Dimensionality Reduction, Manifolds
10716, Spring 2020

Pradeep Ravikumar (amending notes by Larry Wasserman)

We consider two related problems: (i) using low dimensional approximations for dimension reduction and (ii) estimating low dimensional structure.

1 Dimension Reduction

We might not want to estimate lower dimensional structure. Instead, we might just want to use a low dimensional approximation to the data to make other tasks easier. This is dimension reduction.

1.1 Principal Component Analysis (PCA)

Principal components analysis (PCA) finds low dimensional approximations to the data by projecting the data onto linear subspaces.

Let $X \in \mathbb{R}^d$ and let $\mathcal{L}_k$ denote all k-dimensional linear subspaces. The k^{th} principal subspace is

$$\ell_k = \operatorname*{argmin}_{\ell \in \mathcal{L}_k} \mathbb{E}\left(\min_{y \in \ell} \|\widetilde{X} - y\|^2 \right)$$

where $\widetilde{X} = X - \mu$ and $\mu = \mathbb{E}(X)$. The dimension-reduced version of X is then $T_k(X) = \mu + \pi_{\ell_k} X$ where and $\pi_{\ell_k} X$ is the projection of X onto ℓ_k. To find ℓ_k proceed as follows.

Let $\Sigma = \mathbb{E}((X-\mu)(X-\mu)^T)$ denote the covariance matrix, where $\mu = \mathbb{E}(X)$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ be the ordered eigenvalues of Σ and let $e_1, \ldots, e_d$ be the corresponding eigenvectors. Let Λ be the diagonal matrix with $\Lambda_{jj} = \lambda_j$ and let $E = [e_1 \cdots e_d]$. Then the spectral decomposition of Σ is

$$\Sigma = E \Lambda E^T = \sum_j \lambda_j e_j e_j^T.$$

Theorem 1 *The k^{th} principal subspace ℓ_k is the subspace spanned by $e_1, \ldots, e_k$. Furthermore,*

$$T_k(X) = \mu + \sum_{j=1}^{k} \beta_j e_j$$

where $\beta_j = \langle X - \mu, e_j \rangle$. *The risk satisfies*

$$R(k) = \mathbb{E}\|X - T_k(X)\|^2 = \sum_{j=k+1}^{d} \lambda_j.$$

We can restate the result as follows. To minimize

$$\mathbb{E}\|X_i - \alpha - A\beta_i\|^2,$$

with respect to $\alpha \in \mathbb{R}^d$, $A \in \mathbb{R}^{d \times k}$ and $\beta_i \in \mathbb{R}^k$ we set $\alpha = \mu$ and $A = [e_1\ e_2\ \cdots\ e_k]$. Any other solution is equivalent in the sense that it corresponds to the same subspace.

We can choose k by fixing some α and then taking

$$k = \min\left\{ m :\ \frac{R(m)}{R(0)} \leq \alpha \right\} = \min\left\{ m :\ \frac{\sum_{j=1}^{m} \lambda_j}{\sum_{j=1}^{d} \lambda_j} \geq 1 - \alpha \right\}.$$

Let $Y = (Y_1, \ldots, Y_d)$ where $Y_i = e_i^T(X - \mu)$. Then Y is the PCA-transformation applied to X. The random variable Y has the following properties:

Lemma 2 *We have:*

1. $\mathbb{E}[Y] = 0$ *and* $\mathrm{Var}(Y) = \Lambda$.
2. $X = \mu + EY$.
3. $\sum_{j=1}^{m} \mathrm{Var}(Y_j) = \Sigma_{11} + \cdots + \Sigma_{mm}$.

Hence,

$$\frac{\sum_{j=1}^{m} \lambda_j}{\sum_{j=1}^{d} \lambda_j}$$

is the percentage of variance explained by the first m principal components.

The data version of PCA is obtained by replacing Σ with the sample covariance matrix

$$\widehat{\Sigma} = \frac{1}{n} \sum_{i=1}^{n} (X_i - \overline{X}_n)(X_i - \overline{X}_n)^T.$$

Principal Components Analysis (PCA)

1. Compute the sample covariance matrix $\widehat{\Sigma} = n^{-1}\sum_{i=1}^{n}(X_i - \overline{X}_n)(X_i - \overline{X}_n)^T$.
2. Compute the eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots$ and eigenvectors $e_1, e_2, \ldots,$ of $\widehat{\Sigma}$.
3. Choose a dimension k.
4. Define the dimension reduced data $Z_i = T_k(X_i) = \overline{X} + \sum_{j=1}^{k}\beta_{ij}e_j$ where $\beta_{ij} = \langle X_i - \overline{X}, e_j\rangle$.

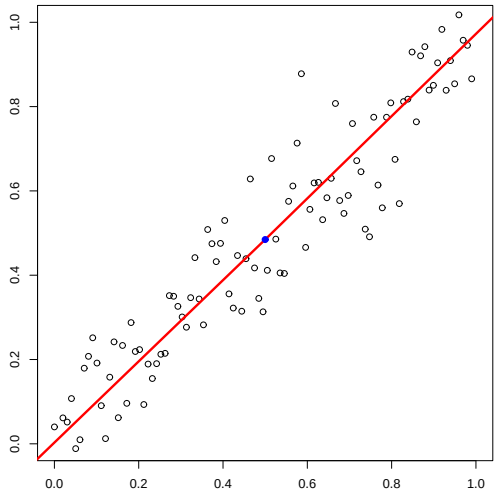

Figure 1: First principal component (red line) in a simple two dimensional example.

Example 3 *Figure 1 shows a synthetic two-dimensional data set together with the first principal component.*

Example 4 *Figure 2 shows some handwritten digits. The eigenvalues and the first few eigenfunctions are shown in Figures 3 and 4. A few digits and their low-dimensional reconstructions are shown in Figure 5.*

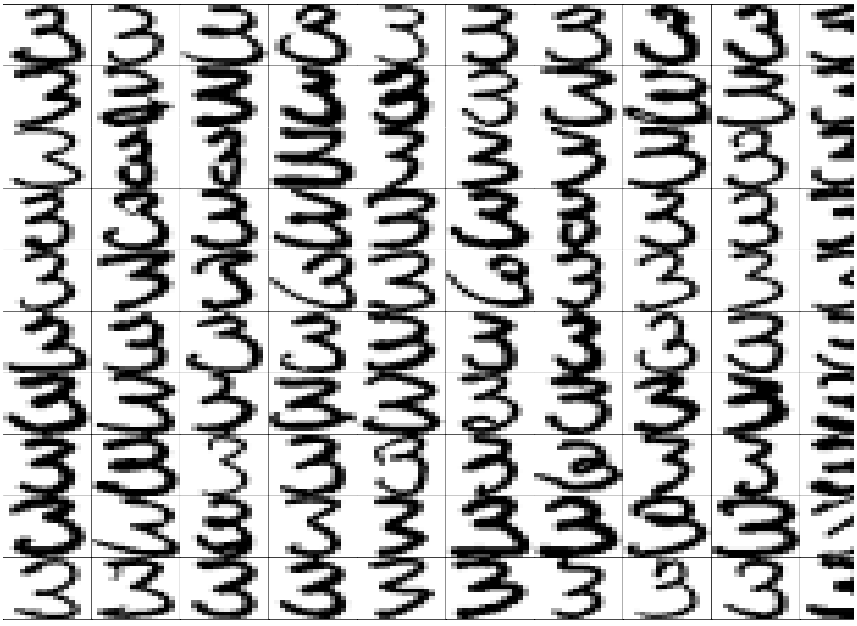

Figure 2: Handwritten digits.

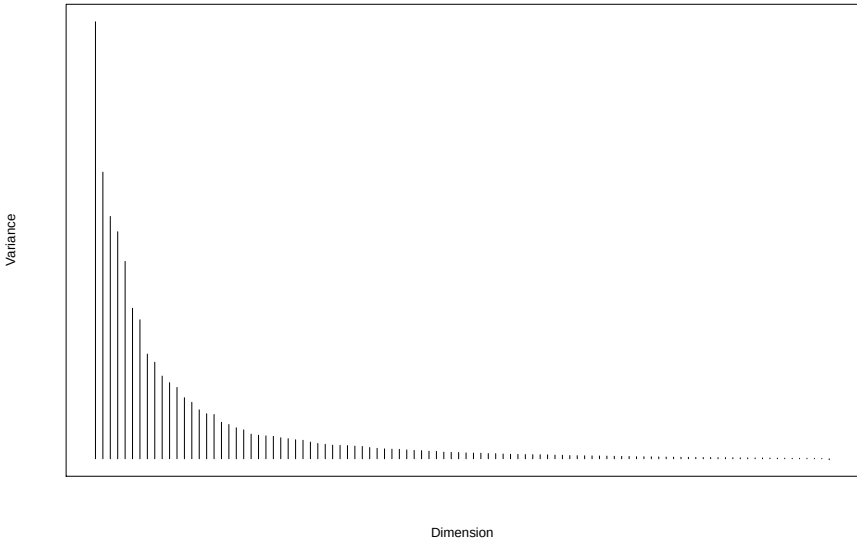

Figure 3: Digits data: eigenvalues

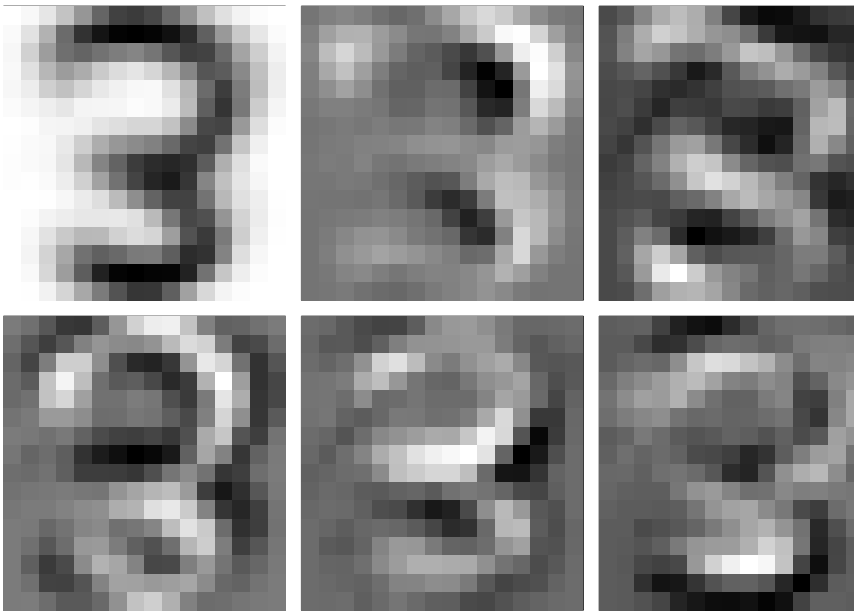

Figure 4: Digits: mean and eigenvectors

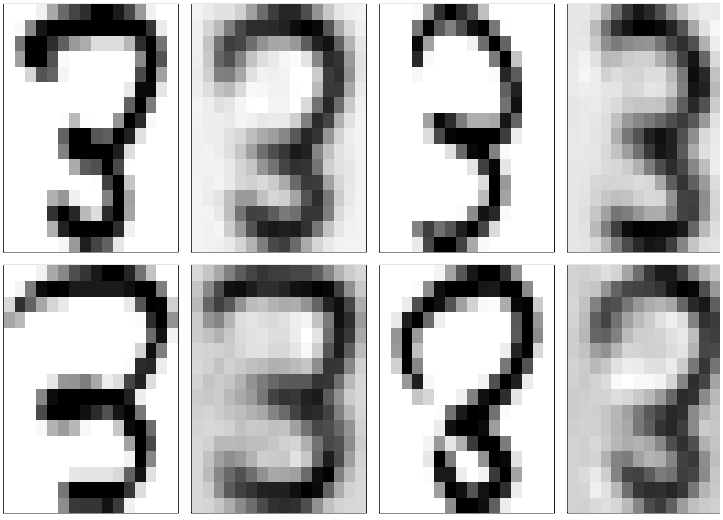

Figure 5: Digits data: Top: digits. Bottom: their reconstructions.

How well does the sample version approximate the population version? For now, assume the dimensions d is fixed and that n is large.

Define the operator norm

$$||\Sigma|| = \sup\left\{ \frac{||\Sigma v||}{||v||} : \ v \neq 0 \right\}.$$

It can be shown that $||\widehat{\Sigma} - \Sigma|| = O_P(1/\sqrt{n})$. According to Weyl's theorem

$$\max_j |\lambda_j(\widehat{\Sigma}) - \lambda_j(\Sigma)| \leq ||\widehat{\Sigma} - \Sigma||$$

and hence, the estimated eigenvalues are consistent. We can also say that the eigenvectors are consistent. We have

$$||\widehat{e}_j - e_j|| \leq \frac{2^{3/2}||\widehat{\Sigma} - \Sigma||}{\min(\lambda_{j-1} - \lambda_j, \lambda_j - \lambda_{j+1})}.$$

(See Yu, Wang and Samworth, arXiv:1405.0680.)

There is a strong connection between PCA and the singular value decomposition (SVD). Let X be an $n \times d$ matrix. The SVD is

$$X = UDV^T$$

where U is an $n \times n$ matrix with orthonormal columns, V is a $d \times d$ matrix with orthonormal columns, and D is an $n \times d$ diagonal matrix with non-negative real numbers on the diagonal (called singular values). Then

$$X^T X = (VDU^T)(UDV^T) = VD^2V^T$$

and hence the singular values are the square root of the eigenvalues of the sample covariance matrix.

1.2 Multidimensional Scaling

A different view of dimension reduction is provided by thinking in terms of preserving pairwise distances. Suppose that $Z_i = T(X_i)$ for $i = 1, \ldots, n$ where $T : \mathbb{R}^d \to \mathbb{R}^k$ with $k < d$. Define the loss function

$$L = \sum_{i,j} (||X_i - X_j||^2 - ||Z_i - Z_j||^2)$$

which measures how well the map T preserves pairwise distances. **Multidimensional scaling** find the linear map T to minimize L.

Theorem 5 *The linear map $T : \mathbb{R}^d \to \mathbb{R}^k$ that minimizes L is the projection onto* $\mathrm{Span}\{e_1, \ldots, e_k\}$ *where $e_1, \ldots, e_k$ are the first k principal components.*

We could use other measures of distortion. In that case, the MDS solution and the PCA solution will not coincide.

1.3 Kernel PCA

To get a ninlinear version of PCA, we can use a kernel. Suppose we have a "feature map" $x \mapsto \Phi(x)$ and want to carry out PCA in this new feature space. To do so however, would require given the empirical covariance matrix of these feature vectors: (a) computing the corresponding eigenvectors, and (b) projecting any feature vector $\Phi(x)$ onto these eigenvectors. Using the kernel trick, we will be able to do so tractably even for infinite-dimensional feature maps.

For the moment, assume that the feature vectors are centered (we return to this point shortly). Define the empirical covariance matrix

$$C_\Phi = \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T.$$

We can define eigenvalues $\lambda_1, \lambda_2, \ldots$ and eigenvectors $v_1, v_2, \ldots$ of this matrix.

It turns out that the eigenvectors are linear combinations of the feature vectors $\Phi(x_1), \ldots, \Phi(x_n)$. To see this, note that

$$\begin{aligned} \lambda v = C_\Phi v &= \frac{1}{n} \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T v \\ &= \frac{1}{n} \sum_{i=1}^{n} \langle\, \Phi(x_i), v \,\rangle\, \Phi(x_i) = \sum_{i=1}^{n} \alpha_i \Phi(x_i) \end{aligned}$$

where

$$\alpha_i = \frac{1}{n} \langle\, \Phi(x_i), v \,\rangle\ = \frac{1}{n\lambda} \langle\, \Phi(x_i), C_\Phi v \,\rangle\ .$$

Estimating weights α. But how to estimate these weights α? Multiplying both sides of the identity,

$$\lambda \sum_{i=1}^{n} \alpha_i \Phi(x_i) = Cv,$$

by $\Phi(x_k)$, we get:

$$\begin{aligned}
\lambda \sum_{i=1}^{n} \alpha_i \left\langle \Phi(x_k), \Phi(x_i) \right\rangle &= \lambda \left\langle \Phi(x_k), Cv \right\rangle \\
&= \lambda \left\langle \Phi(x_k), \frac{1}{n} \sum_{j=1}^{n} \Phi(x_j)\Phi(x_j)^T v \right\rangle \\
&= \left\langle \Phi(x_k), \frac{1}{n} \sum_{j=1}^{n} \Phi(x_j)\Phi(x_j)^T \sum_{i=1}^{n} \alpha_i \Phi(x_i) \right\rangle \\
&= \frac{1}{n} \sum_{i=1}^{n} \alpha_i \left\langle \Phi(x_k), \sum_{j=1}^{n} \left\langle \Phi(x_j), \Phi(x_i) \right\rangle \Phi(x_j) \right\rangle .
\end{aligned}$$

Define the kernel matrix K by $K_{ij} = \left\langle \Phi(x_i), \Phi(x_j) \right\rangle$. Then we can write the above equation as

$$\lambda n K \alpha = K^2 \alpha$$

Thus, to obtain the weights α, we simply need to solve the kernel eigenvalue problem

$$K\alpha = n\lambda\alpha$$

which requires diagonalizing only an $n \times n$ system. We can further show that normalizing the eigenvectors, $\left\langle v, v \right\rangle = 1$ leads to the condition $\lambda \left\langle \alpha, \alpha \right\rangle = 1$.

Projecting onto eigenvectors. In order to compute the kernel PCA projection of a new test point x, it is necessary to project the feature vector $\Phi(x)$ onto the principal direction v_m. This requires the evaluation

$$\begin{aligned}
\left\langle v, \Phi(x) \right\rangle &= \sum_{i=1}^{n} \alpha_i \left\langle \Phi(x_i), \Phi(x) \right\rangle \\
&= \sum_{i=1}^{n} \alpha_i K(x_i, x).
\end{aligned}$$

Thus, the entire procedure — computing the eigenvectors, as well as projecting onto these eigenvectors — uses only the kernel evaluations $K(x, x_i)$ and never requires actual manipulation of feature vectors, which could be infinite dimensional. This is an instance of the *kernel trick*. An arbitrary data point x can then be approximated by projecting $\Phi(x)$ onto the first k vectors. This defines an approximation in the feature space.

One question might be what is the input $\widetilde{x}$ that corresponds to this projection. There is an iterative algorithm for doing this (Mika et al 1998) which turns out to be a weighed version of the mean shift algorithm.

Overall Algorithm. To complete the description of the algorithm, it is necessary to explain how to center the data in feature space using only kernel operations. This is accomplished by transforming the kernel according to

$$\widetilde{K}_{ij} = (K - 1_n K - K 1_n + 1_n K 1_n)_{ij}$$

where

$$1_n \;=\; \frac{1}{n}\begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \cdots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} = \frac{1}{n} 1 1^T$$

where 1 denotes the vector of all ones.

Given a Mercer kernel K and data $X_1, X_2, \ldots, X_n$

1. Center the kernel
2. Compute $K_{ij} = K(X_i, X_j)$
3. Diagonalize K
4. Normalize eigenvector weights $\alpha^{(m)}$ so that $\langle \alpha^{(m)}, \alpha^{(m)} \rangle = \frac{1}{\lambda_m}$
5. Compute the projection of a test point x onto an eigenvector v_m by

$$\langle v_m, \Phi(x) \rangle = \sum_{i=1}^{n} \alpha_i^{(m)} K(X_i, x)$$

Just as for standard PCA, this selects components of high variance, but in the feature space of the kernel. In addition, the "feature functions"

$$f_m(x) = \langle v_m, \Phi(x) \rangle = \sum_{i=1}^{n} \alpha_i^{(m)} K(X_i, x)$$

are orthogonal and act as representative feature functions in the reproducing kernel Hilbert space of the kernel, with respect to the given data. Intuitively, these functions are smooth with respect to the RKHS norm $\|\cdot\|_K$ among all those supported on the data.

Another perspective on kernel PCA is that it is doing MDS on the kernel distances $d_{ij} = \sqrt{2(1 - K(X_i, X_j))}$; see Williams (2002).

1.4 Local Linear Embedding

Local Linear Embedding (LLE) (Roweis et al) is another nonlinear dimension rediction method. The high level idea is to obtain embeddings that **preserve local geometry** as captured by local linear regression weights.

The LLE algorithm is comprised of three steps. First, nearest neighbors are computed for each point $X_i \in \mathbb{R}^d$. Second, each point is regressed onto its neighbors, giving weights w_{ij} so that $X_i = \sum_j w_{ij} X_j$. Third, the $X_i \in \mathbb{R}^d$ are replaced by $Y_i \in \mathbb{R}^m$ where typically $m \ll d$ by solving a sparse eigenvector problem. The result is a highly nonlinear embedding, but one that is carried out by optimizations that are not prone to local minima. Underlying the procedure, as for many "manifold" methods, is a weighted sparse graph that represents the data.

Step 1: Nearest Neighbors. Here the set of the K nearest neighbors in standard Euclidean space is constructed for each data point. Using brute-force search, this requires $O(n^2 d)$ operations; more efficient algorithms are possible, in particular if *approximate* nearest neighbors are calculated. The number of neighbors K is a parameter to the algorithm, but this is the only parameter needed by LLE.

Step 2: Local weights. In this step, the local geometry of each point is characterized by a set of weights w_{ij}. The weights are computed by reconstructing each input X_i as a linear combination of its neighbors, as tabulated in Step 1. This is done by solving the least squares problem

$$\min_w \sum_{i=1}^{n} \|X_i - \sum_j w_{ij} X_j\|_2^2 \tag{1}$$

The weights w_{ij} are constrained so that $w_{ij} = 0$ if X_j is not one of the K nearest neighbors of X_i. Moreover, the weights are normalized to sum to one: $\sum_j w_{ij} = 1$, for $i = 1, \ldots, n$. This normalization ensures that the optimal weights are invariant to rotation, translation, and scaling.

Step 3: Linearization. In this step the points $X_i \in \mathbb{R}^d$ are mapped to $Y_i \in \mathbb{R}^m$, where m selected by the user, or estimated directly from the data. The vectors Y_i are chosen to minimize the reconstruction error under the local linear mappings constructed in the previous step. That is, the goal is to optimize the functional

$$\Psi(y) = \sum_{i=1}^{n} \|Y_i - \sum_j w_{ij} Y_j\|_2^2 \tag{2}$$

where the weights w_{ij} are calculated in Step 2. To obtain a unique solution, the vectors are "centered" to have mean zero and unit covariance:

$$\sum_i Y_i = 0 \quad \frac{1}{n} \sum_i Y_i Y_i^T = I_m \tag{3}$$

Carrying out this optimization is equivalent to finding the lower $(m+1)$ eigenvectors of the $n \times n$ matrix $G = (I - W)^T(I - W)$. The lowest eigenvector has eigenvalue 0, and consists of the all ones vector $(1, 1 \ldots, 1)^T$.

Locally Linear Embedding (LLE). Given n data vectors $X_i \in \mathbb{R}^d$,

1. Compute K nearest neighbors for each point;
2. Compute local reconstruction weights w_{ij} by minimizing

$$\Phi(w) = \sum_{i=1}^{n} \|X_i - \sum_j w_{ij} X_j\|^2 \quad (4)$$

$$\text{subject to} \quad \sum_j w_{ij} = 1; \quad (5)$$

3. Compute outputs $Y_i \in \mathbb{R}^m$ by computing the first m eigenvectors with nonzero eigenvalues for the $n \times n$ matrix $G = (I - W)^T(I - W)$. The reduced data matrix is $[u_1 \cdots u_m]$ where u_j are the eigenvectors corresponding to the first (smallest) nonzero eigenvalues of G.

Note that the last step assumes that the underlying graph encoded by the nearest neighbor graph is connected. Otherwise, there may be more than one eigenvector with eigenvalue zero. If the graph is disconnected, then the LLE algorithm can be run separately on each connected component. However, the recommended procedure is to choose K so that the graph is connected.

Using the simplest algorithms, the first step has time complexity $O(dn^2)$, the second step requires $O(nK^3)$ operations, and the third step, using routines for computing eigenvalues for sparse matrices, requires $O(mn^2)$ operations (and $O(n^3)$ operations in the worse case if sparsity is not exploited and the full spectrum is computed). Thus, for high dimensional problems, the first step is the most expensive. Since the third step computes eigenvectors, it shares the property with PCA that as more dimensions are added to the embedding, the previously computed coordinates do not change.

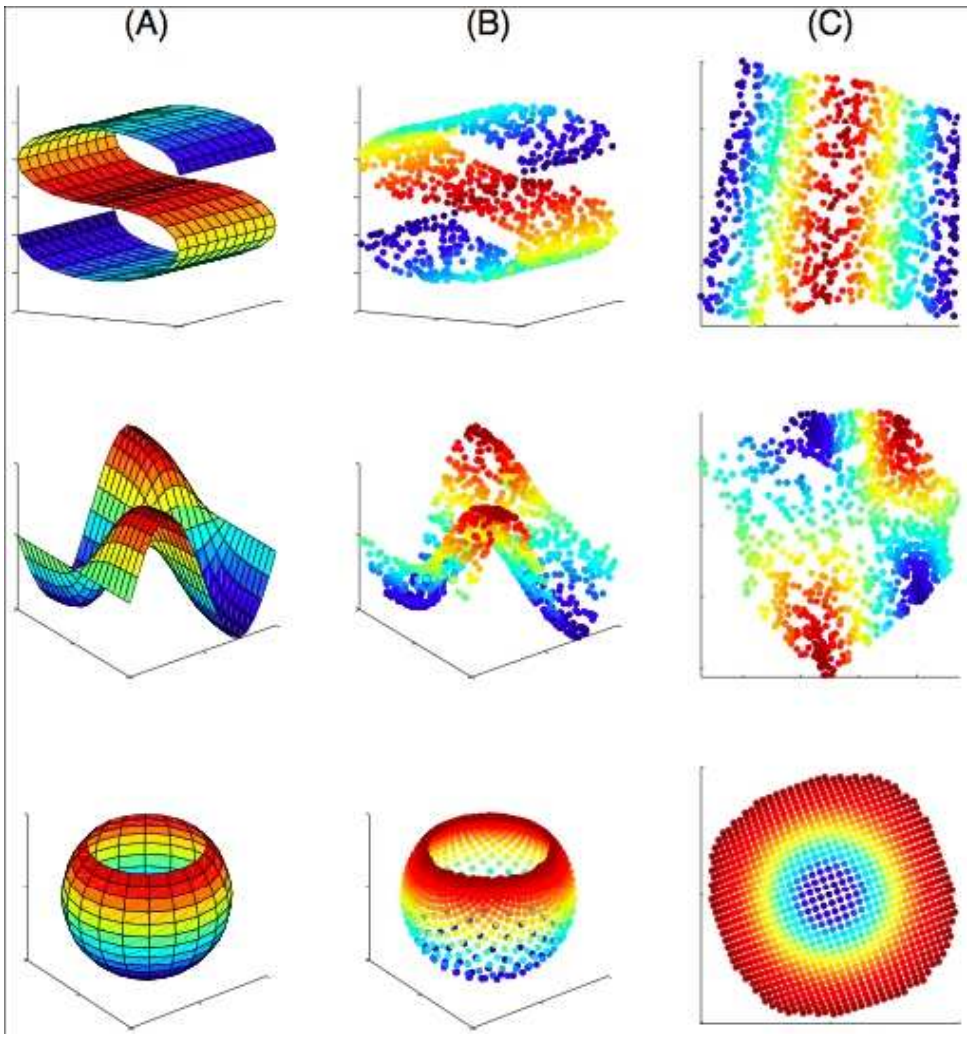

Figure 6: Each data set has $n = 1,000$ points in $d = 3$ dimensions, and LLE was run with $K = 8$ neighbors.

1.5 Isomap

Isomap is a technique that is similar to LLE, intended to provide a low dimensional "manifold" representation of a high dimensional data set. Isomap differs in how it assesses similarity between objects, and in how the low dimensional mapping is constructed.

The first step in Isomap is to constructed a graph with the nodes representing instances $X_i \in \mathbb{R}^d$ to be embedded in a low dimensional space. Standard choices are a k-nearest neighbors, and ϵ-neighborhoods. In the k-nearest neighborhood graph, each point X_i is connected to its closest k neighbors $\mathcal{N}_k(X_i)$, where distance is measured using Euclidean distance in the ambient space $\mathbb{R}^d$. In the ϵ-neighborhood graph, each point X_i is connected to all points $N_\epsilon(X_i)$ within a Euclidean ball of radius ϵ centered at X_i. The graph $G = (V, E)$ by taking edge set $V = \{x_1, \ldots, x_n\}$ and edge set

$$(u, v) \in E \text{ if } v \in \mathcal{N}(u) \text{ or } u \in \mathcal{N}(v) \tag{6}$$

Note that the node degree in these graphs may be highly variable. For simplicity, assume that the graph is connected; the parameters k or ϵ may need to be carefully selected for this to be the case.

The next step is to form a distance between points by taking path distance in the graph. That is $d(X_i, X_j)$ is the shortest path between node X_i and X_j. This distance can be computed for sparse graphs in time $O(|E| + |V| \log |V|)$. The final step is to embed the points into a low dimensional space using metric multi-dimensional scaling.

Isomap. Given n data vectors $X_i \in \mathbb{R}^d$,

1. Compute k nearest neighbors for each point, forming the nearest neighbor graph $G = (V, E)$ with vertices $\{X_i\}$.
2. Compute graph distances $d(X_i, X_j)$ using Dijkstra's algorithm
3. Embed the points into low dimensions using metric multidimensional scaling

Isomap and LLE both obtain nonlinear dimensionality reduction by mapping points into a low dimensional space, in a manner that preserves the local geometry. This local geometry will *not* be preserved by classical PCA or MDS, since far away points on the manifold will be, typically, be mapped to nearby points in the lower dimensional space.

1.6 Laplacian Eigenmaps

A similar approach is based on the use of the graph Laplacian. Recall that if $w_{ij} = K_h\left(\frac{X_i - X_j}{h}\right)$ is a weighting between pairs of points determined by a kernel K, the graph

Laplacian associated W is given by

$$L = D - W \tag{7}$$

where $D = \text{diag}(d_i)$ with $d_i = \sum_j w_{ij}$ the sum of the weights for edges emanating from node i. In Laplacian eigenmaps, the embedding is obtained using the spectral decomposition of L.

In particular, let $y_0, y_1, \ldots, y_k \in \mathbb{R}^n$ denote the first k eigenvectors corresponding to eigenvalues $0 = \lambda_0 < \lambda_1, < \lambda_2 < \cdots < \lambda_{k+1}$ of the Laplacian. This determines an embedding

$$X_i \mapsto (y_{1i}, y_{2i}, \ldots, y_{ki}) \in \mathbb{R}^k \tag{8}$$

into k dimensions.

The intuition behind this approach can be seen from the basic properties of Rayleigh quotients and Laplacians. In particular, we have that the first nonzero eigenvector satisfies

$$y_1 \;=\; \arg\min y_1^T L y_1 \;=\; \arg\min \sum_{i,j} w_{ij} \left(y_{1i} - y_{1j}\right)^2 \tag{9}$$

$$\text{such that } \; y_1^T D y_1 = 1 \tag{10}$$

Thus, the eigenvector minimizes the weighted graph L^2 norm; the intuition is that the vector changes very slowly with respect to the intrinsic geometry of the graph. This analogy is strengthened by consistency properties of the graph Laplacian. In particular, if the data lie on a Riemannian manifold M, and $f : M \to \mathbb{R}$ is a function on the manifold,

$$f^T L f \approx \int_M \|\nabla f(x)\|^2 \, d_M(x) \tag{11}$$

where on the left hand side we have evaluated the function on n points sampled uniformly from the manifold.

1.7 Diffusion Distances

As we saw when we discussed spectral clustering, there are other versions of graph Laplacians such as $D^{-1/2} W D^{-1/2}$ and $D^{-1} W$ that can have better behavior. In fact, let us consider the matrix $L = D^{-1} W$ which, as we shall now see, has a nice interpretation. We can view L as the transition matrix for a Markov chain on the data. This has a population analogue: we define the diffusion (continuous Markov chain) with transition density

$$q(y|x) = \frac{K(x,y)}{s(x)}$$

where $s(x) = \int K(x,y) dP(y)$. The stationary distribution has density $\pi(y) = s(y) / \int s(u) dP(u)$. Then L is just the discrete version of this transition probability. Suppose we run the chain

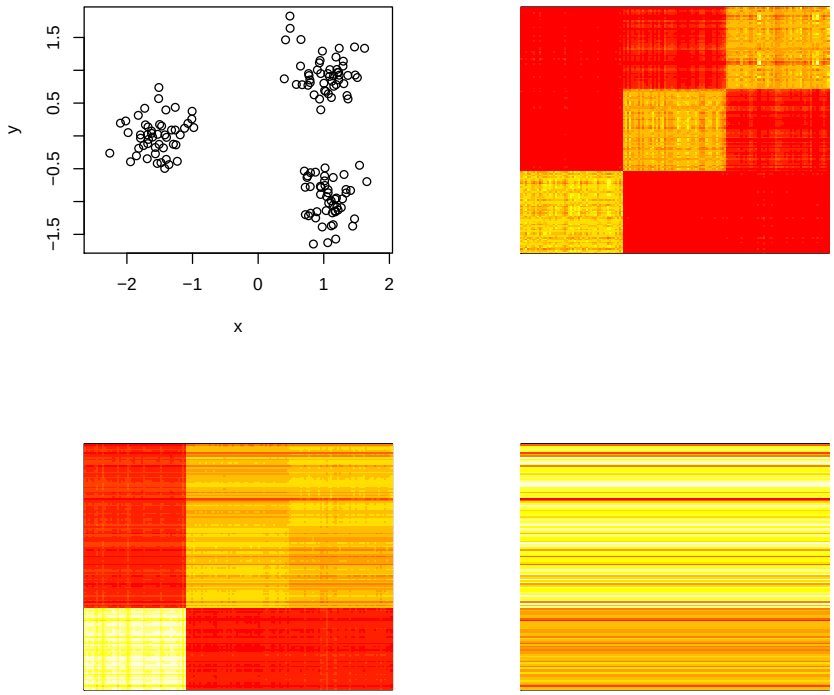

Figure 7: Diffusion maps. Top left: data. Top right: Transition matrix for $t = 1$. Botom left: Transition matrix for $t = 3$. Bottom right: Transition matrix for $t = 64$.

for t steps. The transition matrix is L^t. The properties of this matrix give information on the larger scale structure of the data (the diffusion process). We define the *diffusion distance* by

$$D_t(x, y) = \int (q_t(u|x) - q_t(u|y))^2 \frac{p(u)}{\pi(u)} du$$

which is a measure of how har it is to get from x to y in t steps (Coifman and Lafon, 2006). It can be shown that

$$D_t(x, y) = \sqrt{\sum_j \lambda_j^{2t} (\psi_j(x) - \psi_j(y))^2}$$

where λ_j and ψ_j are the eigenvalues and eigenvectors of q. We can now reduce the dimension of the data by applying MDS to $D_t(x, y)$. Alternatively, they suggest mapping a point x to

$$\Psi_t(x) = (\lambda_1^t \psi_1(x), \ldots, \pi_l^t \psi_k(x))$$

for some k. An example is shown in Figure 7.

1.8 Principal Curves and Manifolds

A nonparametric generalization of principal components is **principal manifolds**. The idea is to replace linear subspaces with more general manifolds. There are many approaches. We will consider an approach due to Smola et al (2001). Ridge estimation, which will see shortly, might be a better way to do this.

Let $X \in \mathbb{R}^d$ and let $\mathcal{F}$ be a set of functions from $[0,1]^k$ to $\mathbb{R}^d$. The principal manifold (or principal curve) is the function $f \in \mathcal{F}$ that minimizes

$$R(f) = \mathbb{E}\left(\min_{z \in [0,1]^k} \|X - f(z)\|^2 \right). \tag{12}$$

To see how general this is, note that we recover principal components as a special case by taking $\mathcal{F}$ to be linear mappings. We recover k-means by taking $\mathcal{F}$ to be all mappings from $\{1, \ldots, k\}$ to $\mathbb{R}^d$.

But our focus in this section is on smooth curves. We will take

$$\mathcal{F} = \left\{ f : \ \|f\|_k^2 \leq C^2 \right\}$$

where $\|f\|_K$ is the norm for a reproducing kernel Hilbert space (RKHS) with kernel K. A common choice is the Gaussian kernel

$$K(z,u) = \exp\left\{ -\frac{\|z-u\|^2}{2h^2} \right\}.$$

To approximate the minimizer, we can proceed as in (Smola, Mika, Schölkopf, Williamson 2001). Fix a large number of points $z_1, \ldots, z_M$ and approximate an arbitrary $f \in \mathcal{F}$ as

$$f(z) = \sum_{j=1}^{M} \alpha_j K(z_j, z)$$

which depends on parameters $\alpha = (\alpha_1, \ldots, \alpha_M)$, where $\alpha_i \in \mathbb{R}^d$. The minimizer can be found as follows. Define latent variables $\xi = (\xi_1, \ldots, \xi_n)$ where $\xi_i \in \mathbb{R}^k$ and

$$\xi_i = \text{argmin}_{\xi \in [0,1]^k} \|X_i - f(\xi)\|^2.$$

For fixed α we find each ξ_i by standard nonlinear function minimization. Given ξ we then find α by minimizing

$$\frac{1}{n} \sum_{i=1}^{n} \|X_i - \sum_{j=1}^{M} \alpha_j K(z_j, \xi_i)\|^2 + \frac{\lambda}{2} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i \alpha_j \, K(z_i, z_j).$$

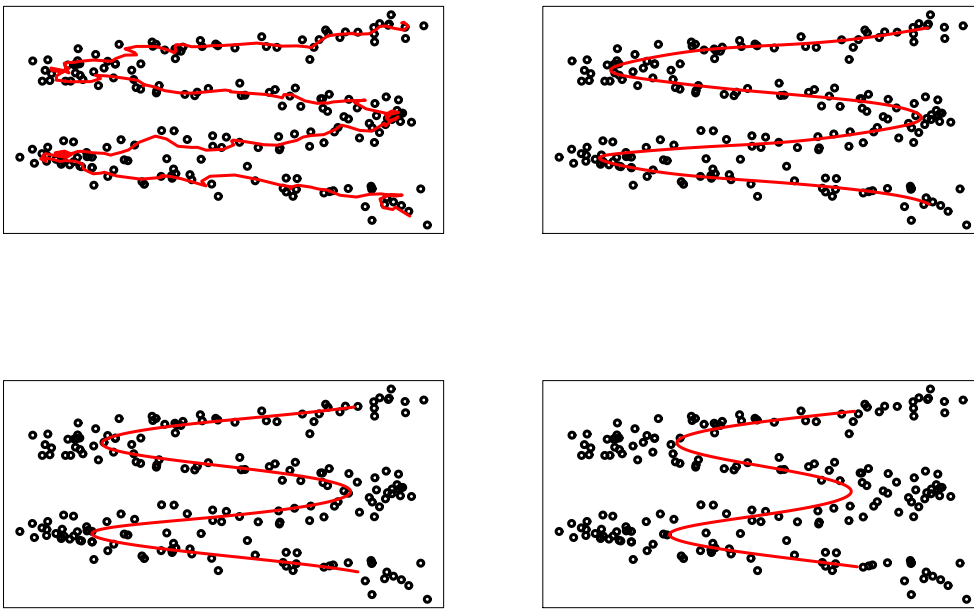

Figure 8: Principal curve with increasing amounts of regularization.

The minimizer is

$$\alpha = \left(\frac{\lambda n}{2} K_z + K_\xi^T K_\xi \right)^{-1} K_\xi^T X$$

where $(K_z)_{ij} = K(z_i, z_j)$ is $M \times M$ and $(K_\xi)_{ij} = K(\xi_i, z_j)$ is $n \times M$. Now we iterate, alternately solving for ξ and α.

Example 6 *Figure 8 shows some data and four principal curves based on increasing degrees of regularization.*

1.9 Random Projections: Part I

A simple method for reducing the dimension is to do a random projection. Surprisingly, this can actually preserve pairwise distances. This fact is known as the Johnson-Lindenstrauss Lemma, and this section is devoted to an elementary proof of this result.[1]

Let $X_1, \ldots, X_n$ be a dataset with $X_i \in \mathbb{R}^d$. Let S be a $m \times d$ matrix filled with iid $N(0,1)$ entries, where $m < d$. Define

$$L(x) = \frac{Sx}{\sqrt{m}}.$$

[1]In this section and the next, we follow some lecture notes by Martin Wainwright.

The matrix S is called a *sketching matrix*. Define $Y_i = L(X_i)$ and note that $Y_i \in \mathbb{R}^m$. The projected dataset $Y_1, \ldots, Y_n$ is lower dimensional.

Theorem 7 (Johnson-Lindenstrauss) *Fix $\epsilon > 0$. Let $m \geq 32 \log n/\epsilon^2$. Then, with probability at least $1 - e^{-m\epsilon^2/16} \geq 1 - (1/n)^2$, we have*

$$(1-\epsilon)||X_i - X_j||^2 \leq ||Y_i - Y_j||^2 \leq (1+\epsilon)||X_i - X_j||^2 \tag{13}$$

for all i, j.

Notice that the embedding dimension m, does not depend on the original dimension d.

Proof. For any $j \neq k$,

$$\frac{||Y_j - Y_k||^2}{||X_i - X_j||^2} - 1 = \frac{||S(X_j - X_k)||^2}{m||X_i - X_j||^2} - 1 = \frac{1}{m}\sum_{i=1}^m Z_i^2 - 1$$

where

$$Z_i = \left\langle S_i, \frac{X_j - X_k}{||X_j - X_k||} \right\rangle$$

where S_i is the i^{th} row of S. Note that $Z_i \sim N(0,1)$ and so $Z_i^2 \sim \chi_1^2$ and $\mathbb{E}[Z_i^2] = 1$. The moment generating function of Z_i^2 is $m(\lambda) = (1-2\lambda)^{-1/2}$ (for $\lambda < 1/2$). So, for $\lambda > 0$ small enough,

$$\mathbb{E}[e^{\lambda(Z_i^2 - 1)}] = \frac{e^{-\lambda}}{\sqrt{1-2\lambda}} \leq e^{2\lambda^2}.$$

Hence,

$$\mathbb{E}\left[\exp\left(\lambda \sum_i (Z_i^2 - 1)\right)\right] \leq e^{2m\lambda^2}.$$

Thus

$$\begin{aligned}
\mathbb{P}\left(\frac{1}{m}\sum_{i=1}^m Z_i^2 - 1 \geq \epsilon\right) &= \mathbb{P}\left(e^{\lambda\sum_{i=1}^m Z_i^2 - 1} \geq e^{\lambda m\epsilon}\right) \\
&\leq e^{-\lambda m\epsilon}\mathbb{E}\left(e^{\lambda\sum_{i=1}^m Z_i^2 - 1}\right) \leq e^{2m\lambda^2 - m\epsilon\lambda} \\
&\leq e^{-m\epsilon^2/8}
\end{aligned}$$

where, in the last step, we chose $\lambda = \epsilon/4$. By a similar argument, we can bound $\mathbb{P}\left(\frac{1}{m}\sum_{i=1}^m Z_i^2 - 1 \leq -\epsilon\right)$. Hence,

$$\mathbb{P}\left(\left|\frac{||S(X_j - X_k)||^2}{m||X_j - X_k||^2} - 1\right| \geq \epsilon\right) \leq 2e^{-m\epsilon^2/8}.$$

By the union bound, the probability that (13) fauls for some pair is at most

$$n^2 2e^{-m\epsilon^2/8} \leq e^{-m\epsilon^2/16}$$

where we used the fact that $m \geq 32 \log n/\epsilon^2$. □

1.10 Random Projections: Part II

The key to the Johnson-Lindenstrauss (JL) theorem was applying concentration of measure to the quantity

$$\Gamma(\mathcal{K}) = \sup_{u \in \mathcal{K}} \left| \frac{||Su||^2}{m} - 1 \right|$$

where

$$\mathcal{K} = \left\{ \frac{X_j - X_k}{||X_j - X_k||} : \ j \neq k \right\}.$$

Note that $\mathcal{K}$ is a subset of the sphere $\mathcal{S}^{d-1}$.

We can generalize this to other subsets of the sphere. For example, suppose that we take $\mathcal{K} = \mathcal{S}^{d-1}$. Let $\widehat{\Sigma} = m^{-1} S^T S$. Note that each row of S_i has mean 0 and variance matrix I and $\widehat{\Sigma}$ is the estimate of the covaraince matrix. Then

$$\begin{aligned} \sup_{u \in \mathcal{K}} \left| \frac{||Su||^2}{m} - 1 \right| &= \sup_{||u||=1} \left| \frac{||Su||^2}{m} - 1 \right| \\ &= \sup_{||u||=1} \left| u^T (m^{-1} S^T S - I) u \right| = ||\widehat{\Sigma} - I|| \end{aligned}$$

which is the operator norm of the difference between the sample covariance and true covariance.

Now consider least squares. Suppose we want to minimize $||Y - X\beta||^2$ where Y is an $n \times 1$ vector and X is a $n \times d$ matrix. If n is large, this may be expensive. We could try to approximate the solution by minimizing $||S(Y - X\beta)||^2$. Since the least squares solution lies in the column space of X, to show that projection via S is a reasonable approximation, suggests taking

$$\mathcal{K} = \left\{ u \in \mathcal{S}^{d-1} : \ u = Xv \text{ for some } v \in \mathbb{R}^d \right\}.$$

It can be shown that, if $\Gamma(\mathcal{K})$ is small, then the solution to the reduced problem approximates the original problem.

How can we bound $\Gamma(\mathcal{K})$? To answer this, we use the *Gaussian width* which is defined by

$$W(\mathcal{K}) = \mathbb{E}\left[\sup_{u \in \mathcal{K}} \langle u, Z \rangle \right]$$

where $Z \sim N(0, I)$ and I is the $d \times d$ identity matrix.

Theorem 8 *Let S be a $m \times d$ Gaussian projection matrix. Let $\mathcal{K}$ be any subset of the sphere and suppose that $m \geq W^2(\mathcal{K})$. Then, for any $\epsilon \in (0, 1/2)$,*

$$\mathbb{P}\left(\Gamma(\mathcal{K}) \geq 4 \left(\frac{W(\mathcal{K})}{\sqrt{m}} + \epsilon \right) \right) \leq 2e^{-m\epsilon^2/2}.$$

In particular, if $m \geq W^2(\mathcal{K})/\delta^2$, then $\Gamma(\mathcal{K}) \leq 8\delta$ with high probability.

Let us return to the JL theorem. In this case,

$$\mathcal{K} = \left\{ \frac{X_j - X_k}{||X_j - X_k||} : \ j \neq k \right\}.$$

In this case $\mathcal{K}$ is finite. The number of elements is $N = \binom{n}{2}$. Note that $\log N \leq 2 \log n$. Since the set is finite, we know from our previous results on expectations of maxima, that

$$W(\mathcal{K}) \leq \sqrt{2 \log N} \leq \sqrt{4 \log n}.$$

According to the above theorem, we need to take $m \geq W^2/\delta^2 \succeq \log n/\delta^2$ which agrees with the JL theorem.

The proof of the theorem is quite long but it basically uses concentration of measure arguments to control the maximum fluctuations as u varies over $\mathcal{K}$. When applied to least squares, if we want to approximate $||Y - X\beta||^2$ it turns out that the Gaussian width has constant order. But if we are interested in approximating $\widehat{\beta}$, we need to set $\delta \approx 1/\sqrt{n}$, so that we need $m \approx n$ which is not useful. However, there is an improvement that uses iterative sketching that only requires $m = O(\log n)$ observations. A good reference is:

M. Pilanci and M. J. Wainwright. Iterative Hessian Sketch: Fast and accurate solution approximation for constrained least-squares. arXiv:1411.0347.

2 Estimating Low Dimensional Structure

Let $Y_1, \ldots, Y_n \sim P$. We can think of the structure we are looking for as a function of P. Examples of such functions include:

$T(P)$ = the support of P
$T(P)$ = ridges of the density p
$T(P)$ = dimension of the support
$T(P)$ = DTM (distance to a measure)
$T(P)$ = persistent homology of DTM.

A common example is when the support of P is a manifold M. In that case, we define the minimax risk

$$R_n = \inf_{\widehat{M}} \sup_{P \in \mathcal{P}} \mathbb{E}_P[H(\widehat{M}, M(P))]$$

where H is the Hausdorff distance:

$$H(A, B) = \inf\{\epsilon : A \subset B \oplus \epsilon \text{ and } B \subset A \oplus \epsilon\}$$

and

$$A \oplus \epsilon = \bigcup_{x \in A} B(x, \epsilon).$$

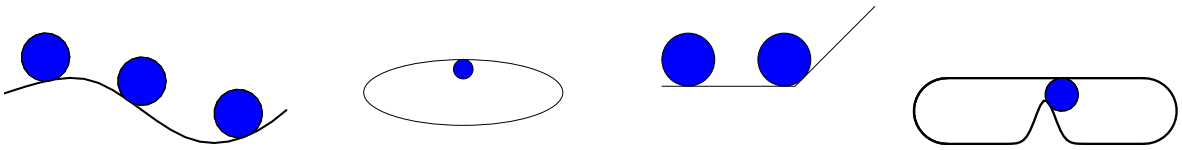

Figure 9: *First two plots: a ball of raidus $r < \kappa$ rolls freely. Third plot: ball cannot roll because reach is 0. Fourth: ball cannot roll because $r > \kappa$.*

2.1 Manifolds

A common starting place is to assume that P is supported on a manifold M. This is usually a bogus assumption. More realistically, the data might be concentrated near a low dimensional structure. Assuming that the structure is smooth and that the support is exactly on this structure is unrealistic. But it is a starting place. So, for now, assume that $Y_i \in \mathbb{R}^D$ and that P is supported on a manifold M of dimension $d < D$.

Just as we needed some conditions on a density function or regression function to estimate it, we needed a condition on a manifold to estimate it. The most common condition is that M has positive reach. The *reach* of a manifold M is the largest r such that $d(x, M) \leq r$ implies that x has a unique projection onto M. This is also called the thickness or condition number of the manifold; see Niyoki, Smale, and Weinberger (2009). Intuitively, a manifold M with reach(M) $= \kappa$ has two constraints:

1. Curvature. A ball or radius $r \leq \kappa$ can roll freely and smoothly over M, but a ball or radius $r > \kappa$ cannot.
2. Separation. M is at least 2κ from self-intersecting.

See Figure 9. Also, normal vectors of length less than κ will not cross. See Figure 10.

The easiest way to estimate a d-manifold embedded in $\mathbb{R}^D$ is just to estimate the support of P. For example, the Devroye-Wise (1980) estimator is

$$\widehat{M} = \bigcup_i B(Y_i, \epsilon).$$

Choosing $\epsilon_n \asymp (1/n)^{1/D}$ we get

$$\mathbb{E}[H(\widehat{M}, M)] \leq \left(\frac{C \log n}{n}\right)^{\frac{1}{D}}.$$

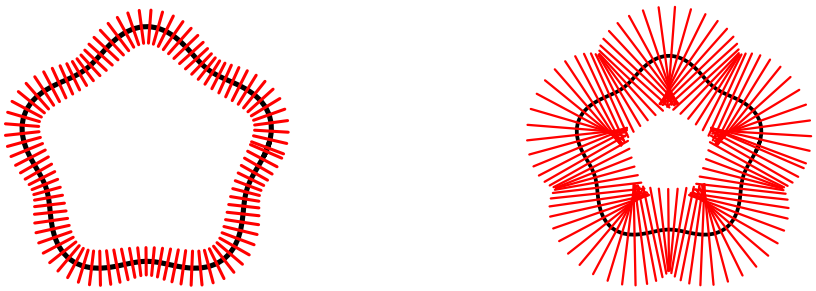

Figure 10: Left: Normal vectors of length $r < \kappa$ don't cross. Right: Normal vectors of length $r > \kappa$ do cross.

This estimator is simple but sub-optimal. Note that the rate depends on the ambient dimension.

Let $Y_1, \ldots, Y_n \sim P$ where

$$Y_i = \xi_i + Z_i$$

where $Y_i \in \mathbb{R}^D$, $\xi_1, \ldots, \xi_n \sim G$ where G is uniform on a d-manifold M and the noise Z_i is perpendicular to M (uniform on the normals). It's a weird model but it was used in Niyogi, Smale, Weinberger (2008). Let $\mathcal{P}$ be the set of distributions with bounded density on d-manifolds with reach at least κ. Then (GPVW 2011)

$$R_n = c\left(\frac{\log n}{n}\right)^{\frac{2}{2+d}}.$$

Thus the rate depends on d not D. I don't know a practical estimator to achieve this rate.

Now suppose that

$$Y_1, \ldots, Y_n \sim (1-\pi)U + \pi G$$

where G is supported on M, $0 < \pi \leq 1$, U is uniform on a compact set $\mathcal{K} \subset \mathbb{R}^D$. Then (GPVW 2012)

$$R_n \asymp \left(\frac{1}{n}\right)^{\frac{2}{d}}.$$

A more realistic model is $Y_i = X_i + Z_i$ where $X_1, \ldots, X_n \sim G$ and and $Z_i \sim N(0, \sigma^2 I_D)$. Then

$$\frac{1}{\log n} \leq R_n \leq \frac{1}{\sqrt{\log n}}.$$

This means that, with additive noise, the problem is hopeless.

On solution is to give up on estimating M and instead estimate some approximation to M. This is the next topic.

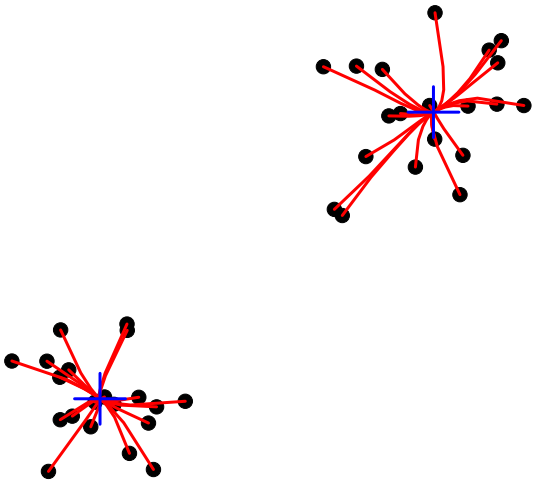

Figure 11: The mean shift algorithm.

2.2 Ridges

A ridge is a high-density, low dimensional structure. A 0-dimensional ridge is just a mode. In this case

$$\nabla p(x) = 0 \quad \text{and} \quad \lambda_{\max}(H(x)) < 0$$

where H is the Hessian. (assuming p is Morse). Recall that a mode can also be thought of as the destination of a gradient ascent path, π_x: i.e.

$$m = \lim_{t \to \infty} \pi_x(t)$$

where

$$\pi_x'(t) = \nabla p(\pi_x(t)).$$

The modes of p can be found by the mean-shift algorithm as in Figure 11.

Higher dimensional ridges can be defined as the zeros of a projected gradient. Think of the ridge of a mountain. The left plot in Figure 12 shows a density with a sharp, one-dimensional ridge. The right plot show the underlying manifold, the ridge, and the ridge of the smoothed density.

To define the ridge formally, let p be a density with gradient g and Hessian H. Denote the eigenvalues of $H(x)$ by

$$\lambda_1(x) \geq \lambda_2(x) \geq \cdots \geq \lambda_d(x) \geq \lambda_{d+1}(x) \geq \cdots \geq \lambda_D(x).$$

Let $U(x) = [W(x) : V(x)]$ be the matrix of eigenvectors. Then $L(x) = V(x)V^T(x)$ is the projector onto the local tangent space. Define the projected gradient $G(x) = L(x)g(x)$.

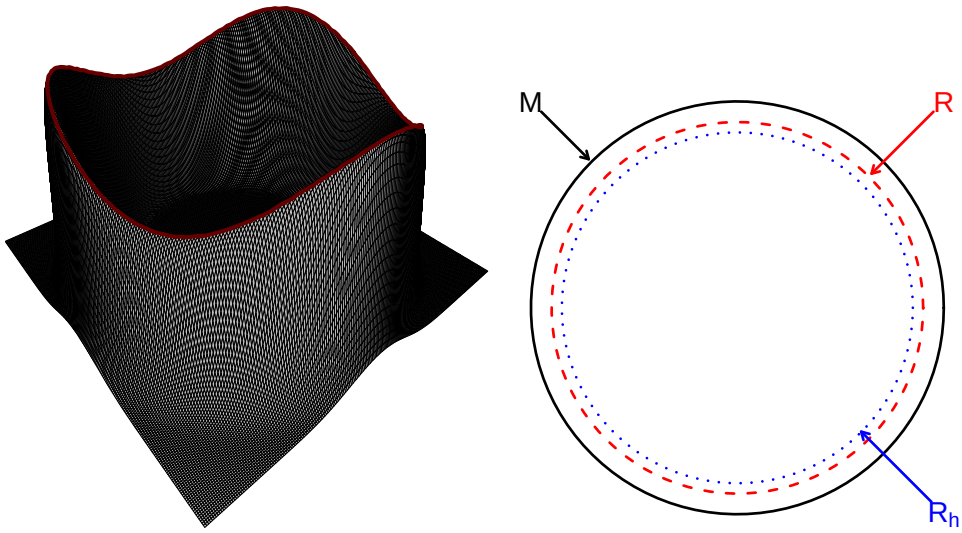

Figure 12: *Left: The one-dimensional ridge of a density. Right: the manifold, the ridge of the density p, and the ridge of the smoothed density* $p \star K_h$.

Finally. define the ridge by

$$R(p) = \left\{ x : \ \lambda_{d+1}(x) < 0 \quad \text{and} \quad G(x) = 0 \right\}.$$

Several other definitions of a ridge have been proposed in the literature; see Eberly (1996). The one we use has several useful properties: if $\widehat{p}$ is close to p then $R(\widehat{p})$ is close in Hausdorff distance to $R(p)$.

And, there is an algorithm to find the ridge: the subspace-constrained mean-shift algorithm (SCMS, Ozertem and Erdogmus 2011). (The usual mean-shift algorithm with a projection step.)

To estimate $R(p)$, estimate the density, its gradient, and its Hessian:

$$\widehat{p}(y) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h^D} K\left(\frac{y - Y_i}{h}\right)$$

$\widehat{g}$ = gradient of $\widehat{p}$ and $\widehat{H}$ = Hessian of $\widehat{p}$. Denoising: remove low density points. Apply the SCMS algorithm.

$\widehat{R}$ is a consistent estimator of R and:

$$H(R, \widehat{R}) = O_P\left(n^{-\frac{2}{8+D}}\right)$$

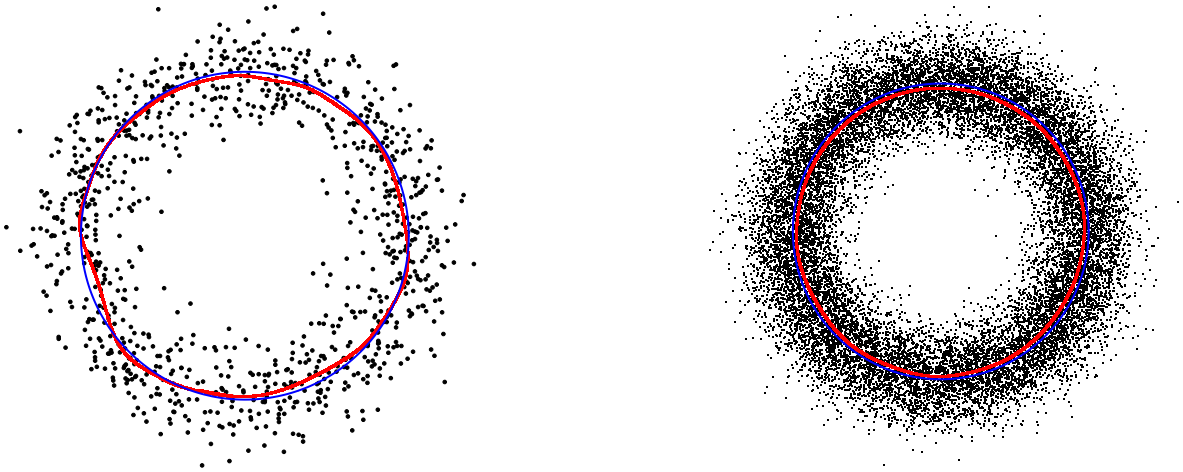

Figure 13: *Left: Manifold in blue. Estimated ridge in red. Right: sample example with more data.*

For fixed bandwidth h (which still captures the shape),

$$H(R_h, \widehat{R}_h) = O_P\left(\sqrt{\frac{\log n}{n}}\right)$$

and $\widehat{R}_h$ is (nearly) homotopic to R_h. See Figures 13 and 14 for examples. A real example is shown in 15 (from Chen, Ho, Freeman, Genovese and Wasserman: arXiv:1501.05303).

How to choose a good bandwidth h is not clear. Figure 16 shows that the ridge is fairly stable as we decrease h until we reach a phase transition where the ridge falls apart.

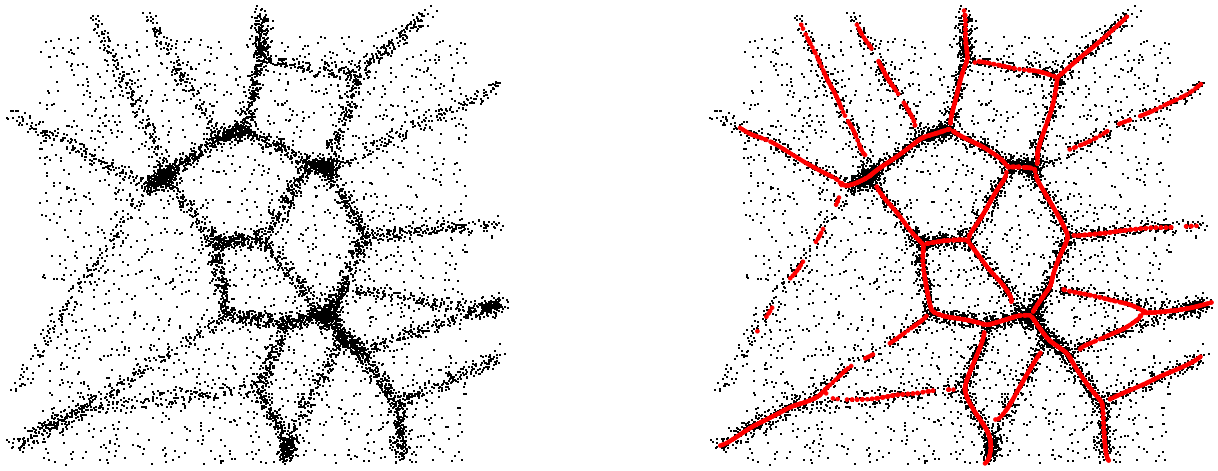

Figure 14: *Left: data. Right: SCMS output.*

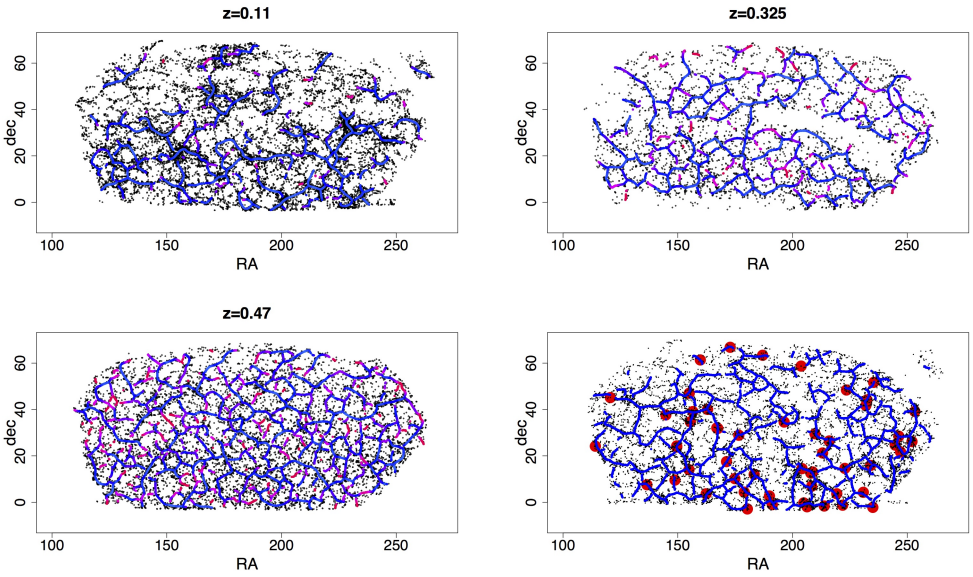

Figure 15: *Galaxy data from the Sloan Digital Sky Survey at three different redshifts. The fourth plot shows known galaxy clusters. From: Chen, Ho, Freeman, Genovese and Wasserman: arXiv:1501.05303*

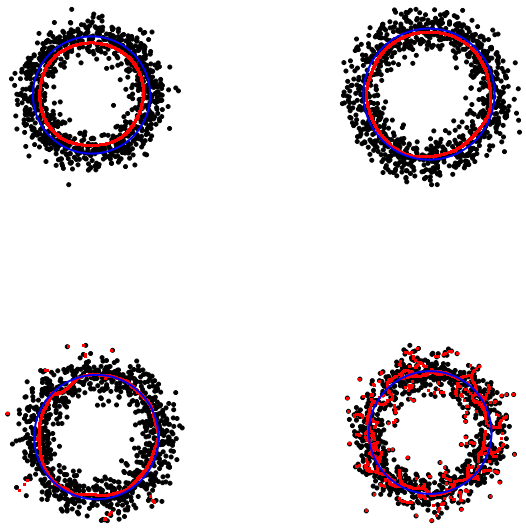

Figure 16: *As we decrease the bandwidth, the ridge is quite stable. Eventually we reach a phase transition where the estimated ridge falls apart.*