An Introduction to Locally Linear Embedding

Lawrence K. Saul
AT&T Labs – Research
180 Park Ave, Florham Park, NJ 07932 USA
lsaul@research.att.com

Sam T. Roweis
Gatsby Computational Neuroscience Unit, UCL
17 Queen Square, London WC1N 3AR, UK
roweis@gatsby.ucl.ac.uk

Abstract

Many problems in information processing involve some form of dimensionality reduction. Here we describe locally linear embedding (LLE), an unsupervised learning algorithm that computes low dimensional, neighborhood preserving embeddings of high dimensional data. LLE attempts to discover nonlinear structure in high dimensional data by exploiting the local symmetries of linear reconstructions. Notably, LLE maps its inputs into a single global coordinate system of lower dimensionality, and its optimizations—though capable of generating highly nonlinear embeddings—do not involve local minima. We illustrate the method on images of lips used in audiovisual speech synthesis.

1 Introduction

Many problems in statistical pattern recognition begin with the preprocessing of multidimensional signals, such as images of faces or spectrograms of speech. Often, the goal of preprocessing is some form of dimensionality reduction: to compress the signals in size and to discover compact representations of their variability.

Two popular forms of dimensionality reduction are the methods of principal component analysis (PCA) [1] and multidimensional scaling (MDS) [2]. Both PCA and MDS are eigenvector methods designed to model linear variabilities in high dimensional data. In PCA, one computes the linear projections of greatest variance from

the top eigenvectors of the data covariance matrix. In classical (or metric) MDS, one computes the low dimensional embedding that best preserves pairwise distances between data points. If these distances correspond to Euclidean distances, the results of metric MDS are equivalent to PCA. Both methods are simple to implement, and their optimizations do not involve local minima. These virtues account for the widespread use of PCA and MDS, despite their inherent limitations as linear methods.

Recently, we introduced an eigenvector method—called locally linear embedding (LLE)—for the problem of nonlinear dimensionality reduction[4]. This problem is illustrated by the nonlinear manifold in Figure 1. In this example, the dimensionality reduction by LLE succeeds in identifying the underlying structure of the manifold, while projections of the data by PCA or metric MDS map faraway data points to nearby points in the plane. Like PCA and MDS, our algorithm is simple to implement, and its optimizations do not involve local minima. At the same time, however, it is capable of generating highly nonlinear embeddings. Note that mixture models for local dimensionality reduction[5, 6], 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 this paper, we review the LLE algorithm in its most basic form and illustrate a potential application to audiovisual speech synthesis[3].

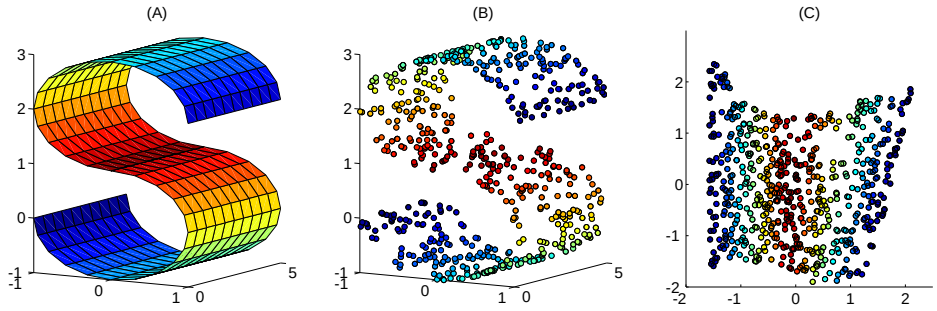

Figure 1: The problem of nonlinear dimensionality reduction, as illustrated for three dimensional data (B) sampled from a two dimensional manifold (A). An unsupervised learning algorithm must discover the global internal coordinates of the manifold without signals that explicitly indicate how the data should be embedded in two dimensions. The shading in (C) illustrates the neighborhood-preserving mapping discovered by LLE.

2 Algorithm

The LLE algorithm, summarized in Fig. 2, is based on simple geometric intuitions. Suppose the data consist of N real-valued vectors $\vec{X}_i$, each of dimensionality D, sampled from some 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.

We can characterize the local geometry of these patches by linear coefficients that reconstruct each data point from its neighbors. In the simplest formulation of LLE, one identifies K nearest neighbors per data point, as measured by Euclidean distance. (Alternatively, one can identify neighbors by choosing all points within a ball of fixed radius, or by using more sophisticated rules based on local metrics.) Reconstruction errors are then measured by the cost function:

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

which adds up the squared distances between all the data points and their reconstructions. The weights W_{ij} summarize the contribution of the jth data point to the ith reconstruction. To compute the weights W_{ij}, we minimize the cost function subject to two constraints: first, 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; second, that the rows of the weight matrix sum to one: $\sum_j W_{ij} = 1$. The reason for the sum-to-one constraint will become clear shortly. The optimal weights W_{ij} subject to these constraints are found by solving a least squares problem, as discussed in Appendix A.

Note that the constrained weights that minimize these reconstruction errors obey an important symmetry: for any particular data point, they are invariant to rotations, rescalings, and translations of that data point and its neighbors. The invariance to rotations and rescalings follows immediately from the form of eq. (1); the invariance to translations is enforced by the sum-to-one constraint on the rows of the weight matrix. A consequence of this symmetry is that the reconstruction weights characterize intrinsic geometric properties of each neighborhood, as opposed to properties that depend on a particular frame of reference.

Suppose the data lie on or near a smooth nonlinear manifold of dimensionality $d \ll D$. To a good approximation, then, 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 intrinsic geometric properties of the

data that are invariant to exactly such transformations. We therefore expect their characterization of local geometry in the original data space to be equally valid for local patches on the manifold. In particular, the same weights W_{ij} that reconstruct the ith data point 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 placing them in the low dimensional embedding space. Assume further that this operation is done in a way that preserves the angles formed by each data point to its nearest neighbors. In this case, the transplantation of each patch involves no more than a translation, rotation, and rescaling of its data, exactly the operations to which the weights are invariant. Thus, when the patch arrives at its low dimensional destination, we expect the same weights to reconstruct each data point from its neighbors.)

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

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

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 coordinates $\vec{Y}_i$. The embedding cost in Eq. (2) defines a quadratic form in the vectors $\vec{Y}_i$. Subject to constraints that make the problem well-posed, it can be minimized by solving a sparse $N \times N$ eigenvector problem, whose bottom d non-zero eigenvectors provide an ordered set of orthogonal coordinates centered on the origin. Details of this eigenvector problem are discussed in Appendix B.

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 in connected components of the graph defined by the weight matrix. The different dimensions in the embedding space can be computed successively; this is done simply by computing the bottom eigenvectors from eq. (2) one at a time. But the computation is always coupled across data points. This is how the algorithm leverages overlapping local information to discover global structure.

Implementation of the algorithm is fairly straightforward, as the algorithm has only one free parameter: the number of neighbors per data point, K. Once neighbors

LLE ALGORITHM

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 eq. (1) by constrained linear fits.
3. Compute the vectors $\vec{Y}_i$ best reconstructed by the weights W_{ij}, minimizing the quadratic form in eq. (2) by its bottom nonzero eigenvectors.

Figure 2: Summary of the LLE algorithm, mapping high dimensional data points, $\vec{X}_i$, to low dimensional embedding vectors, $\vec{Y}_i$.

are chosen, the optimal weights W_{ij} and coordinates Y_i are computed by standard methods in linear algebra. The algorithm involves a single pass through the three steps in Fig. 2 and finds global minima of the reconstruction and embedding costs in Eqs. (1) and (2). As discussed in Appendix A, in the unusual case where the neighbors outnumber the input dimensionality ($K > D$), the least squares problem for finding the weights does not have a unique solution, and a regularization term—for example, one that penalizes the squared magnitudes of the weights—must be added to the reconstruction cost.

The algorithm, as described in Fig. 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 pairwise distance between different data points. A simple variation of LLE, described in Appendix C, can be applied to input of this form. In this way, matrices of pairwise distances can be analyzed by LLE just as easily as MDS[2]; in fact only a small fraction of all possible pairwise distances (representing distances between neighboring points and their respective neighbors) are required for running LLE.

3 Examples

The embeddings discovered by LLE are easiest to visualize for intrinsically two dimensional manifolds. In Fig. 1, for example, the input to LLE consisted $N = 600$ data points sampled off the S-shaped manifold. The resulting embedding shows how the algorithm, using $K = 12$ neighbors per data point, successfully unraveled the underlying two dimensional structure.

Fig. 3 shows another two dimensional manifold, this 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 uncorrelated from one example to the next. The only consistent structure in the resulting images thus described 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 simple to visualize, the manifold of translated faces is highly nonlinear in the high dimensional ($D = 3009$) vector space of pixel coordinates. The bottom portion of Fig. 3 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.

Finally, in addition to these examples, for which the true manifold structure was known, we also applied LLE to images of lips used in the animation of talking heads[3]. Our database contained $N = 8588$ color (RGB) images of lips at 108×84 resolution. Dimensionality reduction of these images ($D = 27216$) is useful for faster and more efficient animation. The top and bottom panels of Fig. 4 show the first two components discovered, respectively, by PCA and LLE (with $K = 16$). If the lip images described a nearly linear manifold, these two methods would yield similar results; thus, the significant differences in these embeddings reveal the presence of nonlinear structure. Note that while the linear projection by PCA has a somewhat uniform distribution about its mean, the locally linear embedding has a distinctly spiny structure, with the tips of the spines corresponding to extremal configurations of the lips.

4 Discussion

It is worth noting that many popular learning algorithms for nonlinear dimensionality reduction do not share the favorable properties of LLE. Iterative hill-climbing methods for autoencoder neural networks[7, 8], self-organizing maps[9], and latent variable models[10] do not have the same guarantees of global optimality or convergence; they also tend to involve many more free parameters, such as learning rates, convergence criteria, and architectural specifications.

The different steps of LLE have the following complexities. In Step 1, 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

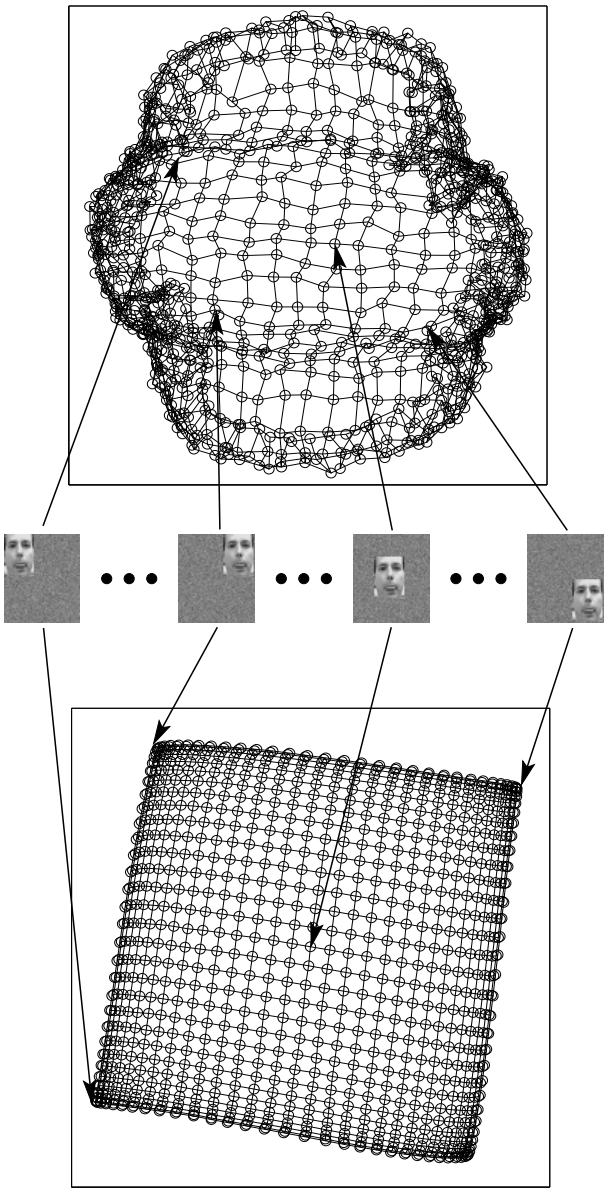

Figure 3: The results of PCA (top) and LLE (bottom), applied to images of a single face translated across a two-dimensional background of noise. Note how LLE maps the images with corner faces to the corners of its two dimensional embedding, while PCA fails to preserve the neighborhood structure of nearby images.

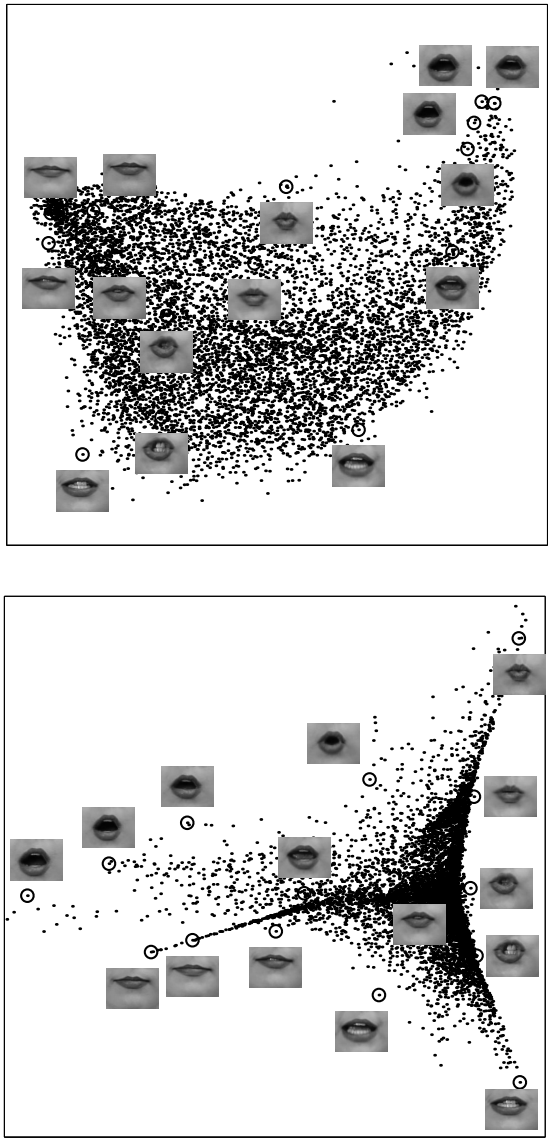

Figure 4: Images of lips mapped into the embedding space described by the first two coordinates of PCA (top) and LLE (bottom). Representative lips are shown next to circled points in different parts of each space. The differences between the two embeddings indicate the presence of nonlinear structure in the data.

data distributions, however – and especially for data distributed on a thin submanifold of the observation space – constructions such as K-D trees can be used to compute the neighbors in $O(N \log N)$ time[13]. In Step 2, computing the reconstruction weights 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. In Step 3, 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. Methods for sparse eigenproblems[14], however, can be used to reduce the complexity to subquadratic in N. Note that as more dimensions are added to the embedding space, the existing ones do not change, so that LLE does not have to be rerun to compute higher dimensional embeddings. The storage requirements of LLE are limited by the weight matrix which is size N by K.

LLE illustrates a general principle of manifold learning, elucidated by Tenenbaum et al[11], that overlapping local neighborhoods—collectively analyzed—can provide information about global geometry. Many virtues of LLE are shared by the Isomap algorithm[11], which has been successfully applied to similar problems in nonlinear dimensionality reduction. Isomap is an extension of MDS in which embeddings are optimized to preserve "geodesic" distances between pairs of data points; these distances are estimated by computing shortest paths through large sublattices of data. A virtue of LLE is that it avoids the need to solve large dynamic programming problems. LLE also tends to accumulate very sparse matrices, whose structure can be exploited for savings in time and space.

LLE is likely to be even more useful in combination with other methods in data analysis and statistical learning. An interesting and important question is how to learn a parametric mapping between the observation and embedding spaces, given the results of LLE. One possible approach is to use $\{\vec{X}_i, \vec{Y}_i\}$ pairs as labeled examples for statistical models of supervised learning. The ability to learn such mappings should make LLE broadly useful in many areas of information processing.

A Constrained Least Squares Problem

The constrained weights that best reconstruct each data point from its neighbors 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:

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

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 covariance matrix,

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

This error can be minimized in closed form, using a Lagrange multiplier to enforce the constraint that $\sum_j w_j = 1$. In terms of the inverse local covariance matrix, the optimal weights are given by:

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

The solution, as written in eq. (5), appears to require an explicit inversion of the local covariance matrix. In practice, a more efficient way to minimize the error is simply to solve the linear system of equations, $\sum_j C_{jk} w_k = 1$, and then to rescale the weights so that they sum to one (which yields the same result). By construction, the local covariance matrix in eq. (4) is symmetric and semipositive definite. If the covariance matrix is singular or nearly singular—as arises, for example, when there are more neighbors than input dimensions ($K > D$), or when the data points are not in general position—it can be conditioned (before solving the system) by adding a small multiple of the identity matrix,

$$C_{jk} \leftarrow C_{jk} + \left(\frac{\Delta^2}{K}\right) \delta_{jk}, \tag{6}$$

where Δ^2 is small compared to the trace of C. This amounts to penalizing large weights that exploit correlations beyond some level of precision in the data sampling process.

B Eigenvector Problem

The embedding vectors $\vec{Y}_i$ are found by minimizing the cost function, eq. (2), for fixed weights W_{ij}:

$$\min_Y \quad \Phi(Y) \quad = \quad \sum_i \left| \vec{Y}_i - \sum\nolimits_j W_{ij} \vec{Y}_j \right|^2 . \tag{7}$$

Note that the cost defines a quadratic form,

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

involving inner products of the embedding vectors and the $N \times N$ matrix M:

$$M_{ij} = \delta_{ij} - W_{ij} - W_{ji} + \sum_k W_{ki} W_{kj}, \tag{8}$$

where δ_{ij} is 1 if $i = j$ and 0 otherwise.

This optimization is performed subject to constraints that make the problem well posed. It is clear that the coordinates $\vec{Y}_i$ can be translated by a constant displacement without affecting the cost, $\Phi(Y)$. We remove this degree of freedom by requiring the coordinates to be centered on the origin:

$$\sum_i \vec{Y}_i = \vec{0}. \tag{9}$$

Also, to avoid degenerate solutions, we constrain the embedding vectors to have unit covariance, with outer products that satisfy

$$\frac{1}{N} \sum_i \vec{Y}_i \vec{Y}_i^\top = I, \tag{10}$$

where I is the $d \times d$ identity matrix. Note that there is no loss in generality in constraining the covariance of $\vec{Y}$ to be diagonal and of order unity, since the cost function in eq. (2) is invariant to rotations and homogeneous rescalings. The further constraint that the covariance is equal to the identity matrix expresses an assumption that reconstruction errors for different coordinates in the embedding space should be measured on the same scale.

The optimal embedding—up to a global rotation of the embedding space—is found by computing the bottom $d + 1$ eigenvectors of the matrix, M; this is a version of the Rayleitz-Ritz theorem[12]. The bottom eigenvector of this matrix, which we discard, is the unit vector with all equal components; it represents a free translation mode of eigenvalue zero. Discarding this eigenvector enforces the constraint that the embeddings have zero mean, since the components of other eigenvectors must sum to zero, by virtue of orthogonality. The remaining d eigenvectors form the d embedding coordinates found by LLE.

Note that the bottom $d+1$ eigenvectors of the matrix M (that is, those corresponding to its smallest $d+1$ eigenvalues) can be found without performing a full matrix diagonalization[14]. Moreover, the matrix M can be stored and manipulated as the sparse symmetric matrix

$$M = (I - W)^T (I - W), \tag{11}$$

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

$$Mv = (v - Wv) - W^\top (v - Wv), \tag{12}$$

requiring just one multiplication by W and one multiplication by $W^\top$, both of which are extremely sparse. Thus, the matrix M never needs to be explicitly created or stored; it is sufficient to store and multiply the matrix W.

C LLE from Pairwise Distances

LLE can be applied to user input in the form of pairwise distances. In this case, nearest neighbors are identified by the smallest non-zero elements of each row in the distance matrix. To derive the reconstruction weights for each data point, we need to compute the local covariance matrix C_{jk} between its nearest neighbors, as defined by eq. (4) in appendix A. This can be done by exploiting the usual relation between pairwise distances and dot products that forms the basis of metric MDS[2]. Thus, for a particular data point, we set:

$$C_{jk} = \frac{1}{2} \left(D_j + D_k - D_{jk} - D_0 \right), \tag{13}$$

where D_{jk} denotes the squared distance between the jth and kth neighbors, $D_\ell = \sum_z D_{\ell z}$ and $D_0 = \sum_{jk} D_{jk}$. In terms of this local covariance matrix, the reconstruction weights for each data point are given by eq. (5). The rest of the algorithm proceeds as usual.

Note that this variant of LLE requires significantly less user input than the complete matrix of pairwise distances. Instead, for each data point, the user needs only to specify its nearest neighbors and the submatrix of pairwise distances between those neighbors. Is it possible to recover manifold structure from even less user input—say, just the pairwise distances between each data point and its nearest neighbors? A simple counterexample shows that this is not possible. Consider the square lattice of three dimensional data points whose integer coordinates sum to zero. Imagine that points with even x-coordinates are colored black, and that points with odd x-coordinates are colored red. The "two point" embedding that maps all black points to the origin and all red points to one unit away preserves the distance between each point and its four nearest neighbors. Nevertheless, this embedding completely fails to preserve the underlying structure of the original manifold.

Acknowledgements

The authors thank E. Cosatto, H.P. Graf, and Y. LeCun (AT&T Labs) and B. Frey (U. Toronto) for providing data for these experiments. S. Roweis acknowledges the support of the Gatsby Charitable Foundation, the National Science Foundation, and the National Sciences and Engineering Research Council of Canada.

References

[1] I.T. Jolliffe, *Principal Component Analysis* (Springer-Verlag, New York, 1989).

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

[3] E. Cosatto and H.P. Graf. Sample-Based Synthesis of Photo-Realistic Talking-Heads. *Proceedings of Computer Animation*, 103–110. IEEE Computer Society (1998).

[4] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science* **290**, 2323-2326 (2000).

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

[6] N. Kambhatla and T. K. Leen. Dimension reduction by local principal component analysis. *Neural Computation* **9**, 1493–1516 (1997).

[7] D. DeMers and G.W. Cottrell. Nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems 5*, D. Hanson, J. Cowan, L. Giles, Eds. (Morgan Kaufmann, San Mateo, CA, 1993), pp. 580–587.

[8] M. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* **37**, 233–243 (1991).

[9] T. Kohonen. *Self-organization and Associative Memory* (Springer-Verlag, Berlin, 1988).

[10] C. Bishop, M. Svensen, and C. Williams. GTM: The generative topographic mapping. *Neural Computation* **10**, 215–234 (1998).

[11] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science* **290**, 2319-2323 (2000).

[12] R. A. Horn and C. R. Johnson. *Matrix Analysis* (Cambridge University Press, Cambridge, 1990).

[13] 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(3), 290-226 (1977).

[14] 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).