1 Spectral Methods for Dimensionality Reduction

Lawrence K. Saul
Kilian Q. Weinberger
Fei Sha
Jihun Ham
Daniel D. Lee

How can we search for low dimensional structure in high dimensional data? If the data is mainly confined to a low dimensional subspace, then simple linear methods can be used to discover the subspace and estimate its dimensionality. More generally, though, if the data lies on (or near) a low dimensional submanifold, then its structure may be highly nonlinear, and linear methods are bound to fail.

Spectral methods have recently emerged as a powerful tool for nonlinear dimensionality reduction and manifold learning. These methods are able to reveal low dimensional structure in high dimensional data from the top or bottom eigenvectors of specially constructed matrices. To analyze data that lies on a low dimensional submanifold, the matrices are constructed from sparse weighted graphs whose vertices represent input patterns and whose edges indicate neighborhood relations. The main computations for manifold learning are based on tractable, polynomial-time optimizations, such as shortest path problems, least squares fits, semidefinite programming, and matrix diagonalization. This chapter provides an overview of unsupervised learning algorithms that can be viewed as spectral methods for linear and nonlinear dimensionality reduction.

1.1 Introduction

dimensionality reduction

The problem of dimensionality reduction—extracting low dimensional structure from high dimensional data—arises often in machine learning and statistical pattern recognition. High dimensional data takes many different forms: from digital image libraries to gene expression microarrays, from neuronal population activities to financial time series. By formulating the problem of dimensionality reduction in a general setting, however, we can analyze many different types of data in the same underlying mathematical framework.

We therefore consider the following problem. Given a high dimensional data set $X = (x_1, \ldots, x_n)$ of input patterns where $x_i \in \mathbb{R}^d$, how can we compute n corresponding output patterns $\psi_i \in \mathbb{R}^m$ that provide a "faithful" low dimensional representation of the original data set with $m \ll d$? By faithful, we mean generally that nearby inputs are mapped to nearby outputs, while faraway inputs are mapped to faraway outputs; we will be more precise in what follows. Ideally, an unsupervised learning algorithm should also estimate the value of m that is required for a faithful low dimensional representation. Without loss of generality, we assume everywhere in this chapter that the inputs are centered on the origin, with $\sum_i x_i = 0 \in \mathbb{R}^d$.

inputs $x_i \in \mathbb{R}^d$
outputs $\psi_i \in \mathbb{R}^m$

This chapter provides a survey of so-called spectral methods for dimensionality reduction, where the low dimensional representations are derived from the top or bottom eigenvectors of specially constructed matrices. The aim is not to be exhaustive, but to describe the simplest forms of a few representative algorithms using terminology and notation consistent with the other chapters in this book. At best, we can only hope to provide a snapshot of the rapidly growing literature on this subject. An excellent and somewhat more detailed survey of many of these algorithms is given by Burges [2005]. In the interests of both brevity and clarity, the examples of nonlinear dimensionality reduction in this chapter were chosen specifically for their pedagogical value; more interesting applications to data sets of images, speech, and text can be found in the original papers describing each method.

spectral methods

The chapter is organized as follows. In section 1.2, we review the classical methods of principal component analysis (PCA) and metric multidimensional scaling (MDS). The outputs returned by these methods are related to the input patterns by a simple linear transformation. The remainder of the chapter focuses on the more interesting problem of nonlinear dimensionality reduction. In section 1.3, we describe several graph-based methods that can be used to analyze high dimensional data that has been sampled from a low dimensional submanifold. All of these graph-based methods share a similar structure—computing nearest neighbors of the input patterns, constructing a weighted graph based on these neighborhood relations, deriving a matrix from this weighted graph, and producing an embedding from the top or bottom eigenvectors of this matrix. Notwithstanding this shared structure, however, these algorithms are based on rather different geometric intuitions and intermediate computations. In section 1.4, we describe kernel-based methods for nonlinear dimensionality reduction and show how to interpret graph-based methods in this framework. Finally, in section 1.5, we conclude by contrasting the properties of different spectral methods and highlighting various ongoing lines of research. We also point out connections to related work on semi-supervised learning, as described by other authors in this volume.

1.2 Linear methods

Principal components analysis (PCA) and metric multidimensional scaling (MDS) are simple spectral methods for linear dimensionality reduction. As we shall see in later sections, however, the basic geometric intuitions behind PCA and MDS also play an important role in many algorithms for nonlinear dimensionality reduction.

1.2.1 Principal components analysis (PCA)

PCA is based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves its covariance structure (up to rotation). In PCA, the input patterns $x_i \in \mathbb{R}^d$ are projected into the m-dimensional subspace that minimizes the reconstruction error,

minimum reconstruction error

$$\mathcal{E}_{\mathrm{PCA}} = \sum_i \left\| x_i - \sum\nolimits_{\alpha=1}^{m} (x_i \cdot e_\alpha)\, e_\alpha \right\|^2, \tag{1.1}$$

where the vectors $\{e_\alpha\}_{\alpha=1}^{m}$ define a partial orthonormal basis of the input space. From eq. (1.1), one can easily show that the subspace with minimum reconstruction error is also the subspace with maximum variance. The basis vectors of this subspace are given by the top m eigenvectors of the $d \times d$ covariance matrix,

covariance matrix

$$C = \frac{1}{n} \sum_i x_i x_i^\top, \tag{1.2}$$

assuming that the input patterns x_i are centered on the origin. The outputs of PCA are simply the coordinates of the input patterns in this subspace, using the directions specified by these eigenvectors as the principal axes. Identifying e_α as the α^{th} top eigenvector of the covariance matrix, the output $\psi_i \in \mathbb{R}^m$ for the input pattern $x_i \in \mathbb{R}^d$ has elements $\psi_{i\alpha} = x_i \cdot e_\alpha$. The eigenvalues of the covariance matrix in eq. (1.2) measure the projected variance of the high dimensional data set along the principal axes. Thus, the number of significant eigenvalues measures the dimensionality of the subspace that contains most of the data's variance, and a prominent gap in the eigenvalue spectrum indicates that the data is mainly confined to a lower dimensional subspace. Fig. 1.1 shows the results of PCA applied to a toy data set in which the inputs lie within a thin slab of three dimensional space. Here, a simple linear projection reveals the data's low dimensional (essentially planar) structure. More details on PCA can be found in Jolliffe [1986]. We shall see in section 1.3.2 that the idea of reducing dimensionality by maximizing variance is also useful for nonlinear dimensionality reduction.

1.2.2 Metric multidimensional scaling (MDS)

Metric MDS is based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves the inner products between

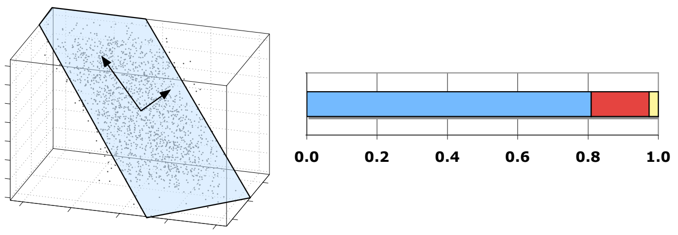

Figure 1.1 Results of PCA applied to $n = 1600$ input patterns in $d = 3$ dimensions that lie within a thin slab. The top two eigenvectors of the covariance matrix, denoted by black arrows, indicate the $m = 2$ dimensional subspace of maximum variance. The eigenvalues of the covariance matrix are shown normalized by their sum: each eigenvalue is indicated by a colored bar whose length reflects its partial contribution to the overall trace of the covariance matrix. There are two dominant eigenvalues, indicating that the data is very nearly confined to a plane.

different input patterns. The outputs $\psi_i \in \mathbb{R}^m$ of metric MDS are chosen to minimize:

$$\mathcal{E}_{\mathrm{MDS}} = \sum_{ij} (x_i \cdot x_j - \psi_i \cdot \psi_j)^2. \tag{1.3}$$

The minimum error solution is obtained from the spectral decomposition of the Gram matrix of inner products,

Gram matrix

$$G_{ij} = x_i \cdot x_j. \tag{1.4}$$

Denoting the top m eigenvectors of this Gram matrix by $\{v_\alpha\}_{\alpha=1}^m$ and their respective eigenvalues by $\{\lambda_\alpha\}_{\alpha=1}^m$, the outputs of MDS are given by $\psi_{i\alpha} = \sqrt{\lambda_\alpha} v_{\alpha i}$.

distance preservation

Though MDS is designed to preserve inner products, it is often motivated by the idea of preserving pairwise distances. Let $S_{ij} = \|x_i - x_j\|^2$ denote the matrix of squared pairwise distances between input patterns. Often the input to MDS is specified in this form. Assuming that the inputs are centered on the origin, a Gram matrix consistent with these squared distances can be derived from the transformation $G = -\frac{1}{2}(I - uu^\top)S(I - uu^\top)$, where I is the $n \times n$ identity matrix and $u = \frac{1}{\sqrt{n}}(1, 1, \ldots, 1)^\top$ is the uniform vector of unit length. More details on MDS can be found in Cox and Cox [1994].

Though based on a somewhat different geometric intuition, metric MDS yields the same outputs $\psi_i \in \mathbb{R}^m$ as PCA—essentially a rotation of the inputs followed by a projection into the subspace with the highest variance. (The outputs of both algorithms are invariant to global rotations of the input patterns.) The Gram matrix of metric MDS has the same rank and eigenvalues up to a constant factor as the covariance matrix of PCA. In particular, letting X denote the $d \times n$ matrix of input patterns, then $C = n^{-1} X X^\top$ and $G = X^\top X$, and the equivalence follows from singular value decomposition. In both matrices, a large gap between the m^{th} and $(m+1)^{\mathrm{th}}$ eigenvalues indicates that the high dimensional input patterns lie to a good approximation in a lower dimensional subspace of dimensionality m. As we shall see in sections 1.3.1 and 1.4.1, useful nonlinear generalizations of metric MDS are obtained by substituting generalized pairwise distances and inner products in place of Euclidean measurements.

1.3 Graph–based methods

Linear methods such as PCA and metric MDS generate faithful low dimensional representations when the high dimensional input patterns are mainly confined to a low dimensional subspace. If the input patterns are distributed more or less throughout this subspace, the eigenvalue spectra from these methods also reveal the data set's intrinsic dimensionality—that is to say, the number of underlying modes of variability. A more interesting case arises, however, when the input patterns lie on or near a low dimensional submanifold of the input space. In this case, the structure of the data set may be highly nonlinear, and linear methods are bound to fail.

Graph-based methods have recently emerged as a powerful tool for analyzing high dimensional data that has been sampled from a low dimensional submanifold. These methods begin by constructing a sparse graph in which the nodes represent input patterns and the edges represent neighborhood relations. The resulting graph (assuming, for simplicity, that it is connected) can be viewed as a discretized approximation of the submanifold sampled by the input patterns. From these graphs, one can then construct matrices whose spectral decompositions reveal the low dimensional structure of the submanifold (and sometimes even the dimensionality itself). Though capable of revealing highly nonlinear structure, graph-based methods for manifold learning are based on highly tractable (i.e., polynomial-time) optimizations such as shortest path problems, least squares fits, semidefinite programming, and matrix diagonalization. In what follows, we review four broadly representative graph-based algorithms for manifold learning: Isomap [Tenenbaum et al., 2000], maximum variance unfolding [Weinberger and Saul, 2005, Sun et al., 2005], locally linear embedding [Roweis and Saul, 2000, Saul and Roweis, 2003], and Laplacian eigenmaps [Belkin and Niyogi, 2003].

1.3.1 Isomap

geodesic distances

Isomap is based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves the pairwise distances between input patterns *as measured along the submanifold from which they were sampled.* The algorithm can be understood as a variant of MDS in which estimates of geodesic distances along the submanifold are substituted for standard Euclidean distances. Fig. 1.2 illustrates the difference between these two types of distances for input patterns sampled from a Swiss roll.

The algorithm has three steps. The first step is to compute the k-nearest neighbors of each input pattern and to construct a graph whose vertices represent input patterns and whose (undirected) edges connect k-nearest neighbors. The edges are then assigned weights based on the Euclidean distance between nearest neighbors. The second step is to compute the pairwise distances Δ_{ij} between all nodes (i,j) along shortest paths through the graph. This can be done using Djikstra's algorithm which scales as $O(n^2 \log n + n^2 k)$. Finally, in the third step, the pairwise distances Δ_{ij} from Djikstra's algorithm are fed as input to MDS, as described in section 1.2.2, yielding low dimensional outputs $\psi_i \in \mathbb{R}^m$ for which $\|\psi_i - \psi_j\|^2 \approx \Delta_{ij}^2$. The value of m required for a faithful low dimensional representation can be estimated by the number of significant eigenvalues in the Gram matrix constructed by MDS.

When it succeeds, Isomap yields a low dimensional representation in which the Euclidean distances between outputs match the geodesic distances between input patterns on the submanifold from which they were sampled. Moreover, there are formal guarantees of convergence [Tenenbaum et al., 2000, Donoho and Grimes, 2002] when the input patterns are sampled from a submanifold that is isometric to a convex subset of Euclidean space—that is, if the data set has no "holes". This condition will be discussed further in section 1.5.

1.3.2 Maximum variance unfolding

Maximum variance unfolding [Weinberger and Saul, 2005, Sun et al., 2005] is based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves the distances and angles between *nearby* input patterns. Like Isomap, it appeals to the notion of isometry and constructs a Gram matrix

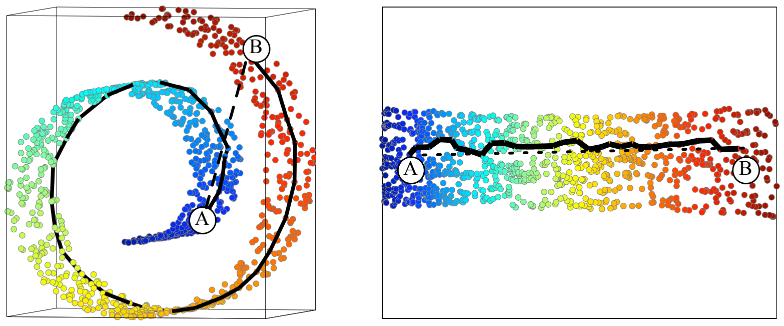

Figure 1.2 *Left:* comparison of Euclidean and geodesic distance between two input patterns A and B sampled from a Swiss roll. Euclidean distance is measured along the straight line in input space from A to B; geodesic distance is estimated by the shortest path (in bold) that only directly connects $k = 12$ nearest neighbors. *Right:* the low dimensional representation computed by Isomap for $n = 1024$ inputs sampled from a Swiss roll. The Euclidean distances between outputs match the geodesic distances between inputs.

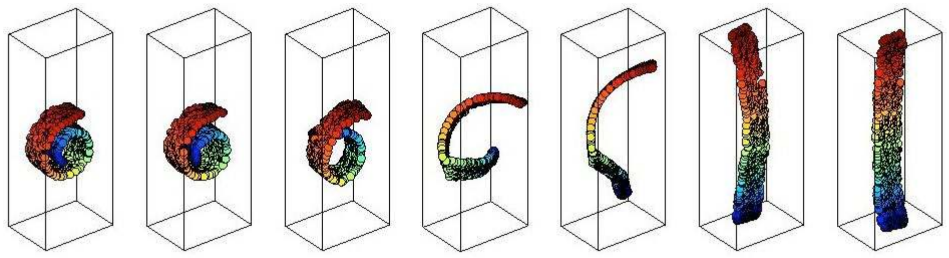

Figure 1.3 Input patterns sampled from a Swiss roll are "unfolded" by maximizing their variance subject to constraints that preserve local distances and angles. The middle snapshots show various feasible (but non-optimal) intermediate solutions of the optimization described in section 1.3.2.

whose top eigenvectors yield a low dimensional representation of the data set; unlike Isomap, however, it does not involve the estimation of geodesic distances. Instead, the algorithm attempts to "unfold" a data set by pulling the input patterns apart as far as possible subject to distance constraints that ensure that the final transformation from input patterns to outputs looks *locally* like a rotation plus translation. To picture such a transformation from $d=3$ to $m=2$ dimensions, one can imagine a flag being unfurled by pulling on its four corners (but not so hard as to introduce any tears).

The first step of the algorithm is to compute the k-nearest neighbors of each input pattern. A neighborhood-indicator matrix is defined as $\eta_{ij}=1$ if and only if the input patterns x_i and x_j are k-nearest neighbors or if there exists another input pattern of which both are k-nearest neighbors; otherwise $\eta_{ij}=0$. The constraints to preserve distances and angles between k-nearest neighbors can be written as:

$$\|\psi_i - \psi_j\|^2 = \|x_i - x_j\|^2 , \tag{1.5}$$

for all (i,j) such that $\eta_{ij}=1$. To eliminate a translational degree of freedom in the low dimensional representation, the outputs are also constrained to be centered on the origin:

$$\sum_i \psi_i = 0 \in \mathbb{R}^m. \tag{1.6}$$

Finally, the algorithm attempts to "unfold" the input patterns by maximizing the variance of the outputs,

$$\mathrm{var}(\psi) = \sum_i \|\psi_i\|^2 , \tag{1.7}$$

while preserving local distances and angles, as in eq. (1.5). Fig. 1.3 illustrates the connection between maximizing variance and reducing dimensionality.

The above optimization can be reformulated as an instance of semidefinite programming [Vandenberghe and Boyd, 1996]. A semidefinite program is a linear program with the additional constraint that a matrix whose elements are linear in semidefinite programming

the optimization variables must be positive semidefinite. Let $K_{ij} = \psi_i \cdot \psi_j$ denote the Gram matrix of the outputs. The constraints in eqs. (1.5–1.7) can be written entirely in terms of the elements of this matrix. Maximizing the variance of the outputs subject to these constraints turns out to be a useful surrogate for minimizing the rank of the Gram matrix (which is computationally less tractable). The Gram matrix K of the "unfolded" input patterns is obtained by solving the semidefinite program:

Maximize trace(K) subject to:
1) $K \succeq 0$.
2) $\Sigma_{ij} K_{ij} = 0$.
3) $K_{ii} - 2K_{ij} + K_{jj} = |||x_i - x_j||^2$ **for all** (i, j) **such that** $\eta_{ij} = 1$.

The first constraint indicates that the matrix K is required to be positive semidefinite. As in MDS and Isomap, the outputs are derived from the eigenvalues and eigenvectors of this Gram matrix, and the dimensionality of the underlying submanifold (i.e., the value of m) is suggested by the number of significant eigenvalues.

1.3.3 Locally linear embedding (LLE)

LLE is based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves the local linear structure of nearby input patterns [Roweis and Saul, 2000]. The algorithm differs significantly from Isomap and maximum variance unfolding in that its outputs are derived from the bottom eigenvectors of a sparse matrix, as opposed to the top eigenvectors of a (dense) Gram matrix.

The algorithm has three steps. The first step, as usual, is to compute the k-nearest neighbors of each high dimensional input pattern x_i. In LLE, however, one constructs a *directed* graph whose edges indicate nearest neighbor relations (which may or may not be symmetric). The second step of the algorithm assigns weights W_{ij} to the edges in this graph. Here, LLE appeals to the intuition that each input pattern and its k-nearest neighbors can be viewed as samples from a small linear "patch" on a low dimensional submanifold. Weights W_{ij} are computed by reconstructing each input pattern x_i from its k-nearest neighbors. Specifically, they are chosen to minimize the reconstruction error:

local linear reconstructions

$$\mathcal{E}_W = \sum_i \left\| x_i - \sum\nolimits_j W_{ij} x_j \right\|^2 . \tag{1.8}$$

The minimization is performed subject to two constraints: (i) $W_{ij} = 0$ if x_j is not among the k-nearest neighbors of x_i; (ii) $\sum_j W_{ij} = 1$ for all i. (A regularizer can also be added to the reconstruction error if its minimum is not otherwise well-defined.) The weights thus constitute a sparse matrix W that encodes local geometric properties of the data set by specifying the relation of each input pattern x_i to its k-nearest neighbors.

In the third step, LLE derives outputs $\psi_i \in \mathbb{R}^m$ that respect (as faithfully as possible) these same relations to their k-nearest neighbors. Specifically, the outputs are chosen to minimize the cost function:

$$\mathcal{E}_\psi = \sum_i \left\| \psi_i - \sum\nolimits_j W_{ij}\psi_j \right\|^2 . \tag{1.9}$$

The minimization is performed subject to two constraints that prevent degenerate solutions: (i) the outputs are centered, $\sum_i \psi_i = 0 \in \mathbb{R}^m$, and (ii) the outputs have unit covariance matrix. The d-dimensional embedding that minimizes eq. (1.9) subject to these constraints is obtained by computing the bottom $m+1$ eigenvectors of the matrix $(I-W)^\top(I-W)$. The bottom (constant) eigenvector is discarded, and the remaining m eigenvectors (each of size n) then yield the low dimensional outputs $\psi_i \in \mathcal{R}^m$. Unlike the top eigenvalues of the Gram matrices in Isomap and maximum variance unfolding, the bottom eigenvalues of the matrix $(I–W)^\top(I–W)$ in LLE do not have a telltale gap that indicates the dimensionality of the underlying manifold. Thus the LLE algorithm has two free parameters: the number of nearest neighbors k and the target dimensionality m.

sparse eigenvalue problem

Fig. 1.4 illustrates one particular intuition behind LLE. The leftmost panel shows $n = 2000$ inputs sampled from a Swiss roll, while the rightmost panel shows the two dimensional representation discovered by LLE, obtained by minimizing eq. (1.9) subject to centering and orthogonality constraints. The middle panels show the results of minimizing eq. (1.9) *without centering and orthogonality constraints*, but with $\ell < n$ randomly chosen outputs constrained to be equal to their corresponding inputs. Note that in these middle panels, the outputs have the same dimensionality as the inputs. Thus, the goal of the optimization in the middle panels is not dimensionality reduction; rather, it is locally linear reconstruction of the entire data set from a small sub-sample. For sufficiently large ℓ, this alternative optimization is well-posed, and minimizing eq. (1.9) over the remaining $n-\ell$ outputs is done by solving a simple least squares problem. For $\ell = n$, the outputs of this optimization are equal to the original inputs; for smaller ℓ, they resemble the inputs, but with slight errors due to the linear nature of the reconstructions; finally, as ℓ is decreased further, the outputs provide an increasingly linearized representation of the original data set. LLE (shown in the rightmost panel) can be viewed a limit of this procedure as $\ell \to 0$, with none of the outputs clamped to the inputs, but with other constraints imposed to ensure that the optimization is well-defined.

1.3.4 Laplacian eigenmaps

Laplacian eigenmaps are based on computing the low dimensional representation of a high dimensional data set that most faithfully preserves proximity relations, mapping nearby input patterns to nearby outputs. The algorithm has a similar structure as LLE. First, one computes the k-nearest neighbors of each high dimensional input pattern x_i and constructs the symmetric undirected graph whose n nodes represent input patterns and whose edges indicate neighborhood relations

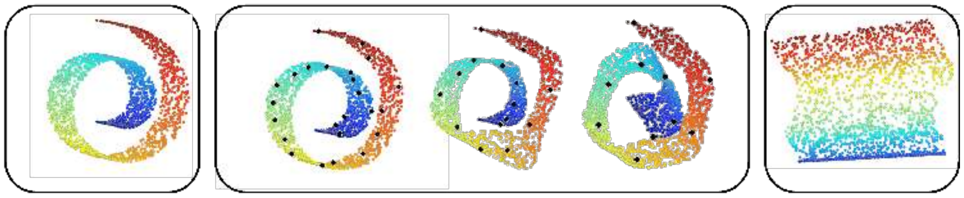

Figure 1.4 Intuition behind LLE. *Left:* $n = 2000$ input patterns sampled from a Swiss roll. Middle: results of minimizing of eq. (1.9) with $k = 20$ nearest neighbors and $\ell = 25$, $\ell = 15$, and $\ell = 10$ randomly chosen outputs (indicated by black landmarks) clamped to the locations of their corresponding inputs. *Right:* two dimensional representation obtained by minimizing eq. (1.9) with no outputs clamped to inputs, but subject to the centering and orthogonality constraints of LLE.

(in either direction). Second, one assigns positive weights W_{ij} to the edges of this graph; typically, the values of the weights are either chosen to be constant, say $W_{ij} = 1/k$, or exponentially decaying, as $W_{ij} = \exp(-\|x_i - x_j\|^2/\sigma^2)$ where σ^2 is a scale parameter. Let $\mathbf{D}$ denote the diagonal matrix with elements $D_{ii} = \sum_j W_{ij}$. In the third step of the algorithm, one obtains the outputs $\psi_i \in \mathbb{R}^m$ by minimizing the cost function:

$$\mathcal{E}_{\mathcal{L}} = \sum_{ij} \frac{W_{ij}\,\|\psi_i - \psi_j\|^2}{\sqrt{D_{ii}D_{jj}}}. \qquad (1.10)$$

proximity-preserving embedding

This cost function encourages nearby input patterns to be mapped to nearby outputs, with "nearness" measured by the weight matrix $\mathbf{W}$. As in LLE, the minimization is performed subject to constraints that the outputs are centered and have unit covariance. The minimum of eq. (1.10) is computed from the bottom $m+1$ eigenvectors of the matrix $\mathcal{L} = I - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$. The matrix $\mathcal{L}$ is a symmetrized, normalized form of the graph Laplacian, given by $\mathbf{D} - \mathbf{W}$. As in LLE, the bottom (constant) eigenvector is discarded, and the remaining m eigenvectors (each of size n) yield the low dimensional outputs $\psi_i \in \mathcal{R}^m$. Again, the optimization is a sparse eigenvalue problem that scales relatively well to large data sets.

1.4 Kernel Methods

Suppose we are given a real-valued function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ with the property that there exists a map $\Phi : \mathbb{R}^d \to \mathcal{H}$ into a dot product "feature" space $\mathcal{H}$ such that for all $x, x' \in \mathbb{R}^d$, we have $\Phi(x) \cdot \Phi(x') = k(x, x')$. The kernel function $k(x, x')$ can be viewed as a nonlinear similarity measure. Examples of kernel functions that satisfy the above criteria include the polynomial kernels $k(x, x') = (1 + x \cdot x')^p$ for positive integers p and the Gaussian kernels $k(x, x') = \exp(-\|x - x'\|^2/\sigma^2)$. Many linear methods in statistical learning can be generalized to nonlinear settings by employing the so-called "kernel trick" — namely, substituting these generalized dot products in feature space for Euclidean dot products in the space of input patterns [Schölkopf and Smola, 2002]. In section 1.4.1, we review the nonlinear generalization of PCA [Schölkopf et al., 1998] obtained in this way, and in section 1.4.2, we discuss the relation between kernel PCA and the manifold learning algorithms of section 1.3. Our treatment closely follows that of Ham et al. [2004].

1.4.1 Kernel PCA

Given input patterns $(x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^d$, kernel PCA computes the principal components of the feature vectors $(\Phi(x_1), \ldots, \Phi(x_n))$, where $\Phi(x_i) \in \mathcal{H}$. Since in general $\mathcal{H}$ may be infinite-dimensional, we cannot explicitly construct the covariance matrix in feature space; instead we must reformulate the problem so that it can be solved in terms of the kernel function $k(x, x')$. Assuming that the data has zero mean in the feature space $\mathcal{H}$, its covariance matrix is given by:

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^\top. \tag{1.11}$$

To find the top eigenvectors of $\mathbf{C}$, we can exploit the duality of PCA and MDS mentioned earlier in section 1.2.2. Observe that all solutions to $\mathbf{Ce} = \nu\mathbf{e}$ with $\nu \neq 0$ must lie in the span of $(\Phi(x_1), \ldots, \Phi(x_n))$. Expanding the αth eigenvector as $e_\alpha = \sum_i v_{\alpha i}\Phi(x_i)$ and substituting this expansion into the eigenvalue equation, we obtain a dual eigenvalue problem for the coefficients $v_{\alpha i}$, given by $Kv_\alpha = \lambda_\alpha v_\alpha$, where $\lambda_\alpha = n\nu_\alpha$ and $K_{ij} = k(x_i, x_j)$ is the so-called kernel matrix—that is, the Gram matrix in feature space. We can thus interpret kernel PCA as a nonlinear version of MDS that results from substituting generalized dot products in feature space for Euclidean dot products in input space [Williams, 2001]. Following the prescription for MDS in section 1.2.2, we compute the top m eigenvalues and eigenvectors of the kernel matrix. The low dimensional outputs $\psi_i \in \mathbb{R}^m$ of kernel PCA (or equivalently, kernel MDS) are then given by $\psi_{i\alpha} = \sqrt{\lambda_\alpha}v_{\alpha i}$.

One modification to the above procedure often arises in practice. In (1.11), we have assumed that the feature vectors in $\mathcal{H}$ have zero mean. In general, we cannot assume this, and therefore we need to subtract the mean $(1/n)\sum_i \Phi(x_i)$ from each feature vector before computing the covariance matrix in eq. (1.11). This leads to a slightly different eigenvalue problem, where we diagonalize $K' = (I - uu^\top)K(I - uu^\top)$ rather than K, where $u = \frac{1}{\sqrt{n}}(1, \ldots, 1)^\top$.

Kernel PCA is often used for nonlinear dimensionality reduction with polynomial or Gaussian kernels. It is important to realize, however, that these generic kernels are not particularly well suited to manifold learning, as described in section 1.3. Fig. 1.5 shows the results of kernel PCA with polynomial ($p = 4$) and Gaussian kernels applied to $n = 1024$ input patterns sampled from a Swiss roll. In neither case do the top two eigenvectors of the kernel matrix yield a faithful low dimensional representation of the original input patterns, nor do the eigenvalue spectra suggest that the input patterns were sampled from a two dimensional submanifold.

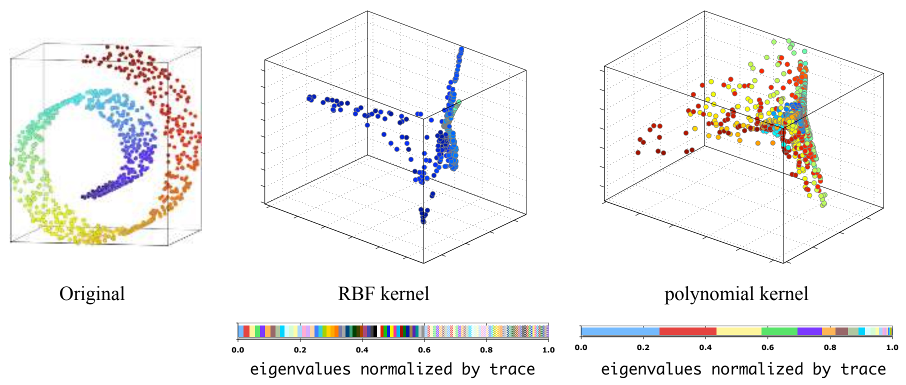

Figure 1.5 Results of kernel PCA with Gaussian and polynomial kernels applied to $n = 1024$ input patterns sampled from a Swiss roll. These kernels do not lead to low dimensional representations that unfold the Swiss roll.

1.4.2 Graph-Based Kernels

All of the algorithms in section 1.3 can be viewed as instances of kernel PCA, with kernel matrices that are derived from sparse weighted graphs rather than a pre-defined kernel function [Ham et al., 2004]. Often these kernels are described as "data-dependent" kernels, because they are derived from graphs that encode the neighborhood relations of the input patterns in the training set. These kernel matrices may also be useful for other tasks in machine learning besides dimensionality reduction, such as classification and nonlinear regression [Belkin et al., 2004]. In this section, we discuss how to interpret the matrices of graph-based spectral methods as kernel matrices.

The Isomap algorithm in section 1.3.1 computes a low dimensional embedding by computing shortest paths through a graph and processing the resulting distances by MDS. The Gram matrix constructed by MDS from these geodesic distances can be viewed as a kernel matrix. For finite data sets, however, this matrix is not guaranteed to be positive semidefinite. It should therefore be projected onto the cone of positive semidefinite matrices before it is used as a kernel matrix in other settings.

Maximum variance unfolding in section 1.3.2 is based on learning a Gram matrix by semidefinite programming. The resulting Gram matrix can be viewed as a kernel matrix. In fact, this line of work was partly inspired by earlier work that used semidefinite programming to learn a kernel matrix for classification in support vector machines [Lanckriet et al., 2004].

The algorithms in sections 1.3.3 and 1.3.4 do not explicitly construct a Gram matrix, but the matrices that they diagonalize can be related to operators on graphs and interpreted as "inverse" kernel matrices. For example, the discrete graph Laplacian arises in the description of diffusion on graphs and can be related to Green's functions and heat kernels in this way [Kondor and Lafferty, 2002, Coifman et al., 2005]. In particular, recall that in Laplacian eigenmaps, low dimensional representations are derived from the bottom (non-constant) eigenvectors of the graph Laplacian L. These bottom eigenvectors are equal to the top eigenvectors of the pseudo-inverse of the Laplacian, $L^{\dagger}$, which can thus be viewed as a (centered) kernel matrix for kernel PCA. Moreover, viewing the elements $L^{\dagger}_{ij}$ as inner products, the squared distances defined by $L^{\dagger}_{ii}+L^{\dagger}_{jj}-L^{\dagger}_{ij}-L^{\dagger}_{ji}$ are in fact proportional to the round-trip commute times of the continuous-time Markov chain with transition rate matrix L. The commute times are nonnegative, symmetric, and satisfy the triangle inequality; thus, Laplacian eigenmaps can be alternately be viewed as MDS on the metric induced by these graph commute times. (A slight difference is that the outputs of Laplacian eigenmaps are normalized to have unit covariance, whereas in MDS the scale of each dimension would be determined by the corresponding eigenvalue of $L^{\dagger}$.)

The matrix diagonalized by LLE can also be interpreted as an operator on graphs, whose pseudo-inverse corresponds to a kernel matrix. The operator does not generate a simple diffusive process, but in certain cases, it acts similarly to the square of the graph Laplacian [Ham et al., 2004].

The above analysis provides some insight into the differences between Isomap, maximum variance unfolding, Laplacian eigenmaps, and LLE. The metrics induced by Isomap and maximum variance unfolding are related to geodesic and local distances, respectively, on the submanifold from which the input patterns are sampled. On the other hand, the metric induced by the graph Laplacian is related to the commute times of Markov chains; these times involve all the connecting paths between two nodes on a graph, not just the shortest one. The kernel matrix induced by LLE is roughly analogous to the square of the kernel matrix induced by the graph Laplacian. In many applications, the kernel matrices in Isomap and maximum variance unfolding have telltale gaps in their eigenvalue spectra that indicate the dimensionality of the underlying submanifold from which the data was sampled. On the other hand, those from Laplacian eigenmaps and LLE do not reflect the geometry of the submanifold in this way.

1.5 Discussion

Each of the spectral methods for nonlinear dimensionality reduction has its own advantages and disadvantages. Some of the differences between the algorithms have been studied in formal theoretical frameworks, while others have simply emerged over time from empirical studies. We conclude by briefly contrasting the statistical, geometrical, and computational properties of different spectral methods and describing how these differences often play out in practice.

theoretical guarantees

Most theoretical work has focused on the behavior of these methods in the limit $n\rightarrow\infty$ of large sample size. In this limit, if the input patterns are sampled from a submanifold of $\mathbb{R}^d$ that is isometric to a convex subset of Euclidean space—that is, if the data set contains no "holes"—then the Isomap algorithm from section 1.3.1

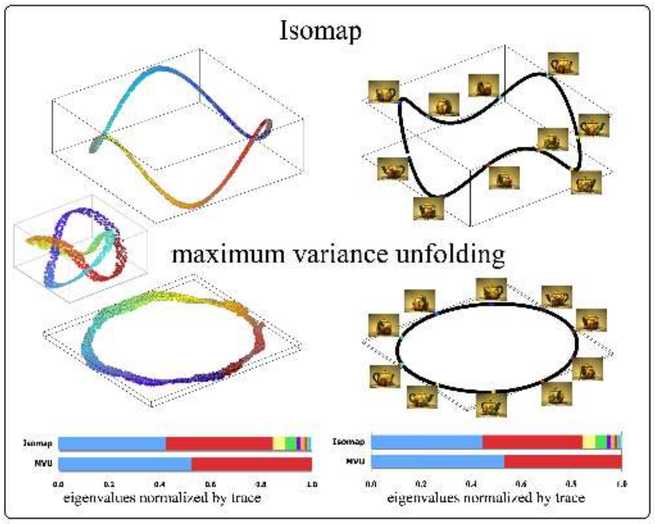

Figure 1.6 Results of Isomap and maximum variance unfolding on two data sets whose underlying submanifolds are not isometric to convex subsets of Euclidean space. *Left:* 1617 input patterns sampled from a trefoil knot. *Right:* $n = 400$ images of a teapot rotated through 360 degrees. The embeddings are shown, as well as the eigenvalues of the Gram matrices, normalized by their trace. The algorithms estimate the dimensionality of the underlying submanifold by the number of appreciable eigenvalues. Isomap is foiled in this case by non-convexity.

will recover this subset up to a rigid motion [Tenenbaum et al., 2000]. Many image manifolds generated by translations, rotations, and articulations can be shown to fit into this framework [Donoho and Grimes, 2002]. A variant of LLE known as Hessian LLE has also been developed with even broader guarantees [Donoho and Grimes, 2003]. Hessian LLE asymptotically recovers the low dimensional parameterization (up to rigid motion) of any high dimensional data set whose underlying submanifold is isometric to an open, connected subset of Euclidean space; unlike Isomap, the subset is not required to be convex.

manifolds with "holes"

The asymptotic convergence of maximum variance unfolding has not been studied in a formal setting. Unlike Isomap, however, the solutions from maximum variance unfolding in section 1.3.2 are guaranteed to preserve distances between nearest neighbors for any finite set of n input patterns. Maximum variance unfolding also behaves differently than Isomap on data sets whose underlying submanifold is isometric to a connected but not convex subset of Euclidean space. Fig. 1.6 contrasts the behavior of Isomap and maximum variance unfolding on two data sets with this property.

computation

Of the algorithms described in section 1.3, LLE and Laplacian eigenmaps scale best to moderately large data sets ($n < 10000$), provided that one uses special-purpose eigensolvers that are optimized for sparse matrices. The internal iterations of these eigensolvers rely mainly on matrix-vector multiplications which can be done in $O(n)$. The computation time in Isomap tends to be dominated by the calculation of shortest paths. The most computationally intensive algorithm is maximum variance unfolding, due to the expense of solving semidefinite programs [Vandenberghe and Boyd, 1996] over $n \times n$ matrices.

For significantly larger data sets, all of the above algorithms present serious challenges: the bottom eigenvalues of LLE and Laplacian eigenmaps can be tightly spaced, making it difficult to resolve the bottom eigenvectors, and the computational bottlenecks of Isomap and maximum variance unfolding tend to be prohibitive. Accelerated versions of Isomap and maximum variance unfolding have been developed by first embedding a small subset of "landmark" input patterns, then using various approximations to derive the rest of the embedding from the landmarks. The landmark version of Isomap [de Silva and Tenenbaum, 2003] is based on the Nyström approximation and scales very well to large data sets [Platt, 2004]; millions of input patterns can be processed in minutes on a PC (though the algorithm makes the same assumption as Isomap that the data set contains no "holes"). The landmark version of maximum variance unfolding [Weinberger et al., 2005] is based on a factorized approximation of the Gram matrix, derived from local linear reconstructions of the input patterns (as in LLE). It solves a much smaller SDP that the original algorithm and can handle larger data sets (currently, up to $n = 20000$), though it is still much slower than the landmark version of Isomap. Note that all the algorithms rely as a first step on computing nearest neighbors, which naively scales as $O(n^2)$, but faster algorithms are possible based on specialized data structures [Friedman et al., 1977, Gray and Moore, 2001, Beygelzimer et al., 2004].

related work

Research on spectral methods for dimensionality reduction continues at a rapid pace. Other algorithms closely related to the ones covered here include hessian LLE [Donoho and Grimes, 2003], c-Isomap [de Silva and Tenenbaum, 2003], local tangent space alignment [Zhang and Zha, 2004], geodesic nullspace analysis [Brand, 2004], and conformal eigenmaps [Sha and Saul, 2005]. Motivation for ongoing work includes the handling of manifolds with more complex geometries, the need for robustness to noise and outliers, and the ability to scale to large data sets.

In this chapter, we have focused on nonlinear dimensionality reduction, a problem in unsupervised learning. Graph-based spectral methods also play an important role in semi-supervised learning. For example, the eigenvectors of the normalized graph Laplacian provide an orthonormal basis—ordered by smoothness—for all functions (including decision boundaries and regressions) defined over the neighborhood graph of input patterns; see chapter **?** by Belkin, Sindhwani, and Niyogi. Likewise, as discussed in chapter **?** by Zhu and Kandola, the kernel matrices learned by unsupervised algorithms can be transformed by discriminative training for the purpose of semi-supervised learning. Finally, in chapter **?**, Vincent, Bengio, Hein, and Zien show how shortest-path calculations and multidimensional scaling can be used to derive more appropriate feature spaces in a semi-supervised setting. In all these ways, graph-based spectral methods are emerging to address the very broad class of problems that lie between the extremes of purely supervised and unsupervised learning.

References

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *Proceedings of the Seventeenth Annual Conference on Computational Learning Theory (COLT 2004)*, pages 624–638, Banff, Canada, 2004.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor, 2004. Submitted for publication.

M. Brand. From subspaces to submanifolds. In *Proceedings of the British Machine Vision Conference*, London, England, 2004.

C. J. C. Burges. Geometric methods for feature extraction and dimensional reduction. In L. Rokach and O. Maimon, editors, *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic Publishers, 2005.

R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucke. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences*, 102:7426–7431, 2005.

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

V. de Silva and J. B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 721–728, Cambridge, MA, 2003. MIT Press.

D. L. Donoho and C. E. Grimes. When does Isomap recover the natural parameterization of families of articulated images? Technical Report 2002-27, Department of Statistics, Stanford University, August 2002.

D. L. Donoho and C. E. Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Arts and Sciences*, 100:5591–5596, 2003.

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

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

J. Ham, D. D. Lee, S. Mika, and B. Schölkopf. A kernel view of the dimensionality reduction of manifolds. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, pages 369–376, Banff, Canada, 2004.

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

R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-02)*, 2002.

G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

J. C. Platt. Fast embedding of sparse similarity graphs. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

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

L. K. Saul and S. T. Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, 4:119–155, 2003.

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

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

F. Sha and L. K. Saul. Analysis and extension of spectral methods for nonlinear dimensionality reduction. In *Proceedings of the Twenty Second International Conference on Machine Learning (ICML-05)*, Bonn, Germany, 2005.

J. Sun, S. Boyd, L. Xiao, and P. Diaconis. The fastest mixing Markov process on a graph and a connection to a maximum variance unfolding problem. *SIAM Review*, 2005. Submitted.

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

L. Vandenberghe and S. P. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, March 1996.

K. Q. Weinberger, B. D. Packer, and L. K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *Proceedings of the Tenth International Workshop on AI and Statistics (AISTATS-05)*, Barbados, WI, 2005.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal on Computer Vision*, 2005. Submitted.

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

Z. Zhang and H. Zha. Principal manifolds and nonlinear dimensionality reduction by local tangent space alignment. *SIAM Journal of Scientific Computing*, 26(1):313–338, 2004.