Reducing Feature-Space Dimensionality
When Data Resides on a Manifold in a Higher Dimensional Euclidean Space

Avinash Kak
Purdue University

April 7, 2013
1:21pm

An RVL Tutorial Presentation

Originally presented in Fall 2008
Minor formatting changes in April 2013

©2013 Avinash Kak, Purdue University
Prologue

Table of Contents

Part 1: (Slides 4 — 17)
  Review of Linear/Greedy Methods

Part 2: (Slides 18 — 24)
  Feature Distributions on Nonlinear Mani-
  f olds

Part 3: (Slides 25 — 43)
  ISOMAP

Part 4: (Slides 44 — 59)
  Locally Linear Embeddings (LLE)
PART 1:

A Brief Review of Linear/Greedy Methods for Reducing the Dimensionality of Feature Spaces

Slides 4 — 17
1.1: Dimensionality Reduction by Linear and Greedy Methods

Traditionally, feature space dimensionality has been reduced by

- Linear methods, such as PCA, LDA, and variations thereof.

- Greedy algorithms that examine each feature, one at a time, and, at each step of the algorithm, adds the best feature to those already retained. A feature is best if it minimizes some cost function.
1.2: PCA, LDA, etc., and k-NN for Image Classification

• Ordinarily, given \( N \) labeled training images

\[ x_i \quad i = 0, 1, 2, \ldots, N - 1 \]

we represent each image as a point in an \( D \) dimensional vector space where \( D \) is the total number of pixels in each image. Each vector dimension stands for the pixel brightness at a particular pixel.

• The PCA approach to dimensionality reduction and image classification consists of first normalizing the images by requiring that \( \bar{x}_i^T \bar{x}_i = 1 \), and then calculating the covariance \( \mathbf{C} \) of the images by
\[
C = \frac{1}{N} \sum_{i=0}^{N-1} \left\{ (x_i - m)(x_i - m)^T \right\}
\]

where the mean image vector is given by

\[
\vec{m} = \frac{1}{N} \sum_{i=0}^{N-1} \vec{x}_i
\]

Subsequently, we carry out an eigendecomposition of the covariance matrix and retain a small number, \(d\), of the eigenvectors. Let’s denote our orthogonal PCA feature set by \(W_d\):

\[
W_d = [\vec{w}_0, \vec{w}_1, ..., \vec{w}_{d-1}]
\]

where \(\vec{w}_i\) denotes the \(i^{th}\) eigenvector of the covariance matrix \(C\).
• We then represent each image in this small $d$-dimensional space by calculating

$$\tilde{y} = W_d^T(x - \bar{m})$$

• To classify an unknown image, a commonly used method consists of first locating all the $N$ training images as points in the $d$ dimensional space and then giving the unknown image the same label as that of its nearest neighbor from the training set.

• Obviously, we can think of many variations on the approach outlined so far. We could, for example, used $k$-NN instead of 1-NN. We could use LDA instead of PCA, etc.
Methods such as PCA, LDA, etc., constitute **linear methods for reducing the dimensionality** of a feature space, linear in the sense that these methods do not involve calculating the minimum of a cost function. [The main focus of this talk is on nonlinear methods for dimensionality reduction. The nonlinear methods may or may not involve finding a local/global minimum for a cost function. The advantage of the nonlinear methods is that they will be able to determine the intrinsic low dimensionality of the data when it resides on some simple surface in an otherwise high-dimensional space.]
1.3: Greedy Methods for Dimensionality Reduction of Feature Spaces

- These are usually iterative methods that start with one best single feature and then add one best feature at a time to those previously retained until you have the desired number of best features.

- At the outset, a single feature is considered best if it minimizes the entropy of all the class distributions projected on to that feature.

- Subsequently, after we have retained a set of features, a new feature from those remaining is considered best if it minimizes the entropy of the class distributions when projected into the subspace formed by the addition of the new feature.
• What I have described above is called the **forward selection method**.

• Along the same lines, one can also devise a **backward elimination method** starts from the full features space and eliminates one feature at a time using entropy-based cost functions.

• Greedy methods are good only when you know that a subset of the input-space features contain sufficient discriminatory power. The goal then becomes to find the subset.

• In general, approaches based on PCA, LDA, etc., work better because now you can look for arbitrary directions in the feature space to find the features that would work best in some low-dimensional space.
1.4: Limitations to Dimensionality Reduction with PCA, LDA, etc.

- Pattern classification of the sort previously mentioned requires that we define a metric in the feature space. A commonly used metric is the Euclidean metric, although, for the sake of computational efficiency, we may use variations on the Euclidean metric.

- But a small Euclidean distance implying two similar images makes sense only when the distributions in the feature space form amorphous clouds. A common example would be the Gaussian distribution or variations thereof.

- However, when the points in a feature space form highly organized shapes, a small Euclidean distance between two points may not imply pattern similarity.
• Consider the two-pixel images formed as shown in the next figure. We will assume that the object surface is Lambertian and that the object is lighted with focussed illumination as shown.

• We will record a sequence of images as the object surface is rotated vis-a-vis the illumination. Our purpose is to collect training images that we may use later for classifying an unknown pose of the object.

• We will assume that the pixel $x_1$ in each image is roughly a quarter of the width from the left edge of each image and the pixel $x_2$ about a quarter of the width from the right edge.

• We will further assume that the sequence of images is taken with the object rotated through all of $360^\circ$ around the axis shown.
Because of Lambertian reflection, the two pixels in image indexed $i$ will be roughly as

\[
(x_1)_i = A \cos \theta_i \\
(x_2)_i = B \cos(\theta_i + 45^\circ)
\]

where $\theta_i$ is the angle between the surface normal at the object point that is imaged at pixel $x_1$ and the illumination vector and where we have assumed that the two panels on the object surface are at a 45 deg angle.
• So as the object is rotated, the image point in the 2D feature space formed by the pixels \((x_1, x_2)\) will travel a trajectory as shown in the next figure. Note that the beginning and the end points of the curve in the feature space are not shown as being the same because we may not expect the reflectance properties of the “back” of the object to be the same as those of the “front.”
• The important point to note is that when the data points in a feature space are as structured as shown in the figure on the previous slide, we cannot use Euclidean sort of a metric in that space as a measure of similarity. Two points, such as A and B marked in the figure, may have short Euclidean distance between them, yet they may correspond to patterns that are far apart from the standpoint of similarity.

• The situation depicted in the figure on the previous slide can be described by saying that the patterns form a 1D manifold in an otherwise 2D feature space. That is, the patterns occupy a space that has, locally speaking, only 1 DOF.
• It would obviously be an error to use linear methods like those based on PCA, LDA, etc., for discrimination between image classes when such class distributions occupy spaces that are more accurately thought of as manifolds.

• In other words, when class distributions do not form volumetric distributions, but instead when they populate structured surfaces, one should not use linear methods like PCA, LDA, etc.
PART 2: Feature Distributions on Nonlinear Manifolds

Slides 18 through 24
2.1: Feature Distributions On Nonlinear Manifolds

- Let’s now add one more motion to the object in the imaging setup shown on Slide 13. In addition to turning the object around its long axis, we will also rock it up and down at its “back” edge while not disturbing the “front” edge. The second motion is depicted in the next figure.

- Let’s also now sample each image at three pixels, as shown in the next figure. Note again, the pixels do not correspond to fixed points on the object surface. Rather, they are three pixels of certain prespecified coordinates in the images. So each image will be now be represented by the following 3-vector:
\[ \vec{x}_i = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \]
• We will assume that the training data is generated by random rotations and random rocking motions of the object between successive image captures by the camera.

• Each training image will now be one point is a 3-dimensional space. Since the brightness values at the pixels $x_1$ and $x_3$ will always be nearly the same, we will see a band-like spread in the $(x_1, x_3)$ plane.

• The training images generated will now form a 2D manifold in the 3D $(x_1, x_2, x_3)$ space as shown in the figure below.
Another example of the data points being distributed on a manifold is shown in the next figure. This figure, generated synthetically, is from the paper by Tenenbaum et al. This figure represents three dimensional data that is sampled from a two-dimensional manifold. [A manifold’s dimensionality is determined by asking the question: How many independent basis vectors do I need to represent a point inside a local neighborhood on the surface of the manifold?]
• To underscore the fact that using straight-line Euclidean distance metric makes no sense when data resides on a manifold, the distribution presented in the previous figure shows two points that are connected by a straight-line distance and a geodesic. The straight-line distance could lead to the wrong conclusion that the points represent similar patterns, but the geodesic distance tells us that those two points correspond to two very different patterns.

• In general, when data resides on a manifold in an otherwise higher dimensional feature space, we want to compare pattern similarity and establish neighborhoods by measuring geodesic distances between the points.
• Again, a manifold a lower-dimensional surface in a higher-dimensional space. And, the geodesic distance between two points on a manifold is the shortest distance between the two points on the manifold.

• As you know, the shortest distance between any two points on the surface of the earth is along the great circle that passes through those points. So the geodesic distances on the earth are along the great circles.
PART 3: Dimensionality Reduction with ISOMAP

Slides 25 through 43
3.1: Calculating Manifold-based Geodesic Distances from Input-Space Distances

- So we are confronted with the following problem: How to calculate the geodesic distances between the image points in a feature space?

- Theoretically, the problem can be stated in the following manner:

- Let $M$ be a $d$-dimensional manifold in the Euclidean space $\mathcal{R}^D$. Let's now define a distance metric between any two points $\vec{x}$ and $\vec{y}$ on the manifold by

$$d_M(\vec{x}, \vec{y}) = \inf_{\gamma} \{ \text{length}(\gamma) \}$$
where $\gamma$ varies over the set of arcs that connect $\vec{x}$ and $\vec{y}$ on the manifold. To refresh your memory, infimum of a set means to return an element that stands for the greatest lower bound vis-a-vis all the elements in the set. In our case, the set consists of the length values associated with all the arcs that connect $\vec{x}$ and $\vec{y}$. The infimum returns the smallest of these length values.

- Our goal is to estimate $d_M(\vec{x}, \vec{y})$ given only the set of points $\{\vec{x}_i\} \subset \mathcal{R}^D$. We obviously have the ability to compute the pairwise Euclidean distances $\| \vec{x}, \vec{y} \|$ in $\mathcal{R}^D$.

- We can use the fact that when the data points are very close together according to, say, the Euclidean metric, they are also likely to be close together on the manifold (if one is present in the feature space).
• It is only the medium to large Euclidean distances that cannot be trusted when the data points reside on a manifold.

• So we can make a graph of all of the points in a feature space in which two points will be directly connected only when the Euclidean distance between them is very small.

• To capture this intuition, we define a graph $G = \{V, E\}$ where the set $V$ is the same as the set of data points $\{\vec{x}_i\}$ and in which $\{\vec{x}_i, \vec{x}_j\} \in E$ provided $\| \vec{x}_i, \vec{x}_j \|$ is below some threshold.
• We next define the following two metrics on the set of measured data points. For every $\vec{x}$ and $\vec{y}$ in the set $\{\vec{x}_i\}$, we define:

$$d_G(\vec{x}, \vec{y}) = \min_P (\| x_0 - x_1 \| + \ldots + \| x_{p-1} - x_p \|)$$
$$d_S(\vec{x}, \vec{y}) = \min_P (d_M(x_0, x_1) + \ldots + d_M(x_{p-1}, x_p))$$

where the path $P = (\vec{x}_0 = \vec{x}, \vec{x}_1, \vec{x}_2, \ldots \vec{x}_p = \vec{y})$ varies over all the paths along the edges of the graph $G$.

• As previously mentioned, our real goal is to estimate $d_M(\vec{x}, \vec{y})$. We want to be able to show that $d_G \approx d_M$. We will establish this approximation by first demonstrating that $d_M \approx d_S$ and then that $d_S \approx d_G$. 
• To establish these approximations, we will use the following inequalities:

\[
\begin{align*}
d_M(\vec{x}, \vec{y}) & \leq d_S(\vec{x}, \vec{y}) \\
d_G(\vec{x}, \vec{y}) & \leq d_S(\vec{x}, \vec{y})
\end{align*}
\]

The first follows from the triangle inequality for the metric \(d_M\). The second inequality holds because the the Euclidean distances \(\|\vec{x}_i - \vec{x}_{i+1}\|\) are smaller than the arc-length distances \(d_M(\vec{x}_i, \vec{x}_{i+1})\).

• The proof of the approximation \(d_M \approx d_G\) is based on demonstrating that \(d_S\) is not too much larger than \(d_M\) and that \(d_G\) is not too much smaller than \(d_S\).
3.2: The ISOMAP Algorithm for Estimating the Geodesic Distances

- The ISOMAP algorithm can be used to estimate the geodesic distances $d_M(\vec{x}, \vec{y})$ on a lower-dimensional manifold that is inside a higher-dimensional Euclidean input space $\mathcal{R}^D$.

- ISOMAP consists of the following steps:

  **Construct Neighborhood Graph:** Define a graph $G$ over all the set $\{\vec{x}_i\}$ of all data points in the underlying $D$-dimensional features space $\mathcal{R}^D$ by connecting the points $\vec{x}$ and $\vec{y}$ if the Euclidean distance $\| \vec{x} - \vec{y} \|$ is smaller than a pre-specified $\epsilon$ (for $\epsilon$-ISOMAP). In graph $G$, set edge lengths equal to $\| \vec{x} - \vec{y} \|$. 
Compute Shortest Paths: Use Floyd’s algorithm for computing the shortest pairwise distances in the graph $G$:

- Initialize $d_G(x, y) = \| x - y \|$ if $\{x, y\}$ is an edge in graph $G$. Otherwise set $d_G(x, y) = \infty$.

- Next, for each node $z \in \{x_i\}$, replace all entries $d_G(x, y)$ by $\min \{d_G(x, y), d_G(x, z) + d_G(z, y)\}$.

- The matrix of final values $D_G = \{d_G(x, y)\}$ will contain the shortest path distances between all pairs of nodes in $G$.

Construct d-dimensional embedding: Now use classical MDS (Multidimensional Scaling) to the matrix of graph distances $D_G$ and thus construct an embedding in a d-dimensional Euclidean space $Y$ that best preserves the manifold’s estimated intrinsic geometry.
3.3: Using MDS along with $D_M$ Distances to Construct Lower-Dimensional Representation for the Data

- MDS finds a set of vectors that span a lower $d$-dimensional space such that the matrix of pairwise Euclidean distances between them in this new space corresponds as closely as possible to the similarities expressed by the manifold distances $d_M(x, y)$.

- Let this new $d$-dimensional space be represented by $\mathcal{R}^d$. Our goal is to map the dataset $\{x_i\}$ from the input Euclidean space $\mathcal{R}^D$ into a new Euclidean space $\mathcal{R}^d$. 
• For convenience of notation, let $\vec{x}$ and $\vec{y}$ represent two arbitrary points in $\mathbb{R}^D$ and \textbf{also} the corresponding points in the target space $\mathbb{R}^d$.

• Our goal is to find the $d$ basis vector for $\mathbb{R}^d$ such that following cost function is minimized:

$$E = \| D_M - D_{\mathbb{R}^d} \|_F$$

where $D_{\mathbb{R}^d}(\vec{x}, \vec{y})$ is the Euclidean distance between mapped points $\vec{x}$ and $\vec{y}$ and where $\| \cdot \|_F$ is the Frobenius norm of a matrix. Recall that for $N$ input data points in $\mathbb{R}^D$, both $D_M$ and $D_{\mathbb{R}^d}$ will be $N \times N$. [For a matrix $A$, its Frobenius norm is given by $\| A \|_F = \sqrt{\sum_{i,j} |A_{ij}|^2}$]
• In MDS algorithms, it is more common to minimize the normalized

\[ E = \frac{\| D_M - D_{R^d} \|_F}{\| D_M \|_F} \]

Quantitative psychologists refer to this normalized form as stress.

• A classical example of MDS is to start with a matrix of pairwise distances between a set of cities and to then ask the computer to situate the cities as points on a plane so that visual placement of the cities would be in proportion to the inter-city distances.
• For algebraic minimization of the cost function, the cost function is expressed as

\[ E = \left\| \tau(D_M) - \tau(D_{R^d}) \right\|_F \]

where the \( \tau \) operator converts the distances to inner products.

• It can be shown that the solution to the above minimization consists of the using the largest \( d \) eigenvectors of the sampled \( \tau(D_M) \) (or, equivalently, the estimated approximation \( \tau(D_G) \)) as the basis vectors for the reduced dimensionality representation of \( R^d \).

• The intrinsic dimensionality of a feature space is found by creating the reduced dimensionality mappings to \( R^d \) for different values of \( d \) and retaining that value for \( d \) for which the residual \( E \) more or less the same as \( d \) is increased further.
• When ISOMAP is applied to the synthetic Swiss roll data shown in the figure on Slide 21, we get the plot shown by the filled circles in the upper right-hand plate of the next figure that is also from the publication by Tenenbaum et al. As you can see, when $d = 2$, $E$ goes to zero, as it should. The other curve in the same plate is for PCA.

• For curiosity’s sake, the graph constructed by ISOMAP from the Swiss roll data is shown in the figure on the next slide.
• In summary, ISOMAP creates a low-dimensional Euclidean representation from an input feature space in which the data resides on a manifold surface which could be a folded or a twisted surface.

• The other plots in the figure on the previous slide are for the other datasets for which Tenenbaum et al. have demonstrated the power of the ISOMAP algorithm for dimensionality reduction.
Tenenbaum et al. also experimented with a dataset consisting of $64 \times 64$ images of a human head (a statue head). The images were recorded with three parameters, left-to-right orientation of the head, top-to-bottom orientation of the head, and by changing the direction of illumination from left to right. Some images from the dataset are shown in the figure below. One can claim that even when you represent the images by vectors in $\mathbb{R}^{4096}$, the dataset has only three DOF intrinsically. This is borne out by the output of ISOMAP shown in the upper-left of the plots on Slide 36.
Another experiment by Tenenbaum et al. involved a dataset consisting of 64 × 64 images of a human hand with two “intrinsic” degrees of freedom: one created by the rotation of the wrist and other created by the unfolding of the figures. The input space in this case is again \( \mathbb{R}^{4096} \). Some of the images in the dataset are shown in the figure below.

The lower-left plate in the plots on Slide 36 corresponds to this dataset.
Another experiment carried out by Tenenbaum et al. used 1000 images of handwritten 2’s, as shown in the figure below. Two most significant features of how most humans write 2’s are referred to as the “bottom loop articulation” and the “top arch articulation”. The authors say they did not expect a constant low-dimensionality to hold over the entire dataset.
3.4: Computational Issues Related to ISOMAP

• ISOMAP calculation is nonlinear because it requires minimization of a cost function — an obvious disadvantage vis-a-vis linear methods like PCA, LDA, etc., that are simple to implement.

• In general, it would require much trial and error to determine the best thresholds to use on the pairwise distances $D(\vec{x}, \vec{y})$ in the input space. Recall that when we construct a graph from the data points, we consider two nodes directly connected when the Euclidean distance between them is below a threshold.
• **ISOMAP** assumes that the same distance threshold would apply everywhere in the underlying high-dimensional input space $\mathcal{R}^D$.

• **ISOMAP** also assumes implicitly that the same manifold would all of the input data.
PART 4: Dimensionality Reduction with the LLE Algorithm

Slides 44 through 59
4.1: Dimensionality Reduction by Locally Linear Embedding (LLE)

- This is also a nonlinear approach, but does not require a global minimization of a cost function.

- LLE is based on the following two notions:
  
  - When data resides on a manifold, any single data vector can be expressed as a linear combination of its $K$ closest neighbors using a coefficient matrix whose rank is less than the dimensionality of the input space $\mathcal{R}^D$.

  - The reconstruction coefficients discovered in expressing a data point in terms of its neighbors on the manifold can then be used directly to construct a low-dimensional Euclidean representation of the original input data.
4.2: Estimating the Weights for Locally Linear Embedding of the Input-Space Data Points

- Let $\bar{x}_i$ be the $i^{th}$ data point in the input space $\mathcal{R}^D$ and let $\{\bar{x}_j|j = 1 \ldots K\}$ be its $K$ closest neighbors according to the Euclidean metric for $\mathcal{R}^D$, as depicted in the figure below.
• The fact that a data point can be expressed as a linear combination of its $K$ closest neighbors can be expressed as

$$\tilde{x}_i = \sum_j w_{ij} \tilde{x}_j$$

The equality in the above relationship is predicated on the assumption that the $K$ closest data points are sufficiently linearly independent in a coordinate frame that is local to the manifold at $x_i$.

• In order to discover the nature of linear dependency between the data point $\tilde{x}_i$ on the manifold and its $K$ closest neighbors, it would be more sensible to minimize the following cost function:

$$\mathcal{E}_i = |\tilde{x}_i - \sum_j w_{ij} \tilde{x}_j|^2$$
• Since we will be performing the same calculations each input data point \( \bar{x}_i \), in the rest of the discussion we will drop the suffix \( i \) and let \( \bar{x} \) stand for any arbitrary data point on the manifold. So for a given \( \bar{x} \), we want to find the best weight vector \( \vec{w} = (w_1, \ldots, w_K) \) that would minimize

\[
\mathcal{E}(\vec{w}) = |\bar{x} - \sum_j w_j \bar{x}_j|^2
\]

• In the LLE algorithm, the weights \( \vec{w} \) are found subject to the condition that \( \sum_j w_j = 1 \). This constraint — a **sum-to-one** constraint — is merely a normalization constraint that expresses the fact that we want the proportions contributed by each of the \( K \) neighbors to any given data point to add up to one.
• We now re-express the cost function at a given input point $\vec{x}$ as

$$\mathcal{E}(\vec{w}) = |\vec{x} - \sum_j w_j \vec{x}_j|^2$$

$$= |\sum_j w_j (\vec{x} - \vec{x}_j)|^2$$

where the second equality follows from the sum-to-unity constraint on the weights $w_j$ at all input data points.

• Let’s now define a **local** covariance at the data point $\vec{x}$ by

$$C_{jk} = (\vec{x} - \vec{x}_j)^T(\vec{x} - \vec{x}_k)$$

The local covariance matrix $C$ is obviously an $K \times K$ matrix whose $(j, k)^{th}$ element is given by **inner** product of the Euclidean distance between $\vec{x}$ and $\vec{x}_j$, on the one hand, and the distance $\vec{x}$ and $\vec{x}_k$, on the other.
• In terms of the local covariance matrix, we can write for the cost function at a given input data point $\vec{x}$:

$$\mathcal{E} = \sum_{j,k} w_j w_k C_{jk}$$

• Minimization of the above cost function subject to the constraint $\sum_j w_j = 1$ using the method of Lagrange multipliers gives us the following solution for the coefficients $w_j$ at a given input data point:

$$w_j = \frac{\sum_k C^{-1}_{jk}}{\sum_i \sum_j C^{-1}_{ij}}$$
4.3: Invariant Properties of the Reconstruction Weights

- The reconstruction weights, as represented by the matrix $W$ of the coefficients at each input data point $\vec{x}$, are invariant to the rotations of the input space. This follows from the fact that the scalar products that form the elements are of the local covariance matrix involve products of Euclidean distances in a small neighborhood around each data point. Those distances are not altered by rotating the entire manifold.

- The reconstruction weights are also invariant to the translations of the input space. This is a consequence of the sum-to-one constraint on the weights.
• We can therefore say “that the reconstruction weights characterize the intrinsic geometrical properties in each neighborhood, as opposed to properties that depend on a particular frame of reference.”
4.4: Constructing a Low-Dimensional Representation from the Reconstruction Weights

- The low-dimensional reconstruction is based on the idea we should use the same reconstruction weights that we calculated on the manifold — that is, the weight represented by the vector $\mathbf{w}$ at each data point — to reconstruct the input data point in a low dimensional space.

- Let the low-dimensional representation of each input data point $\mathbf{x}_i$ be $\mathbf{y}_i$. LLE is founded on the notion that the previously computed reconstruction weights will suffice for constructing a representation of each $\mathbf{y}_i$ in terms of its $K$ nearest neighbors.
• That is, we place our faith in the following equality in the to-be-constructed low-dimensional space:

\[ \vec{y}_i = \sum_j w_{ij} \vec{y}_j \]

But, of course, so far we do not know what these vectors \( \vec{y}_i \) are. So far we only know how they should be related.

• We now state the following mathematical problem: Considering together all the input data points, find the best \( d \)-dimensional vectors \( \vec{y}_i \) for which the following global cost function is minimized

\[ \Phi = \sum_i |\vec{y}_i - \sum_j w_{ij} \vec{y}_j|^2 \]

If we assume that we have a total of \( N \) input-space data points, we need to find \( N \) low-dimensional vector \( \vec{y}_i \) by solving the above minimization.
• The form shown above can be re-expressed as

\[ \Phi = \sum_i \sum_j M_{ij} \vec{y}_i^T \vec{y}_j \]

where

\[ M_{ij} = \delta_{ij} - w_{ij} - w_{ji} + \sum_k w_{ki} w_{kj} \]

where \( \delta_{ij} \) is 1 when \( i = j \) and 0 otherwise.

• As it is, the above minimization is ill-posed unless the following two constraints are also used.

• We eliminate one degree of freedom in specifying the origin of the low-dimensional space by specifying that all of the new \( N \) vectors \( \vec{y}_i \) taken together be centered at the origin:

\[ \sum_i \vec{y}_i = 0 \]
• We require that the embedding vectors have unit variance with outer products that satisfy:

\[
\frac{1}{N} \sum_i \vec{y}_i \vec{y}_i^T = I
\]

where \( I \) is a \( d \times d \) identity matrix.

• The minimization problem is solved by computing the bottom \( d+1 \) eigenvectors of the \( M \) matrix and then discarding the last. The remaining \( d \) eigenvectors are the solution we are looking for. Each eigenvector has \( N \) components. When we arrange these eigenvectors in the form of a \( d \times N \) matrix, the column vectors of the matrix are the \( N \) vectors \( \vec{y}_i \) we are looking for. Recall \( N \) is the total number of input data points.
4.5: Some Examples of Dimensionality Reduction with LLE

In the example shown in the figure below, the input data consists of 600 samples taken from an Swiss roll manifold. The calculations for mapping the input data to a two-dimensional space was carried out with $K = 12$. That is, the local intrinsic geometry at each data point was calculated from the 12 nearest neighbors.
• The next example was constructed from 2000 images \((N = 2000)\) of the same face, with each image represented by a \(20 \times 28\) array of pixels. Therefore, the dimensionality of the input space is 560. The parameter \(K\) was again set to 12 for determining the intrinsic geometry at each 560 dimensional data point. The figure shows a 2-dimensional embeddings constructed from the data. Representative faces are shown next to circled points. The faces at the bottom correspond to the solid trajectory in the upper right portion of the figure.
Acknowledgements

The figures reproduced from the publication by Tenenbaum, de Silva, and Lengford are with permission from Josh Tenenbaum. Similarly, the figures reproduced from the publication by Roweis and Saul are with permission from Sam Roweis.