# Contents

# Local tangent space embedding

**Abstract**

Nonlinear manifold learning techniques, such as Isomap and Local Linear Embedding, attempt to recover a low-dimensional characterization of high-dimensional data by measuring Euclidean distances and reconstructions between nearby data points. Unfortunately, Euclidean differences usually do not correspond to natural transformations of typical data, even locally. For example, in image or document data, Euclidean distances rarely correspond to a meaningful

perceptual difference between nearby objects. In this research we attempt to improve the quality of manifold learning techniques by modeling local neighborhoods in terms of natural transformations between data—for example, by allowing image operations that extend simple differences and linear combinations. We introduce the idea of modeling local tangent spaces of the manifold in terms of these richer transformations. Given a local tangent space representation, we then embed data in a lower dimensional coordinate system while preserving not only local distances and weighted combinations between data, but also the coordinate systems of the local tangent spaces. This leads to improved manifold discovery in natural image data.

## 1 Introduction

Manifold learning is a significant problem across a wide variety of information processing fields including pattern recognition, data compression, machine learning, and database navigation. In many problems, the measured data vectors are high-dimensional but we may have reason to believe that the data lie near a lower-dimensional manifold. In other words, we may believe that high-dimensional data are multiple, indirect measurements of an underlying source, which typically cannot be directly measured. Learning a suitable low-dimensional manifold from high-dimensional data is essentially the same as learning this underlying source. One example of this general approach is measuring the underlying intelligence and mental abilities of a person via psychological and educational tests. In these tests, answers to questions are used as an indirect measurement of mental ability.

Dimensionality reduction [1] can also be seen as the process of deriving a set of degrees of freedom which can be used to reproduce most of the variability of a data set. Consider a set of images produced by the rotation of a face through different angles. Clearly only one degree of freedom is being altered, and thus the images lie along a continuous curve through image space.

Manifold learning techniques can be used in different ways including:

- Data dimensionality reduction: Produces a compact low-dimensional encoding of a given high-dimensional data set.
- Data visualization: Provides an interpretation of a given data set, usually as a by-product of data dimensionality reduction.
- Preprocessing for supervised learning: Unsupervised methods for data dimensionality reduction are used as a preprocessing step in order to simplify subsequent training of a supervised method such as classification.

Many algorithms for dimensionality reduction have been developed to accomplish these tasks. However, since the need for such analysis is raised in many areas of study, contributions to the field have come from many disciplines. While all of these methods have a similar goal, approaches to the problem are different.

Principal components analysis (PCA) [1] is a classical method which provides a sequence of best linear approximations to a given high-dimensional observation. It is one of the most popular techniques for dimensionality reduction. However, its effectiveness is limited by its global linearity. Multidimensional scaling (MDS) [2], which is closely related to PCA, suffers from the same drawback. Factor analysis [3, 4] and independent component analysis (ICA) [5] also assume that the underling manifold is a linear subspace. However, they differ from PCA in the way they model the subspace.

[1] In this document 'manifold learning' and 'dimensionality reduction' are used interchangably.

The subspace modeled by PCA captures the maximum variability in the data, and can be viewed as modeling the covariance structure of the data, whereas factor analysis models the correlation structure. ICA starts from a factor analysis solution and searches for rotations that lead to independent components [4, 6].

In order to resolve the problem of dimensionality reduction in nonlinear cases, many techniques including kernel PCA [7, 8] , locally linear embedding (LLE) [9, 10], and Isomap [11, 12] have been proposed.

This document provides a brief overview of these different approaches and shows their close connection. Afterwards, it motivates and proposes a new research direction which improves the quality of manifold learning techniques by modeling local neighborhoods in terms of natural transformations between data points. Our proposed technique preserves not only local distances and weighted combinations between data, but also the coordinate systems of the local tangent spaces. Section 2 of this document explains Principal components analysis which is the core of many other techniques. In Section 3, kernel PCA, a recent extension to PCA, is discussed. Locally linear embedding, a new and very popular algorithm, is reviewed in Section 4 and its connection to kernel PCA is illustrated. Multidimensional scaling and its recent extension, Isomap, are discussed in Sections 5 and 6 respectively. The last section is devoted to our proposed technique and some preliminary results.

## 2 Principal components analysis

Principal components analysis (PCA) is a very popular technique for dimensionality reduction. Given a set of data on $n$ dimensions, PCA aims to find a linear subspace of dimension lower than $n$ such that the data points lie mainly on this linear subspace. Such a reduced subspace attempts to maintain most of the variability of the data.

The linear subspace can be specified by $d$ orthogonal vectors which form a new coordinate system and are called the 'principal components'. The principal components are orthogonal, linear transformations of the original data points, so there can be no more than $n$ of them. However, the hope is that only $d < n$ principal components are needed to approximate the space spanned by the $n$ original axes.

The most common definition of PCA, due to Hotelling [13], is that, for a given set of data vectors $x_i$, $i \in 1...t$, the $d$ principal axes are those orthonormal axes onto which the variance retained under projection is maximal.

In order to capture as much of the variability as possible, let's choose the first principal component, denoted by $U_1$, to have maximum variance. Suppose that all centered observations are stacked into the columns of an $n \times t$ matrix $X$, where each column corresponds to an $n$ dimensional observation and there are $t$ observations. Let principal component be a linear combination of $X$ defined by coefficients (or weights) $W = [w_1...w_t]$. In matrix form:

$$U_1 = W^T X$$
$$var(U_1) = var(W^T X) = W^T S W$$

where $S$ is the $t \times t$ sample covariance matrix of $X$.

Clearly $var(U_1)$ can be made arbitrarily large by increasing the magnitude of $W$. Therefore, we choose $W$ to maximize $W^T S W$ while constraining $W$ to have unit length.

$$\max\ W^T S W$$
$$subject\ to\ W^T W = 1$$

To solve this optimization problem a Lagrange multiplier $\alpha_1$ is introduced:

$$L(W, \alpha) = W^T S W - \alpha_1 (W^T W - 1) \tag{1}$$

Differentiating with respect to $W$ gives $t$ equations,

$$SW = \alpha_1 W$$

Premultiplying both sides by $W^T$ we have:

$$W^T S W = \alpha_1 W^T W = \alpha_1$$

$var(U_1)$ is maximized if $\alpha_1$ is the largest eigenvalue of $S$.

Clearly $\alpha_1$ and $W$ are an eigenvalue and an eigenvector of $S$. Differentiating (1) with respect to the Lagrange multiplier $\alpha_1$ gives us back the constraint:

$$W^T W = 1$$

This shows that the first principal component is given by the normalized eigenvector with the largest associated eigenvalue of the sample covariance matrix $S$. A similar argument can show that the $d$ dominant eigenvectors of covariance matrix $S$ determine the first $d$ principal components.

Another nice property of PCA, closely related to the original discussion by Pearson [14], is that the projection onto the principal subspace minimizes the squared reconstruction error, $\sum_{i=1}^{t} ||x_i - \hat{x}_i||^2$. In other words, the principal components of a set of data in $\Re^n$ provide a sequence of best linear approximations to that data, for all ranks $d \leq n$

Consider the rank-$d$ linear approximation model as :

$$f(\lambda) = \bar{x} + U_d \lambda$$

This is the parametric representation of a hyperplane of rank $d$.

For convenience, suppose $\bar{x} = 0$ (otherwise the observations can be simply replaced by their centered versions $\tilde{x} = x_i - \bar{x}$). Under this assumption the rank $d$ linear model would be $f(\lambda) = U_d \lambda$, where $U_d$ is a $n \times d$ matrix with $d$ orthogonal unit vectors as columns and $\lambda$ is a vector of parameters. Fitting this model to the data by least squares leaves us to minimize the reconstruction error:

$$\min_{U_d, \lambda_i} \sum_{i}^{t} ||x_i - U_d \lambda_i||^2$$

By partial optimization for $\lambda_i$ we obtain:

$$\frac{d}{d\lambda_i} \Rightarrow \lambda_i = U_d^T x_i$$

Now we need to find the orthogonal matrix $U_d$:

$$\min_{U_d} \sum_{i}^{t} ||x_i - U_d U_d^T x_i||^2$$

Define $H_d = U_d U_d^T$. $H_d$ is a $n \times n$ matrix which acts as a projection matrix and projects each data point $x_i$ onto its rank $d$ reconstruction.

In other words, $H_d x_i$ is the orthogonal projection of $x_i$ onto the subspace spanned by the columns of $U_d$. A unique $H^+$ solution can be obtained by finding the pseudo inverse of $X$, denoted as $X^+$.

$$H^+ = X^+ X$$
$$X = U\Sigma V^T$$
$$X^+ = V\Sigma^+ U^T$$
$$H^+ = U\Sigma V^T V\Sigma^+ U^T = UU^T$$

For each rank $d$, $U_d$ consists of the first $d$ columns of $U$.

---

**Algorithm 1**

**Recover basis:** Calculate $XX^\top = \sum_{i=1}^{t} x_i x_i^\top$ and let $U =$ eigenvectors of $XX^\top$ corresponding to the top $d$ eigenvalues.

**Encode training data:** $C = U^\top X$ where $C$ is a $d \times t$ matrix of encodings of the original data.

**Reconstruct training data:** $\hat{X} = UC = UU^\top X$.

**Encode test example:** $c = U^\top x$ where $c$ is a $d$ dimensional encoding of $x$.

**Reconstruct test example:** $\hat{x} = Uc = UU^\top x$.

---

Figure 1: Direct PCA Algorithm

Clearly the solution for $U$ can be expressed as singular value decomposition (SVD) of $X$.

$$X = U\Sigma V^T$$

since the columns of $U$ in the SVD contain the eigenvectors of $XX^T$. The PCA procedure is summarized in Algorithm 1.

## 3 Kernel PCA

Through the use of kernels, principle components can be computed efficiently in high-dimensional feature spaces that are related to the input space by some nonlinear mapping. PCA is an orthogonal transformation of the coordinate system in which we describe our data.

Kernel PCA finds principal components which are nonlinearly related to the input space. PCA can be formulated entirely in terms of dot products between data points. In kernel PCA, this dot product is replaced by the inner product of a Hilbert space. This is equivalent to performing PCA in the space produced by the nonlinear mapping, where the low-dimensional latent structure is, hopefully, easier to discover.

Consider a feature space $\mathcal{H}$ such that:

$$\Phi : x \to \mathcal{H}, x \mapsto \Phi(x)$$

Suppose $\sum_i^t \Phi(x_i) = 0$ (we will return to this point and show how this condition can be satisfied in Hilbert space).

This allows us to formulate the kernel PCA objective as follows:

$$\min \sum_i^t ||\Phi(x_i) - U_q U_q^T \Phi(x_i)||$$

By the same argument used for PCA, the solution can be found by SVD:

$$\Phi(X) = U \Sigma V^T$$

where $U$ contains the eigenvectors of $\Phi(X)\Phi(X)^T$

However, the singular value decomposition allows us to do much more than simply rederive the principle components algorithm. In fact, given the matrices $\Sigma$ and $V$, one can derive a *dual* form of principle components analysis which allows us to limit the direct dependence on the original dimensionality $n$, via the kernel trick.

Assume that the dimensionality $n$ of the $n \times t$ matrix of data $X$ is large (i.e. $n >> t$). In this case, Algorithm 1 is impractical. We would prefer a run time that depends only on the number of training examples $t$, or that at least has a reduced dependence on $n$.

To reduce the dependence on $n$, first assume that we have a kernel $k(\cdot, \cdot)$ that allows us to compute $k(x, y) = x^\top y$. Given such a function, we can then compute the matrix $X^\top X = K$, such that $k_{ij} = k(x_i, x_j)$. Let $[X^\top X]$ denote the fact that we could compute the matrix $X^\top X$ efficiently using the kernel trick.

The eigenvectors in $U$ corresponding to nonzero singular values in $\Sigma$ (square roots of eigenvalues) are in a one-to-one correspondence with the eigenvectors in $V$.

Now assume that we perform dimensionality reduction on $U$ and keep only the first $d$ eigenvectors, corresponding to the top $d$ nonzero singular values in $\Sigma$. These eigenvectors will still be in a one-to-one correspondence with the first $d$ eigenvectors in $V$:

$$X\,V \;=\; U\,\Sigma$$

where the dimensions of these matrices are:

$$\begin{array}{cccc} X & U & \Sigma & V \\ n \times t & n \times d & d \times d & t \times d \\ & & \text{diagonal} & \end{array}$$

Crucially, $\Sigma$ is now square and invertible, because its diagonal has nonzero entries. Thus, the following conversion between the top $d$ eigenvectors can be derived:

$$U \;=\; X\,V\,\Sigma^{-1} \tag{2}$$

Replacing all uses of $U$ in Algorithm 1 with $XV\Sigma^{-1}$ gives us the dual form of PCA, Algorithm 2 (see Figure 2).

In the derivation of the kernel PCA we assumed that $\Phi(x)$ has zero mean. The following normalization of the kernel satisfies this condition.

$$\tilde{k}(x,y) = k(x,y) - E_x[k(x,y)] - E_y[k(x,y)] + E_x[E_y[k(x,y)]]$$

In order to prove that, define:

$$\tilde{\Phi}(X) = \Phi(X) - E_x[\Phi(X)]$$

Finally, the corresponding kernel is:

$$\tilde{k}(x,y) = \tilde{\Phi}(x)\tilde{\Phi(y)}$$

This expands as follows:

$$\tilde{k}(x,y) = (\Phi(x) - E_x[\Phi(x)]).(\Phi(y) - E_y[\Phi(y)])$$
$$= k(x,y) - E_x[k(x,y)] - E_y[k(x,y)] + E_x[E_y[k(x,y)]]$$

**Algorithm 2**

**Recover basis:** Calculate $[X^\top X] = \sum_{i=1}^{t} k(x_i, x_i)$ and let $V$ = eigenvectors of $X^\top X$ corresponding to the top $d$ eigenvalues. Let $\Sigma$ = diagonal matrix of *square roots* of the top $d$ eigenvalues.

**Encode training data:** $C = U^\top X = \Sigma V^\top$ where $C$ is a $d \times t$ matrix of encodings of the original data.

**Reconstruct training data:** $\hat{X} = UC = U\Sigma V^\top = XV\Sigma^{-1}\Sigma V^\top = XVV^\top$.

**Encode test example:** $c = U^\top x = \Sigma^{-1}V^\top X^\top x = \Sigma^{-1}V^\top[X^\top x]$ where $c$ is a $d$ dimensional encoding of $x$.

**Reconstruct test example:** $\hat{x} = Uc = UU^\top x = XV\Sigma^{-2}V^\top X^\top x = XV\Sigma^{-2}V^\top[X^\top x]$.

Figure 2: Indirect, Dual Form of PCA Algorithm

## 4 Locally linear embedding

Locally linear embedding (LLE), computes low-dimensional, neighborhood preserving embeddings of high-dimensional data. A data set of dimensionality $n$, which is assumed to lie on or near a smooth nonlinear manifold of dimensionality $d < n$, is mapped into a single global coordinate system of lower dimensionality, $d$. The global nonlinear structure is recovered by locally linear fits.

Consider $t$ $n$-dimensional real-valued vectors $x_i$ sampled from some underlying manifold. We can assume each data point and its neighbors lie on, or are close to, a locally linear patch of the manifold. By a linear mapping, consisting of a translation, rotation, and rescaling, the high-dimensional coordinates of each neighborhood can be mapped to global internal coordinates on the manifold. Thus, the nonlinear structure of the data can be identified through two linear steps: first, compute the locally linear patches, and second, compute the linear mapping to the coordinate system on the manifold.

The main goal here is to map the high-dimensional data points to the single global coordinate system of the manifold such that the relationships between neighboring points are preserved. This proceeds in three steps:

1. Identify the neighbors of each data point $x_i$. This can be done by finding the $K$ nearest neighbors, or by choosing all points within some fixed radius, $\epsilon$.

2. Compute the weights that best linearly reconstruct $x_i$ from its neighbors.

3. Find the low-dimensional embedding vector $y_i$ which is best reconstructed by the weights determined in the previous step.

After finding the nearest neighbors in the first step, the second step must compute a local geometry for each locally linear patch. This geometry is characterized by linear coefficients that reconstruct each data point from its neighbors. The following cost function measures the reconstruction error.

$$\varepsilon(W) = \sum_{i=1}^{t} ||x_i - \sum_{j=1}^{K} W_{ij} x_j||^2$$

where the neighbors of $x_i$ are represented by $x_j$.

$W_{ij}$ are computed to minimize this cost function subject to the constraint that the rows of the weight matrix sum to one, $\sum_j W_{ij} = 1$. The optimal weights $W_{ij}$, subject to that constraint, are found by solving a constrained least squares problem.

The $i^{th}$ data point is reconstructed independently of all others, so the minimization can be done one term at a time. This is equivalent to solving a system of equations with $n$ equations and $K$ unknowns.

Ideally, the solution for the weights should be invariant under the linear mapping from the patch to the global coordinates on the manifold. This ensures that the reconstruction holds equally well in both high-dimensional and low-dimensional space. It can be shown that the weights are invariant to a transformation of the data point and its neighbors if and only if all rows of the weight matrix sum to 1.

Since the optimization can be determined one term at a time, the cost function can be minimized individually for each data point $x_i$.

$$\varepsilon_i(w) = ||x_i - \sum_{j=1}^{K} w_j \eta_j||^2$$

$$subject\ to \sum_j w_j = 1$$

where $\eta_j$ denotes neighbors of $i^{th}$ data point. We can rewrite this as:

$$\varepsilon_i(w) = ||x_i - Nw||^2 = ||\chi w - Nw||^2$$

where $\chi$ is the matrix of columns $x_i$ repeated $K$ times, and $N$ is the matrix of columns of $K$ nearest neighbors of $x_i$. Continuing,

$$\begin{aligned} \varepsilon_i(w) &= ||(\chi - N)w||^2 = ((\chi - N)w)^T((\chi - N)w) \\ &= w^T(\chi - N)^T(\chi - N) = w^T G w \end{aligned}$$

where $G = (\chi - N)^T(\chi - N)$. To accommodate the constraint that the weights sum to 1, a Lagrange multiplier, $\lambda$, is introduced. Let $e$ be a column vector of ones.

$$\begin{aligned} L(w, \lambda) &= w^T G w + \lambda(w^T e - e) \\ \frac{dL}{dw} &= 0 = 2Gw + \lambda e \\ Gw &= Ce \end{aligned}$$

In practice, we can solve this with $C$ set arbitrarily to 1 and then rescale so $w$ sums to 1.

The third step of LLE is done by choosing the $d-$dimensional vector $y_i$ to minimize the embedding cost function:

$$\Phi(Y) = \sum_{i=1}^{t} ||y_i - \sum_{j=1}^{t} W_{ij} y_j||^2$$

Note we are optimizing the coordinates $y_i$ while fixing the weights $W_{ij}$. This is equivalent to:

$$\Phi(Y) = \sum_{i=1}^{t} ||YI_i - Yw_i||^2$$

where $I_i$ is the $i^{th}$ column of the identity matrix, and $w_i$ is the $i^{th}$ column of $W$.

$$\min_Y \sum_{i=1}^{t} ||YI_i - Yw_i||^2 =$$

$$\min_Y \; trace \; [Y(I-W)(I-W)^T Y^T] = \min_Y \; trace(YMY^T)$$

The solution for $Y$ can have an arbitrary origin and orientation. In order to make the problem well-posed, these two degrees of freedom must be removed. Requiring the coordinates to be centered on the origin ($\sum_i y_i = 0$), and constraining the embedding vectors to have unit covariance ($Y^T Y = I$), removes the first and second degrees of freedom respectively.

The cost function can be optimized initially by the second of these two constraints:

$$L(Y, \lambda) = YMY^T + \lambda(YY^T - (N-1)I)$$

$$\frac{dL}{dY} = 0 = 2MY^T + 2\lambda Y$$

$$MY^T = \lambda Y^T$$

$L$ is minimized when the columns of $Y^T$ (rows of $Y$) are the eigenvectors associated with the lowest eigenvalues of $M$.

Discarding the eigenvector associated with eigenvalue 0 satisfies the first constraint.

## 4.1 Connection to kernel PCA

LLE can be cast as a special form of kernel PCA [8]. In kernel PCA, the first $d$ coordinates are eigenvectors associated with the $d$ greatest eigenvalues of the Gram matrix.

Let $\lambda$ be the largest eigenvalue of $(I-W)^T(I-W)$.

Define the $LLE$ kernel to be:

$$k_{LLE}(x_i, x_j) = (\lambda I - (I-W)^T(I-W))_{ij}$$

This kernel is, in fact, a similarity measure based on the similarity of the weights required to reconstruct two patterns in terms of $K$ neighboring patterns. Kernel PCA using this kernel provides the LLE embedding coefficients for a $d$-dimensional embedding as the first $d$ eigenvectors.

## 5 Multidimensional scaling (MDS)

Multidimensional scaling (MDS) is another classical approach that maps the original high dimensional space to a lower dimensional space that preserves pairwise distances. MDS addresses the problem of constructing a configuration of $t$ points in Euclidean space by using information about the distances between the $t$ patterns.

A $t \times t$ matrix $D$ is called a distance matrix if it is symmetric and

$$d_{rr} = 0, \quad d_{rs} > 0, \quad r \neq s$$

Given a distance matrix $D$, MDS finds $t$ data points $y_1, ..., y_t$ in $d$ dimensions such that if $\hat{d}_{rs}$ denotes the Euclidean distance between $y_r$ and $y_s$, then $\hat{D}$ is similar to $D$.

Now, for a distance matrix $D$, let

$$A = (a_{rs}), \quad a_{rs} = -\frac{1}{2} d_{rs}^2$$

and set

$$B = HAH$$

where $H$ is a centering matrix defined as $H = I - \frac{1}{n} ee^T$, and $e$ is a column vector containing all ones. When $D$ is the distance matrix of $X$, the original input data, $B$ can be interpreted as the centered inner product matrix for $X$.

$$B = (X - \bar{X})^T = (X - \bar{X})$$

or in different form $B = (HX)(HX)^T$. Let the eigendecomposition of $B$ be $B = V\Lambda V^T$, where $\Lambda$ is a diagonal matrix and $V$ is a matrix whose columns are the eigenvectors of $B$. If $B$ is positive semi-definite of rank $p$, then a configuration corresponding to $B$ can be constructed as follows:

Suppose $\lambda_1 \geq .... \geq \lambda_p$ are the positive eigenvalues of $B$ with corresponding eigenvectors $Y = (Y_{(1)}, ...Y_{(p)})$ normalized by

$$Y_{(i)}^T Y_{(i)} = \lambda_i, \quad i = 1, ..., p$$

Then the points $Y_r$ have interpoint distance given by $D$.

MDS treats the distance matrix $D$ as the starting point. However, all other techniques introduced in previous sections start with a data matrix $X$. Similar to the argument in Section 3, one can show that the eigenvectors in $V$ corresponding to nonzero eigenvalues are in a one-to-one correspondence with the eigenvectors of the sample covariance $(X-\bar{X})(X-\bar{X})^T$. As a matter of fact, as far as Euclidean distance is concerned, MDS and PCA produce the same results. However, the distances need not be based on Euclidean distances and can represent many types of dissimilarities between objects.

## 6 Isomap

Another recent approach to nonlinear dimensionality reduction is the Isomap algorithm. Isomap is a nonlinear generalization of classical MDS. The main contribution is to compute the MDS, not in the input space, but in the geodesic space of the manifold. The geodesic distances represent the shortest paths along the curved surface of the manifold measured as if the surface were flat. This can be approximated by a sequence of short steps between neighboring sample points. Isomap then applies MDS to the geodesic distances to find a low-dimensional mapping with similar pairwise distances.

Like LLE, the Isomap algorithm proceeds through three steps:

1. Find the neighbors of each data point in high-dimensional data space.
2. Compute the geodesic pairwise distances between all points.
3. Embed the data via MDS so as to preserve these distances.

Again like LLE, the first step can be performed by identifying the $K$ nearest neighbors, or by choosing all points within some fixed radius, $\epsilon$. These neighborhood relations are represented by a graph $G$ in which each data point is connected to its nearest neighbors, with edges of weight $d_X(i,j)$ between neighbors.

The geodesic distances $d_M(i,j)$ between all pairs of points on the manifold $M$ are then estimated in the second step. Isomap approximates $d_M(i,j)$ as the shortest path distance $d_G(i,j)$ in the graph $G$. This can be done in different ways including Dijkstra's algorithm [15] and Floyd's algorithm [16].

These algorithms final matrix of graph distances $D_G$ contains the shortest path distance between all pairs of points in $G$.

In its final step, Isomap applies classical MDS to $D_G$ to generate an embedding of the data in a $d$-dimensional Euclidean space $Y$.

The global minimum of the cost function is obtained by setting the coordinates of $y_i$ to the top $d$ eigenvectors of the inner-product matrix $B$ obtained from $D_G$

## 7 Future work

### 7.1 Motivation

Historically, two key approaches to discovering low-dimensional manifolds in high-dimensional data have been to find a mapping from the original space to a lower dimensional space that: (1) preserves pairwise distances (e.g. multidimensional scaling ); or (2) preserves mutual linear reconstruction ability (e.g. principle components analysis ). In each case, globally optimal solutions are linear manifolds. Interestingly, the more recent methods for manifold discovery, Isomap and LLE, are based on exactly these same two principles, with the generalization that the new methods only seek manifold descriptions that *locally* preserve distances and linear reconstructions. In this way, they avoid recovering global linear solutions.

There have been many new variants of these ideas [17, 18, 19, 20, 21], but, even though these techniques produce nonlinear manifolds in different ways, they are generally based on the core assumption that, in natural data,

1. Euclidean distances locally preserve geodesic distances on the manifold [12], or

2. data objects can be linearly reconstructed from other data objects nearby in Euclidean space. [10].

However, these core notions are neither universally applicable nor always effective. For example, in image data it is easy to appreciate the shortcomings of these ideas. For images, weighted linear combinations amount to an awkward transformation whereby source images have their brightness levels adjusted and then are summed directly on top of one another. This is often an unnatural way to capture the image transformations that manifolds are intended to characterize. Figure 3 shows that centered and normalized target images can be fairly well reconstructed from likewise aligned source

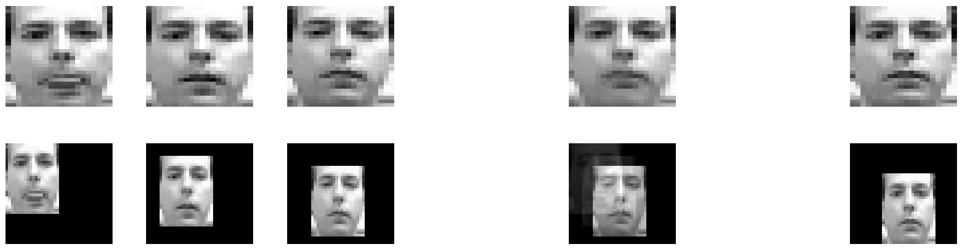

Figure 3: From the left: the first three frames are nearest neighbors, the middle frame is the reconstructed image and the right one is the target. The first row: reconstruction of centered images. Second row: reconstruction of the same images after a shift.

images, but that even a minor shift, rotation or rescaling will quickly limit the ability of this approach to reconstruct a target image. Similarly, measuring Euclidean distances between images can sometimes be a dubious practice since these distances do not always correspond to meaningful perceptual differences.

We propose to characterize manifolds locally by identifying local transformations that preserve invariants of the underlying data. That is, we attempt to characterize those transformations that cause points on the manifold to stay on the manifold. More specifically, we approximate the local tangent space around a data object by considering transformations of that object that cause it to stay on (or near) the manifold. Examining transformations that reconstruct neighboring objects from the input data set is one obvious way to achieve this.

### 7.1.1 Local image transformations

Although our approach is general, to illustrate the concepts concretely we will focus on the special case of modeling manifolds in natural image data. For images, it is easy to propose simple local transformations that capture natural invariants in image data better than simply averaging nearby images together. Consider a very simple class of transformations based on receptive fields of pixel neighborhoods: Given an $n1 \times n2$ image $x$, imagine transforming it into a nearby image $\tilde{x} = T(x, \theta)$, where for each pixel $\tilde{x}_i \in \tilde{x}$ we determine its value from corresponding nearby pixels in $x$. Specifically, we determine

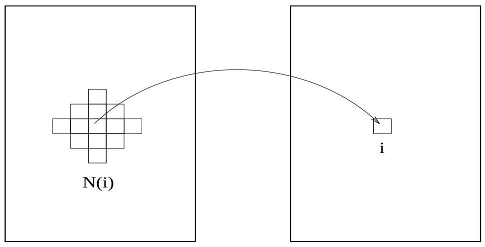

Figure 4: Illustration of local pixel transformation from left image to right

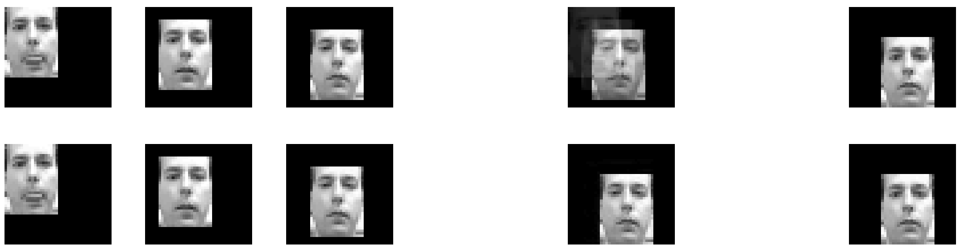

Figure 5: From the left: the first three frames are nearest neighbors, the middle frame is the reconstructed image and the right one is the target. The first row shows reconstruction by adjusting brightness level. The second row illustrates the reconstruction obtained by filter transformation.

$\tilde{x}_i$ according to

$$\tilde{x}_i = \theta^\top x_{N(i)} \tag{3}$$

where $N(i)$ denotes the set of neighboring pixels of pixel $x_i$. Thus $T(\cdot, \theta)$ defines a simple local filter passed over the image, parameterized by a single weight vector $\theta$, as shown in Figure 4.

Although this defines a limited class of image transformations, it obviously enhances the image modeling capabilities of weighted image combinations (which are only based on adjusting the brightness level of source images). Figure 5 shows that similar images can be much better reconstructed by simple filter transformations rather than merely adjusting brightness levels prior to summing. Here minor translations and appearance changes can be adequately modeled in circumstances where brightness changes fail.

## 7.2 Local tangent space modeling

The key to our proposal is to model the local tangent space around high-dimensional data points by a small number of transformations that locally preserve membership in the manifold. Thus, in our approach, a manifold is locally characterized by the invariants it preserves.

We model transformations over the data space by using an operator $T(x, \theta)$ which combines a data object $x$ and a parameter vector $\theta$ to produce a transformed object $\tilde{x} = T(x, \theta)$. In general, we will need to assume very little about this operator, but, by making some very simple (and fairly weak) assumptions about the nature of $T$, we will be able to formulate natural geometric properties that one can preserve in a dimensionality reducing embedding.

First, we assume that $T$ is a *bilinear* operator. That is, $T$ becomes a linear operator on each argument when the other argument is held fixed. Specifically,

$$\begin{aligned} T(ax_1 + bx_2, \theta) &= aT(x_1, \theta) + bT(x_2, \theta) \\ T(x, a\theta_1 + b\theta_2) &= aT(x, \theta_1) + bT(x, \theta_2) \end{aligned} \tag{4}$$

Second, we require the operator to have a local *origin* $\omega$ in the second argument that gives an identity map:

$$T(x, \omega) \;=\; x \text{ for all } x \tag{5}$$

With these properties, we can then naturally equate parameterized transformations with tangent vectors as follows. First note that $T(x, \theta) = x + T(x, \delta)$ for $\delta = \theta - \omega$, since by bilinearity we have

$$T(x, \theta) = T(x, \omega + \delta) = T(x, \omega) + T(x, \delta)$$

and also

$$T(x, \omega) = x$$

Thus, we can interpret every transformation of an object $x$ as a vector sum. That is, if $\tilde{x} = T(x, \theta)$ then the difference $\tilde{x} - x$ is just $T(x, \delta)$.

Now imagine transforming a source object $x_i$ to approximate a nearby target object $x_j$, where both reside on the manifold. The best approximation of $x_j$ by $x_i$ is given by

$$\tilde{x}_{ij} = T(x_i, \tilde{\theta}_{ij})$$

where

$$\tilde{\theta}_{ij} = \arg\min_{\theta} \|x_j - T(x_i, \theta)\|$$

If the approximation error is small, we can claim that the difference vector $\tilde{x}_{ij} - x_i = T(\tilde{\delta}_{ij})$, for $\tilde{\delta}_{ij} = \tilde{\theta}_{ij} - \omega$, is approximately tangent to the manifold at $x_i$. Consider the norm of the diffence vector:

$$\|x_i - \tilde{x}_{ij}\| \quad = \quad \|T(x, \tilde{\delta}_{ij})\| \quad = \quad \|\tilde{\delta}_{ij}\| \, \|T(x, \tilde{\eta}_{ij})\|$$

where $\tilde{\eta}_{ij} = \tilde{\delta}_{ij}/\|\tilde{\delta}_{ij}\|$. Here $T(x, \tilde{\eta}_{ij})$ gives the direction of the approximate tangent vector at $x_i$, and $\|\tilde{\delta}_{ij}\|$ gives the coefficient in direction $\tilde{\eta}_{ij}$. This says that $\tilde{x}_{ij}$ is the projection of $x_j$ onto the tangent plane centered at $x_i$, since $\tilde{x}_{ij} = x_i + \|\tilde{\delta}_{ij}\| T(x, \tilde{\eta}_{ij})$ is the best approximation of $x_j$ in the local tangent space of $x_i$.

Intuitively, when we embed $x_i$ and $\tilde{x}_{ij}$ in a lower dimensional space, say by a mapping $x_i \mapsto y_i$ and $\tilde{x}_{ij} \mapsto \tilde{y}_{ij}$, we would like to preserve the coefficient:

$$\|y_i - \tilde{y}_{ij}\| \quad \approx \quad \|\tilde{\delta}_{ij}\|$$

That is, in the lower-dimensional space, the vector $y_i - \tilde{y}_{ij}$ encodes the embedded direction of the transformation, $T(x_i, \tilde{\eta}_{ij})$, and the length $\|y_i - \tilde{y}_{ij}\|$ encodes the coefficient of the transformation, $\|\tilde{\delta}_{ij}\|$.

## 7.3 Local tangent space embedding algorithm

The local tangent space embedding (LTSE) algorithm proceeds through three steps:

In the first step, the neighbors of each data point $x_i$ are identified as follows:

Compute the best point-to-point approximations. The best approximation of $x_j$ by $x_i$ is given by

$$\tilde{x}_{ij} = T(x_i, \tilde{\theta}_{ij})$$

where

$$\tilde{\theta}_{ij} = \arg\min_{\theta} \|x_j - T(x_i, \theta)\|$$

The $K$ nearest neighbors of data point $x_i$ are the $K$ best approximations amongst all $\tilde{x}_{ij}$.

The second step of LTSE is to reconstruct each data point from its nearest neighbors. Consider a particular data point $x_i$ with $K$ nearest neighbors and reconstruction weights $w_j$. The reconstruction error can be written as:

$$\min_{W} \sum_{i} ||x_i - \tilde{x}_{N_{x_i}} w_i||^2$$

where $\tilde{x}_{N_{x_i}}$ denotes the $K$ nearest neighbors of $x_i$. This cost function should be minimized subject to two constraints: first, that each data point $x_i$ is reconstructed only from its neighbors ($W_{ij} = 0$ if $x_j$ is not among the nearest neighbors of $x_i$). Second, the rows of the weight matrix sum to one ($\sum_j W_{ij} = 1$). This is a constrained least squares problem and, similar to LLE, can be computed in closed form.

In the third step LTSE embeds both $X$ and $\tilde{X}$ (a $t \times t$ matrix containes all $\tilde{x}_{ij}$), into $Y$ and $\tilde{Y}$. Recall that LLE in its third step computes a low dimensional embedding $Y$ based on the reconstruction weights $W_{ij}$ of point $x_i$. LTSE should minimize a similar cost function while preserving $\tilde{\delta}_{ij}$, the coefficient of the transformation in the original space. Actually, the cost function of LTSE in this step consists of two parts. The first part ensures mutual linear reconstruction ability (similar to LLE) and the second part preserves pairwise distances (similar to MDS).

$$\min_{Y,\tilde{Y}} ||Y - \tilde{Y}W||^2 + || - \frac{1}{2} H D^{(\delta)} H - [Y, \tilde{Y}]^T [Y, \tilde{Y}]||^2$$

where $D^{(\delta)}$ is a distance matrix containing all transformation coefficients $\delta_{ij}$ and $[Y, \tilde{Y}]$ denotes a $d \times (K+1)t$ matrix containing the $d \times t$ matrix $Y$ and the $d \times K * t$ matrix $\tilde{Y}$.

The individual parts of this cost function are classical problems. We have seen the solution of the first part in LLE and the second part has been addressed by MDS. Omitting the details, it can be shown that, even with two parts, the problem is still tractable and has a closed form solution. This solution can be represented as eigenvectors of $-\frac{1}{2} H D^{(\delta)} H$, together with an adjustment from the first part of the cost function.

## 7.4 Experimental results

We present preliminary experimental results on natural image data. Here we use the transformation operator on images (3) that was described in Section 7.1.1. First, we need to verify that the bilinearity property (4) and

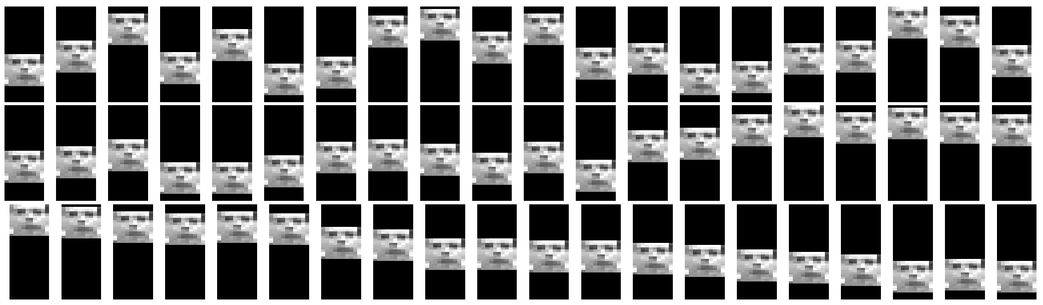

Figure 6: Top: Original data. Middle: 1-dimensional manifold discovered by LLE. Bottom: 1-dimensional manifold discovered by LTSE

origin property (5) are both satisfied. Bilinearity follows trivially from the linearity of (3). Here, the identity transformation $T(x, \omega) = x$ is given by parameters $\omega$ that are zero for all neighboring pixels, excepting the center pixel aligned with the target pixel, where the parameter is 1.

Our first experiment is on a single face image We have conducted experiments on various sets of face images. Although the data is complex, many of these data sets demonstrate manifold behavior over a small number of degrees of freedom.

that has been translated vertically. Figure 6 shows the result of running LLE and Local Tangent Space Embedding (LTSE) on the original data shown at the top. This figure shows that the 1-dimensional manifold discovered by LLE is inferior to that discovered by LTSE, which had no problem tracking the vertical shift in the image set. We then repeated this experiment with a larger number of face images appearing at random translations. Figure 7 again shows improved performance for LTSE over LLE. Here, a two-dimensional manifold was discovered by each technique. LTSE discovered vertical translation as its first dimension, which LLE failed to capture in this case.

Finally, we conducted an experiment on a database of rotating face images. Figure 8 shows the two-dimensional manifold discovered by LLE, whereas Figure 9 shows the two-dimensional manifold recovered by LTSE. In both cases, the first dimension (top) captured the rotation angle of the images, although once again LLE's result is not as good as LTSE's. Interestingly, LTSE (and to a lesser extent LLE) learned to distinguish frontal from profile views in its second dimension.

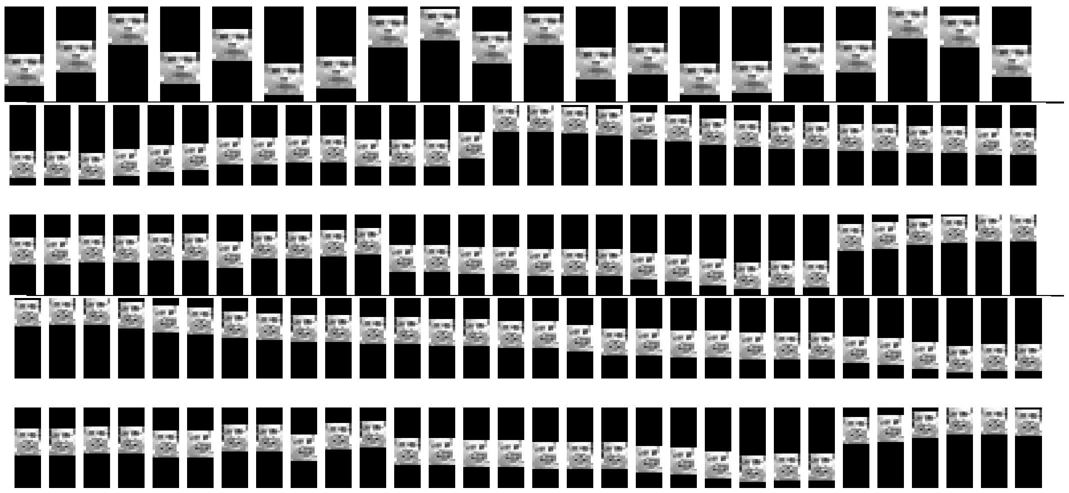

Figure 7: Top: Original data. 2nd and 3rd rows: 2-dimensional manifold discovered by LLE. 4th and 5th rows: 2-dimensional manifold discovered by LTSE

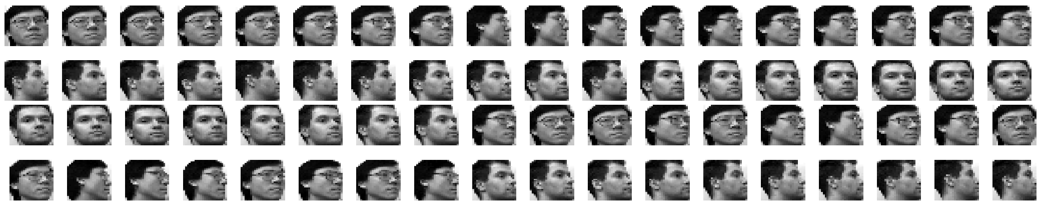

Figure 8: Two-dimensional manifold discovered by LLE

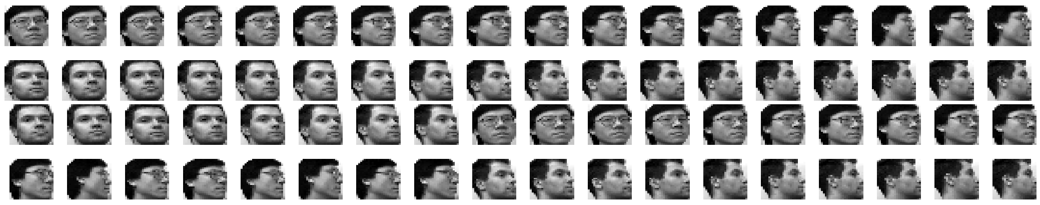

Figure 9: Two-dimensional manifold discovered by LTSE. Note: first dimension captures rotation, whereas second captures frontal views versus side views

## References

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

[2] Trevor F. Cox and Michael A. A. Cox. *Multidimensional Scaling.* Chapman and Hall/CRC, Boca Raton, second edition, 2001.

[3] B.J. Frey. *Graphical models for machine learning and digital communication.* The MIT Press, Cambridge, Mass, 1998.

[4] J. Friedman T. Hastie, R. Tibshirani. *The elements of statistical learning.* Springer, New York, 2002.

[5] A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2:94–128, 1999.

[6] V. Cherkassky and F. Mulier. *Learning from data.* John Wilew and Sons, INC., New York, 1998.

[7] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de–noising in feature spaces. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11.* MIT Press, 1999.

[8] B. Scholkopf and A. smola. *Learning with Kernels.* The MIT Press, Cambridge, Massachusetts, 2002.

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

[10] Lawrence Saul and Sam Roweis. Think globally, fit locally: Unsupervised learning of nonlinear manifolds. *Journal of Machine Learning Research*, 2003.

[11] Joshua B. Tenenbaum. Mapping a manifold of perceptual observations. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *NIPS*, volume 10. The MIT Press, 1998.

[12] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[13] H. Hotelling. Analysis of a complex of statistical variables into components. *Journal of Educational Psychology*, 24:417–441, 1933.

[14] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh and Dublin Philosophical Magazine and Journal of Science*, Sixth Series 2:559–572, 1901.

[15] R.L. Rivest T.H. Cormen, C.E. Leiserson and C.Stein. *Introduction to algorithms*. The MIT Press, Cambridge, Massachusetts, 2001.

[16] A. Gupta V. Kumar, A. Grama and G. Karypis. *Introduction to parallel computing*. Benjamin, Cummings, 1994.

[17] Guy Lebanon. Learning Riemannian metrics. In *UAI*, 2003.

[18] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

[19] Yee Whye Teh and Sam Roweis. Automatic alignment of local representations. In *NIPS*, 2002.

[20] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *NIPS*, 2002.

[21] Hongyuan Zha and Zhenyue Zhang. Isometric embedding and continuum isomap. In *Twentieth International Conference on Machine Learning ICML*, 2003.