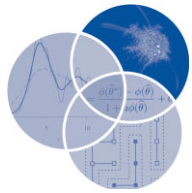
Overview



Introduction to manifold learning

Alan Julian Izenman*

A popular research area today in statistics and machine learning is that of manifold learning, which is related to the algorithmic techniques of dimensionality reduction. Manifold learning can be divided into linear and nonlinear methods. Linear methods, which have long been part of the statistician's toolbox for analyzing multivariate data, include principal component analysis (PCA) and multidimensional scaling (MDS). Recently, there has been a flurry of research activity on nonlinear manifold learning, which includes Isomap, local linear embedding, Laplacian eigenmaps, Hessian eigenmaps, and diffusion maps. Some of these techniques are nonlinear generalizations of the linear methods. The algorithmic process of most of these techniques consists of three steps: a nearest-neighbor search, a definition of distances or affinities between points (a key ingredient for the success of these methods), and an eigenproblem for embedding high-dimensional points into a lower dimensional space. This article gives us a brief survey of these new methods and indicates their strengths and weaknesses.

How to cite this article:
WIREs Comput Stat 2012, 4:439–446. doi: 10.1002/wics.1222

Keywords: dimensionality reduction; Isomap; local linear embedding; Laplacian eigenmaps; diffusion maps

INTRODUCTION

Manifold learning became a new topic of research in the year 2000 when two innovative and novel articles,[1,2] appeared in the same issue of *Science*. These articles addressed the problem of how to recover a nonlinear low-dimensional manifold from data located on that manifold, which is embedded within a higher dimensional ambient space. For example, in computer vision and image analysis, an object is viewed from multiple viewing angles (so that noise is not an issue), and so the observed data lie exactly on a low-dimensional manifold. To resolve this manifold learning problem, the techniques of Isomap and locally linear embedding (LLE) were proposed, and they both had an enormous impact on the development of this topic. The methodologies, algorithms, and solutions given by those articles were different, but both approaches boiled down to extracting the top or bottom few eigenvalues and associated eigenvectors of certain large and sparse matrices. These algorithms were the first attempts at nonlinear manifold learning, using spectral embedding methods.[3] Following the appearance of these two articles, a number of other approaches for solving the same problem were published in a variety of journals and books. Thus, we saw the introduction of Laplacian eigenmaps,[4] Hessian eigenmaps,[5] manifold charting,[6] local tangent alignment,[7] diffusion maps,[8,9] and several other techniques. More likely, however, real data rarely lie exactly on any type of manifold; so, these algorithms have been applied primarily to simulated data that are randomly sprinkled on manifolds having specific quirks (e.g., *S*-curved manifold, Swiss-roll manifold, open box, torus, sphere, fishbowl), and these manifolds are studied in order to expose any weaknesses of these algorithms. Research interest in nonlinear manifold learning has prompted the recent appearance of several books and collections of articles on the topic (see Refs 10–12).

Both *Science* articles addressed the manifold learning problem as one of 'dimensionality reduction,' which has long been an important topic for statisticians.[12,13] The statistical community developed the theory and practice of various methods for

*Correspondence to: ALAN@temple.edu
Department of Statistics, Temple University, Philadelphia, PA, USA

linear dimensionality reduction. The two most widely used linear techniques for dimensionality reduction are principal components analysis (PCA) and multidimensional scaling (MDS), both of which have solutions that are based upon the top few eigenvalues and associated eigenvectors of certain matrices. Recent applications of PCA include problems in computer vision, facial and object recognition, image compression, astronomy, and bioinformatics. MDS had its origins in psychology, but recent applications to bioinformatics have enabled researchers, for example, to construct a global representation of the protein-structure universe. Other linear methods include projection pursuit, which constructs linear projections of high-dimensional data that display non-Gaussian features; independent component analysis, which constructs linear projections of the data that are mutually independent and (with at most one exception) non-Gaussian; and factor analysis, which tries to explain the correlation structure of a set of variables by modeling those variables as a linear combination of a small number of unobserved latent variables or factors. Many of the probability models used for machine learning have been interpreted as latent variable models, and nonlinear factor analysis [14] has been informative in revealing inadequacies in linear relationships between variables, in exploring underlying nonlinear structure in multivariate data, and in expressing ways of thinking about nonlinear manifold learning.

LINEAR MANIFOLD LEARNING

Prior to the year 2000, dimensionality reduction techniques focused on linear manifolds. A linear manifold is a line, a plane, or a hyperplane, depending upon the number of dimensions involved. There have been numerous examples of real data located in some high-dimensional space that have been shown to live close to a much lower dimensional linear manifold. In that sense, the low-dimensional linear manifold provides a succinct summary of the relationships between the observed variables. Linear dimensionality reduction is accomplished by constructing a few linear transformations of those variables, and the specific linear transformations are chosen to possess certain optimality properties.

The most popular linear dimensionality reduction techniques have been PCA, which constructs uncorrelated linear projections of the data, each projection of which has maximal variation, and MDS, which attempts to preserve pairwise distances to be used primarily for data display. Assuming the data $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Re^r$, PCA computes the eigenvalues and corresponding eigenvectors of the $(r \times r)$ sample covariance matrix; for a t-dimensional embedding, PCA retains only those eigenvalue–eigenvector combinations for which the t largest eigenvalues explain a high percentage of the total variation (sum of the eigenvalues) in the data. MDS starts with an $(n \times n)$ matrix $\mathbf{\Delta} = (\delta_{ij})$ of proximities (i.e., distances between all pairs of data points), rather than the data points themselves, and then computes the $(n \times n)$ matrices $\mathbf{A} = (\delta_{ij}^2)$ and $\mathbf{B} = -\frac{1}{2}\mathbf{HAH}$, where $\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{J}_n$ and $\mathbf{J}_n = \mathbf{1}_n\mathbf{1}_n^\tau$ is a matrix of ones. The final step in the MDS algorithm is to extract the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of $\mathbf{B}$, so that, for an embedding of dimension $t \ll r$, the principal coordinates of the embedding are given by the columns of the $(t \times n)$ matrix $\widehat{\mathbf{Y}} = (\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_n) = (\sqrt{\lambda_1}\mathbf{v}_1, \ldots, \sqrt{\lambda_t}\mathbf{v}_t)^\tau$. Refer to Ref 15, Chapters 7 and 13, for details of PCA and MDS, respectively.

NONLINEAR MANIFOLD LEARNING

Algorithms for nonlinear manifold learning seek to learn about the complete low-dimensional representation of an unknown nonlinear manifold that is embedded in some high-dimensional space. We also wish to retain the neighborhood structure of the manifold. In situations where the manifold is highly nonlinear, these algorithms are better at recovering the manifold than are the linear methods.

We can express the problem in the following terms. Let $\{\mathbf{y}_i\}$ represent a finite random sample of n data points that lie on a smooth t-dimensional manifold $\mathcal{M}$ with metric denoted by the geodesic distance,

$$d^{\mathcal{M}}(\mathbf{p}, \mathbf{q}) = \inf_{c \in \mathcal{C}(\mathbf{p},\mathbf{q})} L(c), \tag{1}$$

where $\mathcal{C}(\mathbf{p}, \mathbf{q})$ is the set of all differentiable curves in $\mathcal{M}$ that join up the points $\mathbf{p}$ and $\mathbf{q}$, and $L(c)$ is the arc length of the curve $c \in \mathcal{C}(\mathbf{p}, \mathbf{q})$.[3] These points are nonlinearly embedded by a smooth map ψ into high-dimensional space $\mathcal{X} = \Re^r$ with Euclidean metric $\|\cdot\|_{\mathcal{X}}$, where $t \ll r$. In other words, the embedding map is $\psi : \mathcal{M} \to \mathcal{X}$, and a point $\mathbf{y}$ on the manifold $\mathcal{M}$ is represented by $\mathbf{y} = \phi(\mathbf{x})$, $\mathbf{x} \in \mathcal{X}$, where $\phi = \psi^{-1}$. The goal is to 'learn' about the manifold $\mathcal{M}$ and find an explicit representation of the map ψ; that is, we use unsupervised learning algorithms to discover low-dimensional representations, $\{\mathbf{y}_i\}$ in $\mathcal{M}$, of high-dimensional input data points, $\{\mathbf{x}_i\}$ in $\mathcal{X}$, while remaining faithful to the structure of the input data. What 'faithful' means is that

inputs that are close to each other are mapped to outputs that are close to each other, while distant inputs are mapped to distant outputs. This feature of dimensionality reduction would be helpful, for example, in clustering applications by transforming a complicated high-dimensional problem into a much simpler low-dimensional one. Such a reformulation would also be of use in regression or prediction problems. In general, because it is impossible to visualize points in four or more dimensions, we retain only the two or three-dimensional solutions and plot the n corresponding vector points, $\{\mathbf{y}_i\}$, in two or three-dimensional space.

The algorithms we discuss here each involve three steps: (1) using neighborhood information around each data point, we construct a weighted graph with data points as vertices; (2) a step specific to the algorithm, in which the weighted neighborhood graph is transformed into suitable input for the next step; and (3) a spectral embedding step involving an $(n \times n)$ eigenequation computation. In this article, we describe a few of these nonlinear manifold learning algorithms, namely, Isomap, local linear embedding, Laplacian eigenmaps, Hessian eigenmaps, and diffusion maps. For the mathematical foundations behind nonlinear manifold learning see Ref 3.

Types of Embeddings and Algorithms

The manifold learning problem stated above is ill-posed as it stands, and needs some sort of restrictions on the type of embedding to make it work. Accordingly, embeddings are taken either to be isometric or conformal. Isometric embeddings preserve infinitesimal lengths and angles, while conformal embeddings preserve only infinitesimal angles.[16] In that sense, therefore, all isometric embeddings are special cases of conformal embeddings. The original Isomap algorithm can only recover an isometric embedding, while LLE can recover both types of embeddings. There is now a conformal version of Isomap.[17]

Algorithms for manifold learning can also be of two different kinds, either local methods or global methods, although hybrids of both methods have been proposed. Isomap is a global method because its embedding is based upon the geodesic distances between all pairs of points, while LLE is a local method because its embedding is based upon the relationship of each data point to its neighboring points. A local approach is computationally more efficient than a global approach because the former involves sparse matrices and is more generally applicable to different types of manifolds.

Isomap

Isomap is short for 'isometric feature mapping.'[1] The technique makes two assumptions: (1) the manifold $\mathcal{M}$ is a smooth convex region of $\Re^t$, $t < r$ and (2) the embedding $\psi : \mathcal{M} \to \mathcal{X}$ is an isometry. The second assumption implies that for any pair of points, $\mathbf{y} = \phi(\mathbf{x}), \mathbf{y}' = \phi(\mathbf{x}')$, on the manifold $\mathcal{M}$, the geodesic distance between those points equals the Euclidean distance between their corresponding coordinates, $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$; that is, $d^{\mathcal{M}}(\mathbf{y}, \mathbf{y}') = \|\mathbf{x} - \mathbf{x}'\|_{\mathcal{X}}$. Isomap views the manifold $\mathcal{M}$ as a convex region distorted in any of a number of ways, such as by folding or twisting. Examples of manifolds for which Isomap performs well include rolled-up sheets of paper, open boxes, or open cylinders. Isomap performs poorly when the manifold has holes,[5] which violates the convexity assumption.

Isomap is a nonlinear generalization of the MDS algorithm in which Euclidean distances are replaced by geodesic distances between points. The three steps are as follows:

1. Compute the distances $d_{ij}^{\mathcal{X}} = d^{\mathcal{X}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_{\mathcal{X}}$ between all pairs of points $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, $i, j = 1, 2, \ldots, n$, and then determine which data points are among either the K nearest neighbors or are within $\epsilon > 0$ of that point.

2. Compute a weighted neighborhood graph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$, where the vertices, $\mathcal{V} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, are the input data points and the edges, $\mathcal{E} = \{e_{ij}\}$, indicate the neighborhood relationships between the points. The edge e_{ij} that joins the neighboring points $\mathbf{x}_i$ and $\mathbf{x}_j$ has a weight w_{ij} equal to the distance $d_{ij}^{\mathcal{X}}$ between those points. If no edge exists between any two points, the corresponding weight is zero. The true geodesic distances $\{d_{ij}^{\mathcal{M}}\}$ on the manifold are estimated by the graph distances $\{d_{ij}^{\mathcal{G}}\}$, which are the shortest path distances between all pairs of points in the graph $\mathcal{G}$. Non-neighboring points are connected by a sequence of neighbor-to-neighbor links and the length of this path is the total of the link weights. The shortest path between every pair of vertices in a graph can be computed efficiently using Floyd's algorithm[18] or Dijkstra's algorithm.[19]

3. Apply classical MDS to spectral embedding. Collect the graph distances into an $(n \times n)$ symmetric matrix $\mathbf{D}^{\mathcal{G}} = (d_{ij}^{\mathcal{G}})$. Let $\mathbf{S}^{\mathcal{G}} = ([d_{ij}^{\mathcal{G}}]^2)$. Compute $\mathbf{A}_n^{\mathcal{G}} = -\frac{1}{2}\mathbf{H}\mathbf{S}^{\mathcal{G}}\mathbf{H}$, where $\mathbf{H} = \mathbf{I}_n - n^{-1}\mathbf{J}_n$ is a centering matrix and $\mathbf{J}_n$ is an $(n \times n)$-matrix of ones. The matrix $\mathbf{A}_n^{\mathcal{G}}$ will be nonnegative-definite of rank $t < n$. Denote by $\mathbf{A}_n^{\mathcal{Y}}$ the matrix $\mathbf{A}_n^{\mathcal{G}}$, but where $\mathbf{S}^{\mathcal{Y}} = ([d_{ij}^{\mathcal{Y}}]^2)$ replaces $\mathbf{S}^{\mathcal{G}}$ and $d_{ij}^{\mathcal{Y}} = \|\mathbf{y}_i - \mathbf{y}_j\|_{\mathcal{X}}$. Then, $\|\mathbf{A}_n^{\mathcal{G}} - \mathbf{A}_n^{\mathcal{Y}}\|$ is minimized by the eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_t$ corresponding to the t largest eigenvalues, $\lambda_1, \ldots, \lambda_t$, of $\mathbf{A}_n^{\mathcal{G}}$. The graph

$\mathcal{G}$ is embedded into $\mathcal{Y}$ by the $(t \times n)$-matrix $\widehat{\mathbf{Y}} = (\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_n) = (\sqrt{\lambda_1}\mathbf{v}_1, \ldots, \sqrt{\lambda_t}\mathbf{v}_t)^\tau$. The ith column, $\widehat{\mathbf{y}}_i$, of $\widehat{\mathbf{Y}}$ provides the embedding coordinates in $\mathcal{Y}$ of the ith data point.

Isomap also includes a graphical display for estimating the intrinsic dimensionality of the manifold. Let the $(n \times n)$-matrix $\mathbf{D}_t^{\mathcal{Y}}$ be the set of Euclidean distances between the n columns of $\widehat{\mathbf{Y}}$. Let $R_t^2 = [\text{corr}(\mathbf{D}_t^{\mathcal{Y}}, \mathbf{D}^{\mathcal{G}})]^2$ denote the squared correlation coefficient of all corresponding pairs of entries in the matrices $\mathbf{D}_t^{\mathcal{Y}}$ and $\mathbf{D}^{\mathcal{G}}$. The intrinsic demensionality t^* is that value of t at which an 'elbow' appears in the plot of $1 - R_t^2$ against t. More details and an example can be found in Ref 3. When n is very large, a landmark Isomap algorithm[17] is used to increase efficiency in computing.

Locally Linear Embedding

LLE[2] uses a completely different philosophy to manifold learning than does Isomap. Geometrically, LLE sees a manifold as a collection of possibly overlapping regions; if the number of points in a neighborhood is small and the manifold is smooth, then the regions will appear to be locally linear. LLE performs well when the data are uniformly sampled over the manifold and performs poorly when attempting to recover a closed manifold, such as a sphere, a torus, or a cylinder, which cannot be unfolded by LLE. The main difference between the LLE and Isomap algorithms is in the way that LLE carries out the second step. The three steps are as follows:

1. Fix $K \ll r$, but where $K > t$, and let N_i denote the K nearest-neighbor points of $\mathbf{x}_i$, $i = 1, 2, \ldots, n$. The neighborhood N_i could also represent those points that fall within a ball of radius ϵ of $\mathbf{x}_i$.

2. LLE assumes that every manifold is locally linear. So, we can approximate each point $\mathbf{x}_i$ by a linear function of its K nearest neighbors. Thus, we can write $\widehat{\mathbf{x}}_i = \sum_{i=1}^n w_{ij}\mathbf{x}_j$, where w_{ij} is a scalar weight for $\mathbf{x}_i$ with unit sum, $\sum_j w_{ij} = 1$, for translation invariance; if $\mathbf{x}_\ell \notin N_i$, then $w_{ij} = 0$. Let $\mathbf{W} = (w_{ij})$ be a sparse $(n \times n)$-matrix of weights. The optimization problem is to find an optimal set of weights $\widehat{\mathbf{W}} = (\widehat{w}_{ij})$ by solving

$$\widehat{\mathbf{W}} = \arg\min_{\mathbf{W}} \sum_{i=1}^n \|\mathbf{x}_i - \sum_{j=1}^n w_{ij}\mathbf{x}_j\|^2, \qquad (2)$$

subject to $\sum_j w_{ij} = 1$, $i = 1, 2, \ldots, n$, and the sparseness constraint $w_{i\ell} = 0$ if $\mathbf{x}_\ell \notin N_i$. Fix $\mathbf{x}_i$. We then write the summand as $\|\sum_j w_{ij}(\mathbf{x}_i - \mathbf{x}_j)\|^2 = \mathbf{w}_i^\tau \mathbf{G}_i \mathbf{w}_i$, where $\mathbf{w}_i = (w_{i1}, \ldots, w_{in})^\tau$, only K of which are nonzero, and $\mathbf{G}_i = (G_{i,jk})$ is an $(n \times n)$ Gram matrix (i.e., symmetric, nonnegative-definite) with $G_{i,jk} = (\mathbf{x}_i - \mathbf{x}_j)^\tau(\mathbf{x}_i - \mathbf{x}_k)$, $j, k \in N_i$. We form the Lagrangian function,

$$f(\mathbf{x}_i) = \mathbf{w}_i^\tau \mathbf{G}_i \mathbf{w}_i - \mu(\mathbf{1}_n^\tau \mathbf{w}_i - 1), \qquad (3)$$

which is minimized with respect to $\mathbf{w}_i$ by $\widehat{\mathbf{w}}_i = \frac{\mu}{2}\mathbf{G}_i^{-1}\mathbf{1}_n$. Premultiplying this last result by $\mathbf{1}_n^\tau$ yields the optimal weights

$$\widehat{\mathbf{w}}_i = \frac{\mathbf{G}_i^{-1}\mathbf{1}_n}{\mathbf{1}_n^\tau \mathbf{G}_i^{-1}\mathbf{1}_n}, \qquad (4)$$

where if $\mathbf{x}_\ell \notin N_i$, then $\widehat{w}_{i\ell} = 0$.

3. Fix the matrix $\widehat{\mathbf{W}} = (\widehat{w}_{ij})$ and compute the low-dimensional embedding. We wish to find the $(t \times n)$-matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $t \ll r$, of embedding coordinates that solves

$$\widehat{\mathbf{Y}} = \arg\min_{\mathbf{Y}} \sum_{i=1}^n \|\mathbf{y}_i - \sum_{j=1}^n \widehat{w}_{ij}\mathbf{y}_j\|^2, \qquad (5)$$

subject to the constraints on the $\{\mathbf{y}_i\}$ that the mean vector is zero and the covariance matrix is the identity; that is, $\sum_i \mathbf{y}_i = \mathbf{Y}\mathbf{1}_n = \mathbf{0}$ and $n^{-1}\sum_i \mathbf{y}_i\mathbf{y}_i^\tau = n^{-1}\mathbf{Y}\mathbf{Y}^\tau = \mathbf{I}_t$. We can write the matrix of embedding coordinates as

$$\widehat{\mathbf{Y}} = \arg\min_{\mathbf{Y}} \text{tr}\{\mathbf{Y}\mathbf{M}\mathbf{Y}^\tau\}, \qquad (6)$$

where $\mathbf{M}$ is the sparse, symmetric, and nonnegative-definite $(n \times n)$-matrix $\mathbf{M} = (\mathbf{I}_n - \widehat{\mathbf{W}})^\tau(\mathbf{I}_n - \widehat{\mathbf{W}})$. The objective function $\text{tr}\{\mathbf{Y}\mathbf{M}\mathbf{Y}^\tau\}$ has a unique global minimum given by the eigenvectors corresponding to the smallest $t+1$ eigenvalues of $\mathbf{M}$. The smallest eigenvalue of $\mathbf{M}$ is zero with corresponding eigenvector $\mathbf{v}_n = n^{-1/2}\mathbf{1}_n$. We can ignore the smallest eigenvalue and associated eigenvector because the sum of the coefficients of each of the other eigenvectors, which are orthogonal to $n^{-1/2}\mathbf{1}_n$, is zero, and hence this will constrain the embeddings to have sum zero. The optimal solution is $\widehat{\mathbf{Y}} = (\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_n) = (\mathbf{v}_{n-1}, \ldots, \mathbf{v}_{n-t})^\tau$, where $\mathbf{v}_{n-j}$ is the n-dimensional eigenvector corresponding to the $(j+1)$st smallest eigenvalue of $\mathbf{M}$.

Laplacian Eigenmaps

The Laplacian eigenmaps algorithm[4] is closely related to LLE, and in some situations, the solutions are equivalent. The three steps are

1. Define the set N_i to be the neighborhood of the point $\mathbf{x}_i$ composed of either the K nearest neighbors

of that point or by all those points that lie within an ϵ-neighborhood of that point.

2. For $\mathbf{x}_i$, compute a symmetric, $(n \times n)$, weighted adjacency matrix $\mathbf{W} = (w_{ij})$, where the distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined by an isotropic diffusion kernel, $w_{ij} = w_\sigma(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\}$ if $\mathbf{x}_j \in N_i$ and zero otherwise, $i, j = 1, 2, \ldots, n$, where $\sigma > 0$ is the kernel bandwidth. The further apart are the two points $\mathbf{x}_i$ and $\mathbf{x}_j$, the smaller will be the weight w_{ij}. Using the elements of $\mathbf{W}$, let $\mathcal{G}$ denote the resulting weighted graph.

3. Let $\mathbf{D} = (d_{ij})$ be an $(n \times n)$ diagonal matrix with diagonal elements $d_{ii} = \sum_{j \in N_i} w_{ij}$, $i = 1, 2, \ldots, n$. The $(n \times n)$ symmetric matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, which is called the graph Laplacian for the graph $\mathcal{G}$, is nonnegative-definite. The $(t \times n)$-matrix $\mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, which is used to embed the graph $\mathcal{G}$ into $\Re^t$, where $\mathbf{y}_i$ is the embedding coordinates of the ith point, is found by minimizing

$$\sum_i \sum_j w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 = \mathrm{tr}\{\mathbf{YLY}^\tau\}. \tag{7}$$

Thus, we seek the solution,

$$\widehat{\mathbf{Y}} = \arg \min_{\mathbf{Y}:\mathbf{YDY}^\tau = \mathbf{I}_t} \mathrm{tr}\{\mathbf{YLY}^\tau\}, \tag{8}$$

where $\mathbf{Y}$ is restricted so that $\mathbf{YDY}^\tau = \mathbf{I}_t$ to prevent a collapse onto a subspace of fewer than $t - 1$ dimensions. The solution of this minimization problem can be shown to be the solution of the generalized eigenequation $\mathbf{Lv} = \lambda\mathbf{Dv}$, which, in turn, is given by the eigenvalues and eigenvectors of the $(n \times n)$ symmetric matrix $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = (\ell_{ij}/\sqrt{d_{ii}d_{jj}})$, called the normalized graph Laplacian, where $\mathbf{L} = (\ell_{ij})$. If data are sampled uniformly from a low-dimensional manifold, the smallest eigenvectors of $\mathbf{P}$ form a discrete approximation to the Laplace–Beltrami operator on the manifold.[4] As with LLE, the smallest eigenvalue λ_n of $\mathbf{P}$ is zero with corresponding eigenvector $\mathbf{v}_n = \mathbf{1}_n$. Ignoring the smallest eigenvalue and vector, the rows of $\widehat{\mathbf{Y}}$ are the eigenvectors $\widehat{\mathbf{Y}} = (\widehat{\mathbf{Y}}_1, \ldots, \widehat{\mathbf{Y}}_n) = (\mathbf{v}_{n-1}, \ldots, \mathbf{v}_{n-t})^\tau$ corresponding to the next t smallest eigenvalues, $\lambda_{n-1} \leq \cdots \leq \lambda_{n-t}$, of $\mathbf{P}$.

Theoretical work on Laplacian eigenmaps has yielded the following asymptotic result.[20] If the data $\{\mathbf{x}_i\}$ are obtained as an independent uniformly distributed sample over the manifold $\mathcal{M}$, then, as $n \to \infty$ and the kernel bandwidth $\sigma \to 0$, the discrete graph Laplacian has been shown to converge to the continuous Laplace–Beltrami operator on the manifold. Additional large-sample results for variance and bias of the estimator are also available.[20]

Hessian Eigenmaps

Hessian eigenmaps[5] (also known as Hessian LLE or HLLE) have a lot in common with both LLE and Laplacian eigenmaps. Hessian eigenmaps were introduced to improve upon the local linearity assumption of LLE by employing a Hessian to compute the local curvature of the manifold at each data point. HLLE takes the Laplacian eigenmaps algorithm and replaces the Laplacian by the Hessian, where a Hessian of a function is the matrix of its partial second derivatives (see Refs 3 and 15, Chapter 16).

The low-dimensional representation that minimizes the curvature of the manifold is obtained from the smallest $t + 1$ eigenvectors of a discrete approximation to the Hessian. As in the above techniques, the smallest eigenvalue is zero and its eigenvector contains the constant functions. We discard both this smallest eigenvalue and its corresponding eigenvector. The remaining t smallest eigenvectors provide the desired embedding coordinates. HLLE assumes that the ambient space is (1) connected and (2) locally isometric to the manifold, unlike Isomap, which assumes that the ambient space is convex and the embedded space is globally isometric to the ambient space. HLLE can handle large amounts of data and performs well even when the manifold has holes.[5]

Diffusion Maps

Diffusion maps,[8,9] like Laplacian eigenmaps, were originally motivated as a nonlinear dimensionality reduction method and not as a solution to the manifold learning problem. Although diffusion maps are derived using a different type of argument than the other nonlinear manifold learning techniques, there are also three steps to the algorithm. The first two steps are similar to those of LLE and Laplacian eigenmaps. For the third step, the method constructs a Markov chain over a graph of the data points, and then carries out an eigenanalysis of the probability transition matrix of the Markov chain to obtain the embedding coordinates in low-dimensional space.

1. As with LLE and Laplacian eigenmaps, define N_i to be the set of K nearest-neighbor points (or ϵ-nearest-neighbor points) to the data point $\mathbf{x}_i$, $i = 1, 2, \ldots, n$.

2. The n data points $\{\mathbf{x}_i\}$ can be regarded as vertices of a graph $\mathcal{G}$. As edges (i.e., connections between pairs of adjacent vertices) in $\mathcal{G}$, use the appropriate entries of a weighted adjacency matrix $\mathbf{W} = (w_\sigma(\mathbf{x}_i, \mathbf{x}_j))$, where the distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined by an isotropic diffusion kernel, $w_\sigma(\mathbf{x}_i, \mathbf{x}_j) = \exp\{-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\}$, $\mathbf{x}_j \in N_i$, where σ is

the bandwidth of the kernel, and zero otherwise, as in the second step of Laplacian eigenmaps.

3. This is the spectral embedding step. As for Laplacian eigenmaps, construct the diagonal matrix $\mathbf{D} = (d_{ij})$ and the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$. We wish to solve the eigenequation $\mathbf{Lv} = \lambda\mathbf{Dv}$. The normalized graph Laplacian given by the $(n \times n)$-matrix $\mathbf{P} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is a stochastic matrix with all row sums equal to one. Hence, $\mathbf{P}$ can be interpreted as defining a random walk on the graph $\mathcal{G}$.[21] With this in mind, we define $\mathbf{X}(t)$ to be a Markov random walk over $\mathcal{G}$ using the weights in $\mathbf{W}$ and starting at an arbitrary point in $\mathcal{G}$ at time $t = 0$. The next point in our walk in one time step is determined by the probability transition matrix $\mathbf{P} = (p(\mathbf{x}_j|\mathbf{x}_i))$, whose ijth entry is

$$p(\mathbf{x}_j|\mathbf{x}_i) = P\{\mathbf{X}(t+1) = \mathbf{x}_j|\mathbf{X}(t) = \mathbf{x}_i\} = \frac{w_\sigma(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j=1}^{n} w_\sigma(\mathbf{x}_i, \mathbf{x}_j)}, \quad (9)$$

and whose row sums equal one. The matrix $\mathbf{P}$ defines the entire Markov chain on $\mathcal{G}$. $\mathbf{P}$ has eigenvalues $\lambda_0 = 1 \geq \lambda_1 \geq \cdots \geq \lambda_{n-1} \geq 0$ and two sets of eigenvectors, a left set defined by $\boldsymbol{\phi}_j^\tau \mathbf{P} = \lambda_j \boldsymbol{\phi}_j^\tau$ and a right set defined by $\mathbf{P}\boldsymbol{\psi}_k^\tau = \lambda_j \boldsymbol{\psi}_k^\tau$, where $\boldsymbol{\phi}_j = (\phi_j(\mathbf{x}_1), \ldots, \phi_j(\mathbf{x}_n))^\tau$ and $\boldsymbol{\psi}_k = (\psi_k(\mathbf{x}_1), \ldots, \psi_k(\mathbf{x}_n))^\tau$ are biorthogonal; that is, $\boldsymbol{\phi}_j^\tau \boldsymbol{\psi}_k = 1$ if $j = k$ and zero otherwise, $j, k = 1, 2, \ldots, n$. The largest eigenvalue, $\lambda_0 = 1$, has associated right eigenvector $\boldsymbol{\psi}_0 = \mathbf{1}_n = (1, 1, \ldots, 1)^\tau$ and left eigenvector $\boldsymbol{\phi}_0$. Thus, $\mathbf{P}$ is diagonalizable as the product, $\mathbf{P} = \boldsymbol{\Psi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^\tau$, where $\boldsymbol{\Phi} = (\boldsymbol{\phi}_0, \ldots, \boldsymbol{\phi}_{n-1})$, $\boldsymbol{\Psi} = (\boldsymbol{\psi}_0, \ldots, \boldsymbol{\psi}_{n-1})$, and $\boldsymbol{\Lambda} = \text{diag}\{\lambda_0, \ldots, \lambda_{n-1}\}$. Let $\mathbf{P}^m = (p_m(\mathbf{x}_j|\mathbf{x}_i))$, where the ijth entry,

$$p_m(\mathbf{x}_j|\mathbf{x}_i) = P\{\mathbf{X}(t+m) = \mathbf{x}_j|\mathbf{X}(t) = \mathbf{x}_i\}, \quad (10)$$

represents the transition probability of going from point $\mathbf{x}_i$ to point $\mathbf{x}_j$ in m time steps. Fix $0 < m < \infty$. On the graph $\mathcal{G}$, we define the following diffusion distance between conditional probabilities:

$$d_m^2(\mathbf{x}_i, \mathbf{x}_j) = \|p_m(\cdot|\mathbf{x}_i) - p_m(\cdot|\mathbf{x}_j)\|^2 = \sum_{\mathbf{z}} (p_m(\mathbf{z}|\mathbf{x}_i) - p_m(\mathbf{z}|\mathbf{x}_j))^2 w(\mathbf{z}), \quad (11)$$

where $w(\mathbf{z}) = 1/\phi_0(\mathbf{z})$ is a weight function that penalizes differences occurring in regions of low-density more than in regions of high-density; $\phi_0(\mathbf{z})$ can be viewed as both the unique stationary probability distribution for the Markov chain on the graph $\mathcal{G}$ (after taking an infinite number of steps from any starting point) and as a nonparametric density estimate at the point $\mathbf{z}$.[15,20] The diffusion distance, which yields information regarding how many paths exist between the points $\mathbf{x}_i$ and $\mathbf{x}_j$, will be small if the points are connected by many paths in the graph. From the expression for $\mathbf{P}$, we see that $\mathbf{P}^m = \boldsymbol{\Psi}\boldsymbol{\Lambda}^m\boldsymbol{\Phi}$, with ijth entry,

$$p_m(\mathbf{x}_j|\mathbf{x}_i) = \phi_0(\mathbf{x}_j) + \sum_{k=1}^{n-1} \lambda_k^m \psi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j). \quad (12)$$

Substituting this expression into the diffusion distance and using the fact that ϕ_0 is constant (and can, therefore, be ignored), we have that

$$d_m^2(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=1}^{n-1} \lambda_k^{2m} (\psi_k(\mathbf{x}_i) - \psi_k(\mathbf{x}_j))^2. \quad (13)$$

The eigenvalues of $\mathbf{P}$ decay to zero relatively fast, the speed of decay depending upon the topology of the graph $\mathcal{G}$. So, only the first few terms in this sum need to be retained. We approximate the diffusion distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ by

$$d_m^2(\mathbf{x}_i, \mathbf{x}_j) \approx \sum_{k=1}^{t} \lambda_k^{2m} (\psi_k(\mathbf{x}_i) - \psi_k(\mathbf{x}_j))^2 = \|\boldsymbol{\Psi}_m(\mathbf{x}_i) - \boldsymbol{\Psi}_m(\mathbf{x}_j)\|^2, \quad (14)$$

where $\boldsymbol{\Psi}_m(\mathbf{x}) = (\lambda_1^m \psi_1(\mathbf{x}), \ldots, \lambda_t^m \psi_t(\mathbf{x}))^\tau$. The coordinates of the n t-vectors are, thus, given by the columns of $\widehat{\mathbf{Y}} = (\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_n) = (\boldsymbol{\Psi}_m(\mathbf{x}_1), \ldots, \boldsymbol{\Psi}_m(\mathbf{x}_n))$.

We mention the following asymptotic properties of diffusion maps.[22] If we assume that the data $\{\mathbf{x}_i\}$ constitute an iid sample drawn from some probability density $p(\mathbf{x})$ defined over the manifold $\mathcal{M}$, then, as $n \to \infty$, the random walk on the discrete graph converges to a random walk on the continuous manifold $\mathcal{M}$, and, as $\sigma \to 0$ in the diffusion kernel, the random walk on $\mathcal{M}$ (following a suitable scaling wrt time) converges, in turn, to a diffusion process whose probability density changes continuously over time.

Nonlinear PCA

A different approach to nonlinear manifold learning is to construct nonlinear versions of linear manifold learning techniques. Specifically, how does one generalize PCA to a nonlinear version of PCA? There have been several nonlinear PCA routines proposed, each motivated by a different optimality property of PCA; these nonlinear generalizations of PCA include polynomial PCA,[23] principal curves and surfaces,[24]

multilayer autoassociative neural networks,[25] and kernel PCA.[26] The most interesting fact is all kernel-based manifold learning algorithms have been shown to be special cases of kernel PCA.[27]

CONCLUSION

We have described various algorithms for linear and nonlinear manifold learning. Each algorithm boils down to the extraction of eigenvalues and eigenvectors from some large symmetric matrix, which is different for the different algorithms.

All of the nonlinear manifold learning algorithms are controlled by the tuning parameters K and ϵ, which control the size of the neighborhoods around each data point. Making K or ϵ too large or too small can change the nature and dimensionality of the embedding solution. Determining an optimal choice of K or ϵ is still an open question. Another important issue is how to deal with new data points. All of these algorithms are batch algorithms and so they would require one to rerun the entire algorithm on a data set consisting of the original data augmented by the new points. Thus, all the manifold learning algorithms suffer from not being generalizable. This also holds for data arriving sequentially, where running any of these algorithms repeatedly becomes computationally challenging. Some efforts in this direction have appeared; see Ref 28. A further issue is that each of these algorithms assumes that the data are randomly sampled from some probability distribution on the manifold. What happens if real data do not lie exactly on the manifold, but may be close to the manifold (in some probabilistic sense)? Can we still recover the manifold? In general, the closer the data points are to the manifold, the more likely the algorithms will recover the appropriate manifold. However, if some of the data points (possibly outliers) lie too far away from the nonlinear manifold, and are instead located closer to a different region of the manifold, then this will create havoc with the process of recovering the manifold structure. This problem is still an open one.

REFERENCES

1. Tenenbaum JB, de Silva V, Langford JC. A global geometric framework for nonlinear dimensionality reduction. *Science* 2000, 290:2319–2323.
2. Roweis ST, Saul LK. Nonlinear dimensionality reduction by locally linear embedding. *Science* 2000, 290:2323–2326.
3. Izenman AJ. Spectral embedding methods for manifold learning. In: Ma Y, Fu Y, eds. *Manifold Learning Theory and Applications*, chapter 1. Boca Raton, FL: CRC Press; 2012, 1–36.
4. Belkin M, Niyogi P. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Adv Neural Inform Process Syst* 2002, 14:585–591.
5. Donoho D, Grimes C. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proc Natl Acad Sci* 2003, 100:5591–5596.
6. Brand M. Charting a manifold. *Adv Neural Inform Process Syst* 2003, 15:961–968.
7. Zhang Z, Zha H. Principal manifolds and nonlinear dimension reduction via local tangent space alignment. *SIAM J Sci Comput* 2004, 26:313–338.
8. Nadler B, Lafon S, Coifman RR, Kevrekidis IG. Diffusion maps, spectral clustering, and eigenfunctions of fokker–planck operators. *Advances in Neural Information Processing Systems*, vol, 18. Cambridge, MA: MIT Press; 2005, 955–962.
9. Coifman RR, Lafon S. Diffusion maps. *Appl Comput Harmonic Anal* 2006, 21:5–30.
10. Ma Y, Fu Y, eds. *Manifold Learning Theory and Applications*. Boca Raton, FL: CRC Press; 2012.
11. Gorban AN, Kegl B, Wunsch DC, Zonovyev A, eds. *Principal Manifolds for Data Visualization and Dimensionality Reduction*. New York, NY: Springer; 2008.
12. Lee JA, Verleysen M. *Nonlinear Dimensionality Reduction*. New York, NY: Springer; 2007.
13. Carreira-Perpiñán MÁ. *Dimensionality Reduction*. Boca Raton, FL: CRC Press; 2013.
14. Yalcin I, Amemiya Y. Nonlinear factor analysis as a statistical method. *Statistical Science* 2001, 16:275–294.
15. Izenman AJ. *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. New York, NY: Springer; 2008.
16. De Silva V, Tenenbaum JB. Local versus global methods for nonlinear dimensionality reduction. *Advances in Neural Information Processing Systems*, vol. 15. Cambridge, MA: MIT Press; 2003, 705–712.
17. De Silva V, Tenenbaum JB. Unsupervised learning of curved manifolds. In: Denison DD, Hansen MH, Holmes CC, Mallick B, Yu B, eds. *Nonlinear Estimation and Classification*, vol. 171. New York: Springer; 2003, 453–466.
18. Floyd RW. Algorithm 97. *Commun ACM* 1962, 5:345.
19. Dijkstra EW. A note on two problems in connection with graphs. *Numer Math* 1959, 1:269–271.

20. Singer A. From graph to manifold laplacian: the convergence rate. *Appl Comput Harmonic Anal* 2006, 21:128–134.
21. Meila M, Shi J. A random walks view of spectral segmentation. In Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics (AISTATS). San Francisco, CA, 2001, 92–97.
22. Nadler B, Lafon S, Coifman RR, Kevrekidis IG. Diffusion maps, spectral clustering, and the reaction coordinates of dynamical systems. *Appl Comput Harmonic Anal* 2006, 21:113–127.
23. Gnadesikan R, Wilk MB. Data analytic methods in multivariate statistical analysis. In: Krishnaiah PR, ed. *Multivariate Analysis II*. New York: Academic Press; 1969.
24. Hastie T, Steutzle W. Principal curves. *J Am Stat Assoc* 1989, 84:502–516.
25. Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. *AIChE J* 1991, 37:233–243.
26. Schölkopf B, Smola AJ, Muller K-R. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Comput* 1998, 10:1299–1319.
27. Ham J, Lee DD, Mika S, Schölkopf B. A kernel view of the dimensionality reduction of manifolds, Technical Report TR–110. Germany: Max Planck Institute für biologische Kybernetik, 2003.
28. Law MHC, Zhang N, Jain A. Nonlinear manifold learning for data stream. In: Berry MW, Dayal U, Kamath C, Skillicorn D, eds. *Proceedings of the Fourth SIAM International Conference in Data Mining*, vol. 4. Florida: SIAM; 2004, 33–44.

