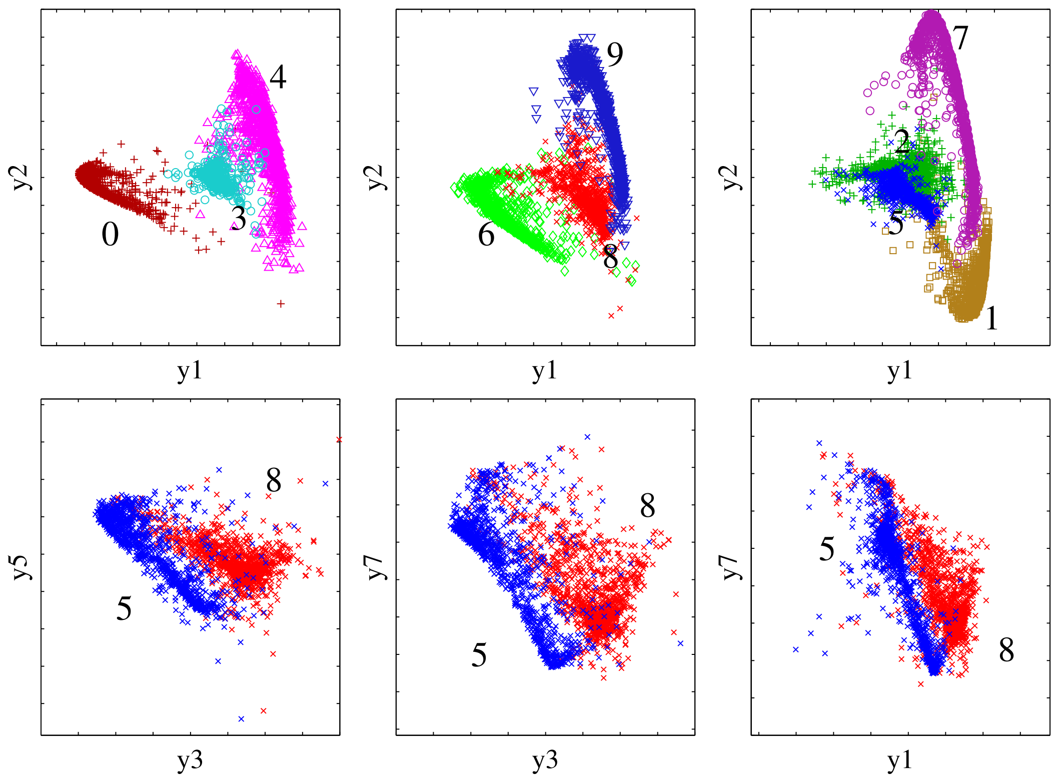

Figure 14: Embeddings of $N = 11000$ handwritten digits from $K = 18$ nearest neighbors per digit. The inputs $\vec{X}_i$ were grayscale images of handwritten numerals (ZERO through NINE) taken from the USPS data set (Hull, 1994) at 16×16 resolution ($D = 256$) A maximum manifold dimensionality of $d = 8$ was enforced by singular value decomposition of the local Gram matrices, as described in Section 5.4. The top panels show the first two coordinates discovered by LLE. Many digit classes (labelled) are well clustered in just these two dimensions. Classes that overlap in the first two dimensions are typically separated in others, as the bottom panels show for FIVES versus EIGHTS in the third, fifth and seventh LLE coordinates.

dimensional subspace ($d = 8$) before performing the least squares reconstructions and computing the weights W_{ij}. Interestingly, the resulting embedding provides a low dimensional clustering of the handwritten digits, even though LLE did not itself make use of the labels distinguishing different digits. Note that digit classes confounded in certain projections are often separated in others. Enforcing the dimensionality of $d = 8$ is not essential for this effect; it is interesting to see, however, that this number does yield an accurate clustering, with the higher-order coordinates discovered by LLE playing as important a role as the first and second.

6. From embeddings to mappings

LLE provides an embedding for the fixed set of training data to which the algorithm is applied. Often, however, we need to *generalize* the results of LLE to new locations in the input space. For example, suppose that we are asked to compute the output $\vec{y}$ corresponding to a new input $\vec{x}$. In principle, we could rerun the entire LLE algorithm with the original data set augmented by the new input. For large data sets of high dimensionality, however, this approach is prohibitively expensive. For the purposes of generalization, it is therefore useful to derive an explicit mapping between the high and low dimensional spaces of LLE that does not require an expensive eigenvector calculation for each query. A natural question is how to construct such a mapping given the results of LLE from a previous run of the algorithm. In this section, we describe two possible solutions—one non-parametric, one parametric—to this problem.

6.1 Non-parametric Model

The non-parametric solution relies on a natural mapping between the low and high dimensional spaces of LLE. In particular, to compute the output $\vec{y}$ for a new input $\vec{x}$, we can do the following: (i) identify the K nearest neighbors of $\vec{x}$ among the training inputs; (ii) compute the linear weights w_j that best reconstruct $\vec{x}$ from its neighbors, subject to the sum-to-one constraint, $\sum_j w_j = 1$; (iii) output $\vec{y} = \sum_j w_j \vec{Y}_j$, where the sum is over the outputs corresponding to the neighbors of $\vec{x}$. A non-parametric mapping from the embedding space to the input space can be derived in the same manner: to compute the input $\vec{x}$ for a new output $\vec{y}$, identify the nearest neighbors of $\vec{y}$ among the training outputs, compute the analogous reconstruction weights, and output $\vec{x} = \sum_j w_j \vec{X}_j$. (Note that in this direction, moreover, there does not exist the option of rerunning LLE on an augmented data set.)

To evaluate the usefulness of this non-parametric mapping, we investigated LLE as a front end for statistical pattern recognition. Dimensionality reduction is often used as a form of feature extraction to simplify or accelerate other algorithms in machine learning. We compared the features produced by LLE to those of PCA for a benchmark problem in handwritten digit recognition. The raw data were images of handwritten digits from the USPS database, as described in Section 5.4. Half the images (5500 examples, 550 per digit) was used for training, and the other half for testing. On test images, LLE features were computed by nearest-neighbor interpolation from the training images (as described above), while PCA features were computed by subtracting the mean of the training images then projecting onto their principal components.

Two simple classifiers were implemented: a K-nearest-neighbor classifier—with K chosen to optimize leave-one-out performance on the training set—and a log-linear classifier obtained from multiclass logistic regression (or softmax regression) (Bishop, 1996). The parameters of the softmax regression were set to maximize the conditional log likelihood of the training set. As features for these classifiers, we used either the first d coordinates discovered by LLE on the entire (unlabeled) training set of 5500 images, or else the projection onto the first d principal components of the training data, as computed by running PCA on the training set (i.e. subtracting the sample mean and finding the leading eigenvectors of the sample covariance matrix). LLE was run using $K = 18$ nearest neighbors per data point to compute the reconstruction weights.

Figure 15 shows the results of these experiments. For small numbers of features, LLE leads to significantly lower error rates than PCA on both the training and test sets (though only the test error rates are shown in the figure). These results suggest that nearest-neighbor interpolation is an

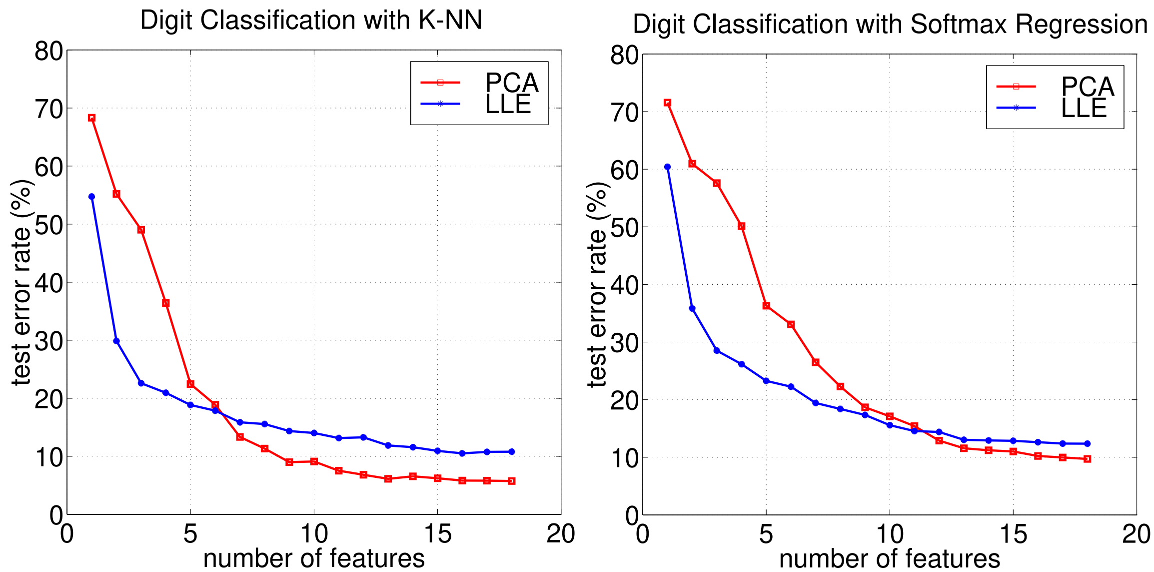

Figure 15: Comparison of LLE and PCA features for classification of handwritten digits from the USPS database (Hull, 1994). Both K-nearest neighbor (K-NN) and log-linear classifiers were evaluated; see the text for more details. The plots show error rates on the test set, for which LLE features were computed by nearest-neighbor interpolation. LLE features initially lead to better performance, but a crossover in error rates occurs as the number of features approaches the neighborhood size ($K = 18$) in LLE. For reference, the test error rate of K-NN applied directly to the high dimensional ($D = 256$) digit images (without any feature extraction) is 7.6%, using 4 nearest neighbors.

effective way to apply LLE to test examples. Note, however, that a crossover in error rates occurs as the number of features approaches the neighborhood size ($K = 18$) used in the preprocessing by LLE. In this regime, it appears that LLE cannot extract further information from its locally linear fits; thus, its performance saturates while that of PCA continues to improve. Taking more neighbors in LLE might extend the number of meaningful features that could be extracted, but in this case, one would lose the benefits of nonlinearity (assuming that the sampling density of the data is held fixed, and that the underlying manifold is only locally linear over a fixed scale).

In summary, the non-parametric mapping derived from nearest-neighbor interpolation is based on the same underlying intuitions as LLE. It provides a simple way to generalize to new data when the assumptions of local linearity are met. An open problem is to establish the asymptotic conditions under which this non-parametric mappings yield the same (or nearly the same) result as rerunning LLE on the augmented data set of original plus new inputs. Though straightforward to implement, this approach to generalization has the disadvantage that it requires access to the entire set of previously analyzed inputs and outputs — potentially a large demand in storage. Also, the non-parametric mapping can change discontinuously as query points move between different neighborhoods of the inputs $\vec{X}_i$ or outputs $\vec{Y}_i$. These concerns motivate the parametric model discussed in the next section.

6.2 Parametric Model

Methods in supervised learning and function approximation can be used to derive more compact mappings that generalize over large portions of the input and embedding space. In particular, one can take the input-output pairs of LLE as training data for an invertible function approximator and learn a parametric mapping between the two spaces. Here, we discuss one such approach that is based on similar intuitions as its non-parametric counterpart.

Given the results of LLE, we consider how to learn a probabilistic model of the joint distribution, $P(\vec{x},\vec{y})$, over the input and embedding spaces. The joint distribution can be used to map inputs to outputs and vice versa by computing the expected values $\mathrm{E}[\vec{y}|\vec{x}]$ and $\mathrm{E}[\vec{x}|\vec{y}]$. To represent the joint distribution, we propose a mixture model (McLachlan and Basford, 1988) that is specifically tailored to data that lies on a low dimensional manifold. The individual components of this mixture model are used to represent the densities of locally linear neighborhoods on the manifold. Mixture models have been widely used for this purpose in the past (Beymer and Poggio, 1996; Bregler and Omohundro, 1995; Hinton et al., 1997; Kambhatla and Leen, 1997; Roweis et al., 2002; Saul and Rahim, 1999; Vlassis et al., 2002), so their treatment here is necessarily brief.

The model that we consider is a mixture of conditional linear models with Gaussian noise distributions. It describes a three-step generative process for high and low dimensional vectors $\vec{x}\in\mathcal{R}^D$ and $\vec{y}\in\mathcal{R}^d$. First, a discrete hidden variable z is sampled from its prior distribution, $P(z)$, to select a particular neighborhood on the manifold. Next, a d-dimensional vector $\vec{y}$ is sampled from a conditional Gaussian distribution, $P(\vec{y}|z)$, with mean vector $\vec{\nu}_z$ and (full) covariance matrix Σ_z. Finally, a D-dimensional vector $\vec{x}$ is sampled from a conditional Gaussian distribution, $P(\vec{x}|\vec{y},z)$ with mean vector $\Lambda_z\vec{y}+\vec{\mu}_z$ and diagonal covariance matrix Ψ_z. Here, Λ_z is a $D\times d$ loading matrix that describes the locally linear mapping from low to high dimensional observations. The model is similar to an unsupervised mixture of factor analyzers (Rubin and Thayer, 1982; Ghahramani and Hinton, 1996), except that the low dimensional variable $\vec{y}$ is observed, not hidden, and the Gaussian distributions $P(\vec{y}|z)$ have nonzero mean vectors and full covariance matrices. The overall distribution is given by:

$$\begin{aligned}
P(\vec{x},\vec{y},z) &= P(\vec{x}|\vec{y},z)P(\vec{y}|z)P(z)\\
P(\vec{x}|\vec{y},z) &= \frac{|\Psi_z|^{-1/2}}{(2\pi)^{D/2}}\exp\left\{-\frac{1}{2}\left[\vec{x}-\Lambda_z\vec{y}-\vec{\mu}_z\right]^T\Psi_z^{-1}\left[\vec{x}-\Lambda_z\vec{y}-\vec{\mu}_z\right]\right\}\\
P(\vec{y}|z) &= \frac{|\Sigma_z|^{-1/2}}{(2\pi)^{d/2}}\exp\left\{-\frac{1}{2}\left[\vec{y}-\vec{\nu}_z\right]^T\Sigma_z^{-1}\left[\vec{y}-\vec{\nu}_z\right]\right\}.
\end{aligned}$$

The parameters in this model which need to be estimated from data are the prior probabilities $P(z)$, the mean vectors $\vec{\nu}_z$ and $\vec{\mu}_z$, the full covariance matrices Σ_z and the diagonal covariance matrices Ψ_z, and the loading matrices Λ_z. The training examples for the model consist of the input-output pairs from a D-dimensional data set $\vec{X}$ and its d-dimensional locally linear embedding $\vec{Y}$.

The parameters in this model can be learned by an Expectation-Maximization (EM) algorithm for maximum likelihood estimation (Dempster et al., 1977). The EM algorithm is an iterative procedure that attempts to maximize the total log-likelihood of observed input-output pairs in the training set. The re-estimation formulae for this model are given in Appendix A.

Figure 16 shows a model with 32 mixture components learned from the results of LLE on the S-shaped manifold in Figure 1. The Gaussian distributions $P(\vec{x}|\vec{y},z)$ are depicted by planes centered on the points $\Lambda_z\vec{\nu}_z+\vec{\mu}_z$, whose normal vectors are perpendicular to the subspaces spanned by the rows

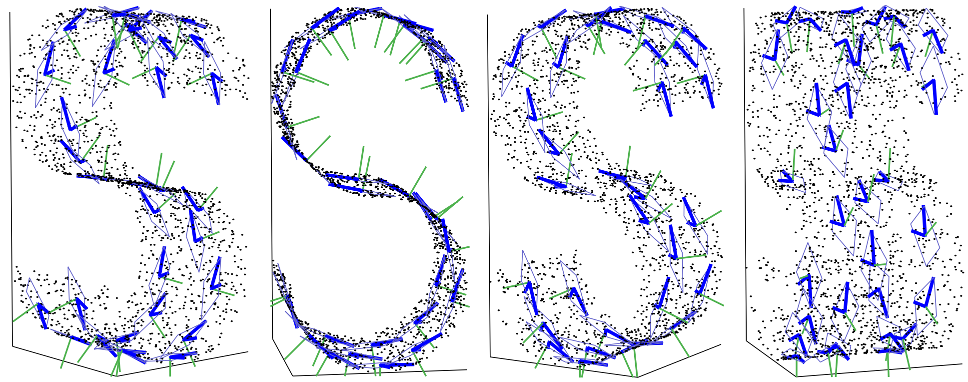

Figure 16: Mixture of linear models learned from data sampled from the surface of a two dimensional manifold. Each mixture component parameterizes the density over a locally linear neighborhood on the manifold. The mixture model with 32 components was trained on the input-output pairs of LLE shown in the top row of Figure 1. Thick lines indicate increasing coordinates on the manifold, while light squares (and their normals) indicate the subspaces modeled by individual mixture components. Parameters of the mixture model were estimated by the EM algorithm described in Section 6.2 and appendix A.

of Λ_z. These subspaces very accurately model the locally linear neighborhoods on the manifold ($d=2$) from which the data was generated. Note that mixture models in high dimensional spaces (even $D=3$) are typically plagued by very poor local minima, and that an unsupervised mixture of factor analyzers—treating $\vec{y}$ as a hidden variable—would be unlikely to discover a solution of this quality. Clamping these latent variables to the outputs of LLE, as is done here, makes learning much easier.

7. Discussion

We conclude by tracing the origins of this work, comparing LLE to other well known algorithms for nonlinear dimensionality reduction, and mentioning some open problems for future research.

7.1 Early Motivation

The motivation for LLE arose from an extended line on work on mixture models. A number of researchers had shown that mixtures of locally linear models could be used to parameterize the distributions of data sets sampled from underlying manifolds (Bregler and Omohundro, 1995; Kambhatla and Leen, 1997; Ghahramani and Hinton, 1996; Hinton et al., 1997; Saul and Rahim, 1999). These density models, however, exhibited a peculiar degeneracy: their objective functions (measuring either least squares reconstruction error or log likelihood) were invariant to arbitrary rotations and reflections of the local coordinate systems in each linear model. In other words, their learning algorithms did not favor a consistent alignment of the local linear models, but instead yielded internal representations that changed unpredictably as one traversed connected paths on the manifold.

LLE was designed to overcome this shortcoming—to discover a single global coordinate system of lower dimensionality. Initially, we imagined that such a coordinate system could be obtained by patching together the local coordinate systems of individual components in a mixture model (Hinton and Revow, 1998). Difficulties with this approach led us to consider a non-parametric setting in which the local coordinate systems were defined by each data point and its nearest neighbors. The main novelty of LLE lies, we believe, in its appeal to particular symmetries. The reconstruction weights in LLE capture the intrinsic geometric properties of local neighborhoods—namely, those properties invariant to translation, rotation, and scaling. The appeal to these symmetries was directly motivated by the degeneracy observed in earlier work on mixture models.

While LLE is a non-parametric method, recent studies (Roweis et al., 2002; Verbeek et al., 2002a) have, in fact, shown that it is possible to learn a probabilistic mixture model whose individual coordinate systems are aligned in a consistent way, thus unifying nonlinear dimensionality reduction and density estimation in a single framework. These approaches rely on LLE to initialize certain parameter estimates and overcome the otherwise difficult problem of local maxima. Related eigenvector methods have also been proposed (Brand, 2003; Teh and Roweis, 2003) that decouple the processes of density estimation and manifold learning into two consecutive steps. First, a local density model, with the degeneracies described above, is fit to the data. Second, the internal coordinate systems of the model are "post-aligned" in a way that does not change the likelihood, but achieves the goal of learning global structure.

7.2 Related and Ongoing Work

At its core, LLE uses linear methods—least squares optimization and matrix diagonalization—to obtain highly nonlinear embeddings. The only element of nonlinearity is introduced by the first step of the algorithm—a nearest neighbor search—which can be viewed as a highly nonlinear thresholding procedure. Because its optimizations are straightforward to implement and do not involve local minima, LLE compares favorably in implementation cost to purely linear methods, such as PCA and classical MDS. Unlike these methods, however, LLE can be used to address problems in nonlinear dimensionality reduction, and so may yield superior results. LLE also generates a sparse eigenvalue problem, as opposed to the dense eigenvalue problems in PCA and MDS. This has many advantages for scaling its computations up to large, high dimensional data sets.

LLE illustrates a general principle, elucidated by earlier studies (Martinetz and Schulten, 1994; Tenenbaum, 1998), that overlapping local neighborhoods—collectively analyzed—can provide information about global geometry. The formulation of LLE in terms of reconstruction weights and eigenvectors arose, somewhat serendipitously, from a completely unrelated line of work in signal processing (Saul and Allen, 2001). LLE is more properly viewed, however, as belonging to a family of recently proposed algorithms that use eigenvector methods to solve highly nonlinear problems in dimensionality reduction, clustering, and image segmentation (Belkin and Niyogi, 2002; Ng et al., 2002; Schölkopf et al., 1998; Shi and Malik, 2000; Tenenbaum et al., 2000; Weiss, 1999). These algorithms—discussed in more detail below—avoid many of the pitfalls that plague other nonlinear approaches, such as autoencoder neural networks (DeMers and Cottrell, 1993; Kramer, 1991), self-organizing maps (Durbin and Wilshaw, 1987; Kohonen, 1988), latent variable models (Bishop et al., 1998), principal curves and surfaces (Hastie and Stuetzle, 1989; Verbeek et al., 2002b), and many variants on multidimensional scaling (Cox and Cox, 1994; Klock and Buhmann, 1999; Littman et al., 1992; Takane and Young, 1977). These latter approaches, especially those based on hill-climbing methods, do not have the same guarantees of global optimality or convergence as eigenvector methods; they also tend to involve many free parameters, such as learning rates, initial conditions, convergence criteria, and architectural specifications—all of which must be tuned by the user or set by cross validation.

The first and third steps of LLE are similar to those of the normalized cut algorithm for image segmentation (Shi and Malik, 2000) and related Laplacian-based methods for clustering (Ng et al., 2002) and dimensionality reduction (Belkin and Niyogi, 2002). At the heart of all these algorithms is a sparse eigenvalue problem derived from a weighted graph representing neighborhood relations. Recent work (Belkin and Niyogi, 2002) has related LLE to the Laplacian-based methods and argued that both approaches can be understood in terms of a unified framework for clustering and dimensionality reduction (see also Brand and Huang, 2003; Bengio et al., 2003). There have been several extensions of the normalized cut algorithm for clustering and image segmentation—to directed graphs (Yu and Shi, 2001) and to probabilistic settings (Meila and Shi, 2000)–that would be interesting to explore for problems in manifold learning. Likewise, Figure 14 gives an indication that LLE might be useful for certain types of clustering: the algorithm's unsupervised dimensionality reduction of $N = 11000$ handwritten digits (from $D = 256$ to $d = 8$) largely preserves the separation between different classes.

A different but equally successful approach to nonlinear dimensionality reduction is the Isomap algorithm (Tenenbaum, 1998; Tenenbaum et al., 2000). Isomap is a nonlinear generalization of MDS in which embeddings are optimized to preserve "geodesic" distances between pairs of data points—that is to say, distances along the manifold from which the data is sampled. These distances are estimated by computing shortest paths through large sublattices of data. Like LLE, the Isomap algorithm has three steps: (i) construct a graph in which each data point is connected to its nearest neighbors; (ii) compute the shortest distance between all pairs of data points among only those paths that connect nearest neighbors; (iii) embed the data via MDS so as to preserve these distances. Though similiar in its aims, Isomap is based on a radically different philosophy than LLE (as well as the other Laplacian-based spectral methods discussed above). In particular, Isomap attempts to preserve the global geometric properties of the manifold, as characterized by the geodesic distances between faraway points, while LLE attempts to preserve the local geometric properties of the manifold, as characterized by the linear coefficients of local reconstructions.

As LLE and Isomap are based on somewhat different intuitions, when they break down, they tend to make different errors. The embeddings of LLE are optimized to preserve the geometry of nearby inputs; though the collective neighborhoods of these inputs are overlapping, the coupling between faraway inputs can be severely attenuated if the data is noisy, sparse, or weakly connected. Thus, the most common failure mode of LLE (often arising if the manifold is undersampled) is to map faraway inputs to nearby outputs[6] in the embedding space. By contrast, the embedding cost for Isomap is dominated by the (geodesic) distances between faraway inputs. Thus, its embeddings are biased to preserve the separation of faraway inputs at the expense of distortions in the local geometry. Depending on the application, one algorithm or the other may be most appropriate. A difficult example for LLE is shown in Figure 17, where the data was generated from the volume of a three-dimensional "barbell". For this example, most settings of the parameters K and Δ in the

6. Such failures can be detected by computing pairwise distances between *outputs* and testing that nearby outputs correspond to nearby inputs. Note that one could modify the embedding cost function in Equation (2) to include repulsive as well as attractive terms—in other words, to push non-neighbors apart as well as to keep neighbors close. This, however, creates a less sparse eigenvalue problem.

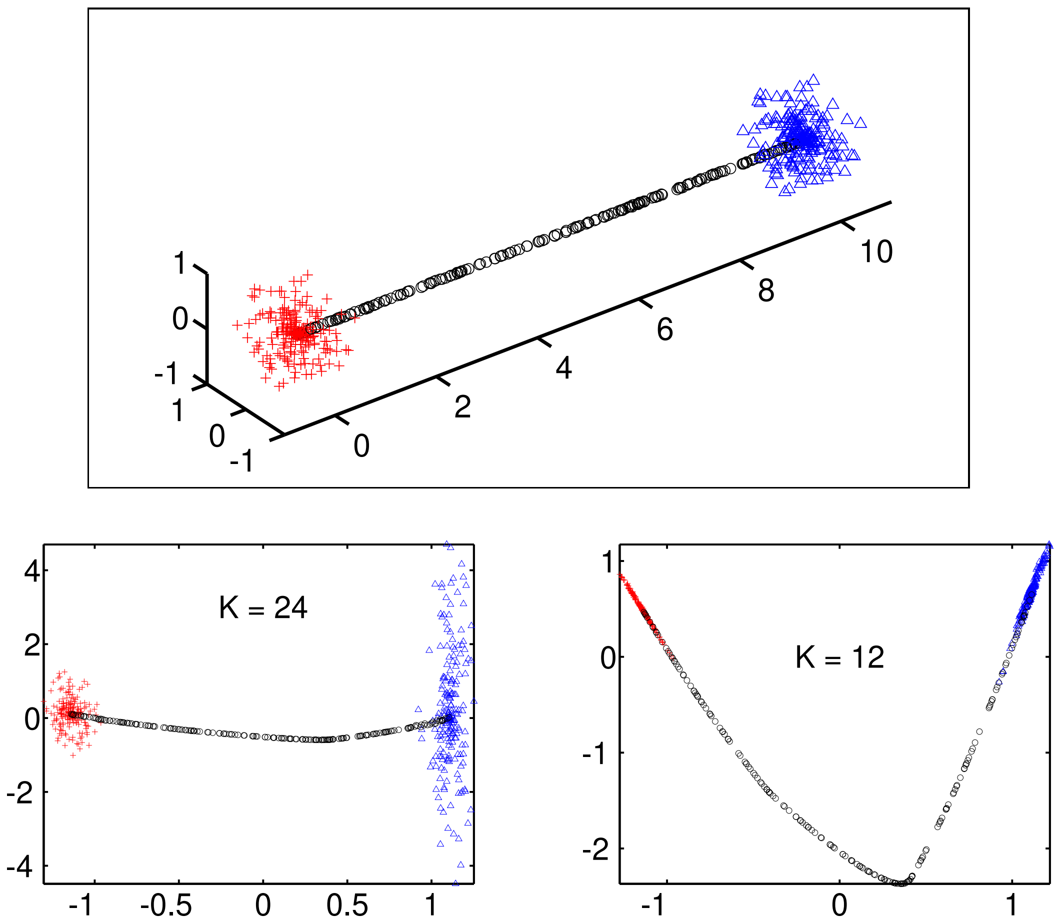

Figure 17: A difficult data set for LLE. Top: data generated from the volume of a three-dimensional barbell. Bottom left: a two dimensional embedding obtained from carefully tuned parameter settings for the number of nearest neighbors, K, and the regularization coefficient, $\Delta = 0.3$. Bottom right: a more typical result from LLE, where the algorithm fails to discover a faithful embedding. The different colors and symbols reveal the embedding for different parts of the barbell.

LLE algorithm do *not* lead to reasonable results. Arguably, in this case, the data is best described as belonging to a collection of manifolds of different dimensionality. Thus, it would appear that a hybrid strategy of some sort (e.g., identifying weakly connected components, varying the number of neighbors K per data point) is required by LLE to give consistently faithful embeddings.

Other important differences between LLE and Isomap are worth mentioning. In terms of computational requirements, LLE does not involve the need to solve large dynamic programming problems, such as computing the geodesic distances between data points. It also creates only very sparse matrices, whose structure can be exploited for savings in time and space. This efficiency gain is especially important when attempting to diagonalize large matrices. By subsampling the data to use only certain “landmark” points, Isomap’s optimization can also be made relatively sparse, although at the expense of approximating its original objective function.

In Section 3, we conjectured that under appropriate conditions, LLE can recover conformal mappings—mappings which locally preserve angles, but not necessarily distances. Such mappings cannot generally be recovered by Isomap, whose embeddings explicitly aim to preserve the distances between inputs. Noting this, de Silva and Tenenbaum (2002) recently proposed a variant of Isomap that is able to recover conformal mappings, under the assumption that the data is distributed uniformly (or with known density) in the low dimensional embedding space. The new algorithm, called c-Isomap, uses the observed density in the high dimensional input space to estimate and correct for the local neighborhood scaling factor of the conformal mapping.

Certain formal guarantees have been established for Isomap (Tenenbaum et al., 2000; Donoho and Grimes, 2002) and c-Isomap (de Silva and Tenenbaum, 2002), and more recently, for a reformulation of LLE that uses Hessian information (Donoho and Grimes, 2003). The strongest results show that in the asymptotic limit of infinite sampling, both Isomap and “Hessian LLE” can recover manifolds that are locally isometric to subsets of a lower dimensional Euclidean space. For Isomap, these subsets must be open and convex, whereas for Hessian LLE, they need only be open and connected. The contrast here is particularly interesting, showing that the principles behind Isomap and LLE give rise to algorithms with provably different properties. Notwithstanding these recent results, our theoretical understanding of algorithms for nonlinear dimensionality reduction is far from complete. Important issues for LLE are its non-asymptotic behavior for finite data sets and its suitability for manifolds that can be conformally (but not isometrically) mapped to subsets of a lower dimensional Euclidean space.

Yet another eigenvector-based algorithm for nonlinear dimensionality reduction is kernel PCA (Schölkopf et al., 1998). This approach builds on the observation that PCA can be formulated entirely in terms of dot products between data points. In kernel PCA, one substitutes the inner product of a Hilbert space for the normally Euclidean dot products of PCA. This amounts to performing PCA on a nonlinear mapping of the original inputs into a different (possibly infinite dimensional) space where their intrinsically low dimensional structure is easier to discover. Williams (2001) has pointed out a connection between kernel PCA and metric MDS. Unlike LLE, kernel PCA does not appeal explicitly to the notion that the data lies on a manifold. However, by viewing the elements of the cost matrix M in Equation (7) as kernel evaluations for pairs of inputs, LLE can be seen as kernel PCA for a particular choice of kernel (Schölkopf and Smola, 2002, see p. 455). In related work, a “kernalized” version of LLE has been proposed (DeCoste, 2001) to visualize the effects of different kernels.

This paper has focused on the application of LLE to high dimensional data sampled (or believed to be sampled) from a low dimensional manifold. LLE can also be applied to data for which the existence of an underlying manifold is not obvious. In particular, while we have focused on real valued signals such as images, relationships between categorial or discrete valued quantities can also be analyzed with LLE. In previous work (Roweis and Saul, 2000), for example, we applied LLE to documents of text, where it can be viewed as a nonlinear alternative to the traditional method of latent semantic analysis (Deerwester et al., 1990).

Whatever the application, though, certain limitations of the algorithm should be kept in mind. Not all manifolds are suitable for LLE, even in the asymptotic limit of infinite data. How should we handle manifolds, such as the sphere and the torus, that do not admit a uniformly continuous mapping to the plane (Pless and Simon, 2001)? Likewise, how should we embed data sets whose intrinsic dimensionality is not the same in all parts of space (as in the barbell example), or whose structure is better described as fractal? Further work is needed in these settings. A closely related issue—how to cope with poorly or nonuniformly sampled data—should also be investigated. While the success of LLE hinges on a sufficiently dense sampling of the underlying manifold, recent work (Ham et al., 2003) suggests that bootstrapping methods and self-consistency constraints can be used to improve the algorithm's performance on smaller data sets.

7.3 Summary

In this paper, we have provided a thorough survey of the LLE algorithm—the details of its implementation, an assortment of possible uses and extensions, and its relation to other eigenvector methods for clustering and nonlinear dimensionality reduction. LLE is, of course, an unsupervised learning algorithm, one that does not require labeled inputs or other types of feedback from the learning environment. An oft-made criticism of unsupervised algorithms is that they attempt to solve a harder problem than is necessary for any particular task (or in some cases, even the wrong problem altogether). In our view, LLE belongs to a new class of unsupervised learning algorithms that removes much of the force behind this argument. These new algorithms do not make strong parametric assumptions, and they are distinguished by simple cost functions, global optimizations, and the potential to exhibit highly nonlinear behavior. We expect these algorithms to be broadly useful in many areas of information processing, and particularly as a tool to simplify and accelerate other forms of machine learning in high dimensional spaces.

Acknowledgements

The authors thank E. Cosatto, H. P. Graf, and Y. LeCun and B. Frey for providing data for these experiments. J. Platt, H. Hoppe, and R. Zemel all suggested the method in Section 5.4 for enforcing the intrinsic dimensionality of the recovered manifold, and T. Jaakkola pointed out certain properties of convex reconstructions, as discussed in Section 5.2. We thank J. Tenenbaum, P. Niyogi, D. Donoho, and C. Grimes for many useful discussions, G. Hinton and M. Revow for sharing a preprint on nonlinear dimensionality reduction with locally linear models, and M. Klass for suggesting the role of uniform continuity as a constraint on the mappings learnable by LLE. We are also grateful to the anonymous reviewers, whose comments helped to improve nearly every section of the paper. S. Roweis acknowledges the support of the National Sciences and Engineering Research Council of Canada and the Institute for Robotics and Intelligent Systems (IRIS).

Appendix A. EM Algorithm for Mixture of Linear Models

In this section, we present the update rules for the EM algorithm in Section 6.2. The algorithm is used to estimate the parameters of that section's generative model for input-output pairs of LLE, here denoted by $\{\vec{x}_n, \vec{y}_n\}_{n=1}^N$. The derivation is a special case of the full derivation of the EM algorithm for mixtures of factor analyzers (Ghahramani and Hinton, 1996) and is not repeated here. The E-step of the EM algorithm uses Bayes' rule to compute the posterior probabilities:

$$P(z|\vec{y}_n, \vec{x}_n) = \frac{P(\vec{x}_n|\vec{y}_n, z)P(\vec{y}_n|z)P(z)}{\sum_{z'} P(\vec{x}_n|\vec{y}_n, z')P(\vec{y}_n|z')P(z')}. \tag{10}$$

The M-step uses these posterior probabilities to re-estimate the parameters of the model. To simplify the updates in the M-step, we introduce the following notation:

$$\begin{aligned} \gamma_{zn} &= P(z|\vec{x}_n,\vec{y}_n), \\ \omega_{zn} &= \frac{\gamma_{zn}}{\sum_{n'}\gamma_{zn'}}. \end{aligned}$$

Here, γ_{zn} and ω_{zn} are the elements of $M\times N$ matrices, where M is the number of mixture components and N is the number of examples. In terms of this notation, the M-step consists of the following updates, to be performed in the order shown:

$$\vec{\nu}_z \leftarrow \sum_n \omega_{zn}\vec{y}_n, \tag{11}$$

$$\vec{\Sigma}_z \leftarrow \sum_n \omega_{zn}\left[\vec{y}_n-\vec{\nu}_z\right]\left[\vec{y}_n-\vec{\nu}_z\right]^T, \tag{12}$$

$$\Lambda_z \leftarrow \sum_n \omega_{zn}\vec{x}_n(\vec{y}_n-\vec{\nu}_z)^T\Sigma_z^{-1}, \tag{13}$$

$$\vec{\mu}_z \leftarrow \sum_n \omega_{zn}\left[\vec{x}_n-\Lambda_z\vec{y}_n\right], \tag{14}$$

$$\left[\vec{\Psi}_z\right]_{ii} \leftarrow \sum_n \omega_{zn}\left[\vec{x}_n-\Lambda_z\vec{y}_n-\vec{\mu}_z\right]_i^2, \tag{15}$$

$$P(z) \leftarrow \frac{\sum_n\gamma_{zn}}{\sum_{z'n'}\gamma_{z'n'}}. \tag{16}$$

The total log-likelihood, $\mathcal{L}=\sum_n \log P(\vec{x}_n,\vec{y}_n)$, can also be computed at each iteration, with the marginal likelihood $P(\vec{x}_n,\vec{y}_n)$ for each input-output pair given by the denominator of Equation (10). The updates in Equations (11–16) are derived in a standard way from the auxiliary function for EM algorithms (Dempster et al., 1977). Thus, they lead to monotonic improvement in the total log-likelihood and converge to a stationary point in the model's parameter space.

References

H. Attias. Independent factor analysis. *Neural Computation*, 11(4):803–851, 1999.

Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 2000.

M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 585–591, Cambridge, MA, 2002. MIT Press.

Y. Bengio, P. Vincent, and J.F. Paiement. Learning eigenfunctions of similarity: linking spectral clustering and kernel PCA. Technical Report 1232, Departement d'Informatique et Recherche Oprationnelle, Universite de Montreal, 2003.

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

C. Bishop, M. Svensen, and C. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215, 1998.

C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1996.

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.

M. Brand and K. Huang. A unifying theorem for spectral embedding and clustering. In C. M. Bishop and B. J. Frey, editors, *Proceedings of the Ninth International Workshop on Aritficial Intelligence and Statistics*, Key West, FL, January 2003.

C. Bregler and S. Omohundro. Nonlinear image interpolation using manifold learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 973–980, Cambridge, MA, 1995. MIT Press.

E. Cosatto and H. P. Graf. Sample-based synthesis of photo-realistic talking-heads. In *Proceedings of Computer Animation*, pages 103–110. IEEE Computer Society, 1998.

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

P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The Helmholtz machine. *Neural Computation*, 7(5):889–904, 1995.

V. de Silva and J. Tenenbaum. Unsupervised learning of curved manifolds. In *Proceedings of the MSRI workshop on nonlinear estimation and classification*. Springer Verlag, 2002.

D. DeCoste. Visualizing Mercel kernel feature spaces via kernelized locally linear embedding. In *Proceedings of the Eighth International Conference on Neural Information Processing (ICONIP-01)*, Shanghai, China, November 2001.

S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

D. DeMers and G.W. Cottrell. Nonlinear dimensionality reduction. In D. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 580–587, San Mateo, CA, 1993. Morgan Kaufmann.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–37, 1977.

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.

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.

R. Durbin and D. Wilshaw. An analogue approach to the traveling salesman problem using an elastic net method. *Nature*, 326:689–691, 1987.

D.R. Fokkema, G.L.G. Sleijpen, and H.A. van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal Scientific Computing*, 20(1):94–125, 1998.

J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

K. Fukunaga and D. R. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 20(2):176–193, 1971.

Z. Ghahramani and G. E. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1 (revised February 1997), Department of Computer Science, University of Toronto, May 1996.

A. G. Gray and A. W. Moore. N-Body problems in statistical learning. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 521–527, Cambridge, MA, 2001. MIT Press.

J. H. Ham, D. D. Lee, and L. K. Saul. Learning high dimensional correspondences from low dimensional manifolds. Submitted for publication, 2003.

T. J. Hastie and W. Stuetzle. Principal curves and surfaces. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

G. E. Hinton, P. Dayan, and M. Revow. Modeling the manifolds of handwritten digits. *IEEE Transactions on Neural Networks*, 8:65–74, 1997.

G. E. Hinton and M. Revow. Using mixtures of factor analyzers for segmentation and pose estimation. Unpublished, 1998.

G. E. Hinton and T. J. Sejnowski, editors. *Unsupervised Learning and Map Formation: Foundations of Neural Computation*. MIT Press, Cambridge, MA, 1999.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1990.

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

A. Hyvärinen. Independent component analysis in the presence of gaussian noise by maximizing joint likelihood. *Neurocomputing*, 22:49–67, 1998.

P. Indyk. Dimensionality reduction techniques for proximity problems. In *Proceedings of the Eleventh ACM-SIAM Symposium on Discrete Algorithms (SODA '00)*, pages 371–378, 2000.

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

G. G. Judge and T. T. Takayama. Inequality restrictions in regression analysis. *Journal of the American Statistical Association*, 61(313):166–181, March 1966.

N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9:1493–1516, 1997.

D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the Thirty Fourth ACM Symposium on the Theory of Computing (STOC '02)*, pages 741–750, 2002.

B. Kegl. Intrinsic dimension estimation using packing numbers. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

H. Klock and J. Buhmann. Data visualization by multidimensional scaling: a deterministic annealing approach. *Pattern Recognition*, 33:651, 1999.

T. Kohonen. *Self-organization and Associative Memory*. Springer-Verlag, Berlin, 1988.

M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37:233, 1991.

Y. LeCun, L. Bottou, G. Orr, and K.-R. Müller. Efficient backprop. In G. Orr and Müller K.-R., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.

M. L. Littman, D. F. Swayne, N. Dean, and A. Buja. Visualizing the embedding of objects in Euclidean space. In H. J. N. Newton, editor, *Computing Science and Statistics: Proceedings of the 24th Symposium on the Interface*, pages 208–217. Interface Foundation of North America, 1992.

T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7:507, 1994.

G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, 1988.

M. Meila and J. Shi. Learning segmentation by random walks. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 873–879, Cambridge, MA, 2000. MIT Press.

A. W. Moore, A. Connolly, C. Genovese, A. Gray, L. Grone, N. Kanidoris II, R. Nichol, J. Schneider, A. Szalay, I. Szapudi, and L. Wasserman. Fast algorithms and efficient statistics: N-point correlation functions. In *Proceedings of the MPA/MPE/ESO Conference on Mining the Sky*, 2000.

A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856, Cambridge, MA, 2002. MIT Press.

S. Omohundro. Five balltree construction algorithms. Technical Report TR-89-063, International Computer Science Institute, December 1989.

S. Omohundro. Bumptrees for efficient function, constraint, and classification learning. In R. Lippmann, J. Moody, and D. Touretzky, editors, *Advances in Neural Information Processing 3*, pages 693–699, San Mateo, CA, 1991. Morgan Kaufmann.

P. Perona and M. Polito. Grouping and dimensionality reduction by locally linear embedding. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1255–1262, Cambridge, MA, 2002. MIT Press.

K. Pettis, T. Bailey, A. Jain, and R. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 1(1):25–37, 1979.

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

W. H. Press, S. E. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1993.

S. T. Roweis. EM algorthms for PCA and SPCA. In M. Kearns, M. Jordan, and S. Solla, editors, *Advances in neural information processing systems 10*, pages 626–632, Cambridge, MA, 1998. MIT Press.

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

S. T. Roweis, L. K. Saul, and G. E. Hinton. Global coordination of locally linear models. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 889–896, Cambridge, MA, 2002. MIT Press.

D. B. Rubin and D. T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47:69–76, 1982.

L. K. Saul and J. B. Allen. Periodic component analysis: an eigenvalue method for representing periodic structure in speech. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 807–813, Cambridge, MA, 2001. MIT Press.

L. K. Saul and M. G. Rahim. Maximum likelihood and minimum classification error factor analysis for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 8(2): 115–125, 1999.

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

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

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

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

Y. Takane and F. W. Young. Nonmetric individual differences multidimensional scaling: an alternating least squares method with optimal scaling features. *Psychometrika*, 42:7, 1977.

R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2), 1972.

R. Tarjan. Data structures and network algorithms. In *CBMS*, volume 44. Society for Industrial and Applied Mathematics, 1983.

Y. W. Teh and S. T. Roweis. Automatic alignment of hidden representations. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

J. Tenenbaum. Mapping a manifold of perceptual observations. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, *Advances in Neural Information Processing Systems 10*, pages 682–688, Cambridge, MA, 1998. MIT Press.

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

M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

J. J. Verbeek, N. Vlassis, and B. Kröse. Coordinating mixtures of probabilistic principal component analyzers. Technical Report IAS-UVA-02-01, Computer Science Institute, University of Amsterdam, The Netherlands, February 2002a.

J.J. Verbeek, N. Vlassis, and B. Kröse. A k-segments algorithm for finding principal curves. *Pattern Recognition Letters*, 23(8):1009–1017, 2002b.

N. Vlassis, Y. Motomura, and B. Kröse. Supervised dimension reduction of intrinsically low-dimensional data. *Neural Computation*, 14(1):191–215, 2002.

Y. Weiss. Segmentation using eigenvectors: a unifying view. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 975–982, 1999.

C. K. I. Williams. On a connection between kernel PCA and metric multidimensional scaling. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 675–681, Cambridge, MA, 2001. MIT Press.

S. Yu and J. Shi. Grouping with directed relationships. In M. Figueiredo, J. Zerubia, and A. K. Jain, editors, *Proceedings of the Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, (EMMCVPR-01)*, volume 2134 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2001.