Representation Learning
10716: Advanced Machine Learning
Pradeep Ravikumar

1 Spectral Clustering

We have looked at approaches to extract clusters, including density based approaches that estimated disconnected components after removing low density regions (or points). Suppose we are not interested in clusters per se, but lower-dimensional features that "respect" the clusters. What does that mean? A natural requirement might be that features of points within a cluster are very similar to each other, and features of points in different clusters are far.If the features are exactly the same for points within a cluster, then we could simply read off the cluster labels from the features. And if they are mostly the same, then we can do so after a simple partitional clustering of the features. One approach to do so is "spectral clustering," which refers to a class of representation learning methods that use eigenvectors of appropriately constructed similarity matrices. An excellent tutorial on spectral clustering, and on which this section heavily relies on, is von Luxburg (2006). More detail can be found in Chung (1997). The reason it's called spectral clustering is that these spectral features can be used to extract clusters.

Let G be an undirected graph with n vertices. Typically these vertices correspond to observations $X_1, \ldots, X_n$. Let W be an $n \times n$ symmetric weight matrix. Say that X_i and X_j are connected if $W_{ij} > 0$. The simplest type of weight matrix has entries that are either 0 or 1. For example, we could define

$$W_{ij} = I(||X_i - X_j|| \leq \epsilon).$$

An example of a more general weight matrix is $W_{ij} = e^{-||X_i - X_j||^2/(2h^2)}$.

The degree matrix D is the $n \times n$ diagonal matrix with $D_{ii} = \sum_{j=1}^n W_{ij}$. The graph Laplacian is

$$L = D - W. \tag{1}$$

The graph Laplacian has many interesting properties which we list in the following result. Recall that a vector v is an eigenvector of L if there is a scalar λ such that $Lv = \lambda v$ in which case we say that λ is the eigenvalue corresponding to v.

Theorem 1 *The graph Laplacian L has the following properties:*

1. *For any vector $f = (f_1, \ldots, f_n)^T$,*

$$f^T L f = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n W_{ij}(f_i - f_j)^2.$$

2. *L is symmetric and positive semi-definite.*

3. *The smallest eigenvalue of L is 0. When there is a single smallest eigenvalue, the corresponding eigenvector is $(1, 1, \ldots, 1)^T$.*

4. *L has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_k$.*

5. *The number of eigenvalues that are equal to 0 is equal to the number of connected components of G. That is, $0 = \lambda_1 = \ldots = \lambda_k$ where k is the number of connected components of G. The corresponding eigenvectors $v_1, \ldots, v_k$ are orthogonal and each is constant over one of the connected components of the graph.*

Part 1 of the theorem says that L is like a derivative operator. The last part shows that we can use the graph Laplacian to find the connected components of the graph.

Proof.

(1) This follows from direct algebra.

(2) Since W and D are symmetric, it follow that L is symmetric. The fact that L is positive semi-definite folows from part (1).

(3) Let $v = (1, \ldots, 1)^T$. Then

$$Lv = Dv - Wv = \begin{pmatrix} D_{11} \\ \vdots \\ D_{nn} \end{pmatrix} - \begin{pmatrix} D_{11} \\ \vdots \\ D_{nn} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

which equals $0 \times v$.

(4) This follows from parts (1)-(3).

(5) First suppose that $k = 1$ and thus that the graph is connected. We already know that $\lambda_1 = 0$ and $v_1 = (1, \ldots, 1)^T$. Suppose there were another eigenvector v with eigenvalue 0. Then

$$0 = v^T L v = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij}(v(i) - v(j))^2.$$

It follows that $W_{ij}(v(i) - v(j))^2 = 0$ for all i and j. Since G is connected, for any pair of nodes i, j, there is a path $i_0 = i, i_1, \ldots, i_m = j$, such that $W_{i_\ell i_{\ell+1}} > 0$, which entails that:

$v(i_\ell) = v(i_{\ell'})$ for all nodes in the path, and consequently that $v_i = v_j$. Since this holds for all i, j, v is constant, and lies in the span of v_1.

Now suppose that the graph G has k components. Let n_j be the number of nodes in components j. We can relabel the vertices so that the first n_1 nodes correspond to the first connected component, the second n_2 nodes correspond to the second connected component and so on. Let $v_1 = (1, \ldots, 1, 0, \ldots, 0)$ where the 1's correspond to the first component. Let Let $v_2 = (0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$ where the 1's correspond to the second component. Define $v_3, \ldots, v_k$ similarly. Due to the re-ordering of the vertices, L has block diagonal form:

$$L = \begin{pmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{pmatrix}.$$

Here, each L_i corresponds to one of the connected components of the graph. It is easy to see that for $j = 1, \ldots, k$, $L\, v_j = 0$ so that each v_j is an eigenvector with zero eigenvalue. Suppose that v is any eigenvector with 0 eigenvalue. Arguing as before, v must be constant over each component, so that it lies in the span of $\{v_j\}_{j=1}^k$.

□

Let's collect these into a $\mathbb{R}^{n \times k}$ matrix

$$V = \begin{pmatrix} \mathbf{1} & & & \\ & \mathbf{1} & & \\ & & \ddots & \\ & & & \mathbf{1} \end{pmatrix}.$$

Note that since zero is a repeated eigenvalue, we need not get exactly the vector of ones as above, but could get set of k orthogonal vectors that span the above subspace. Specifically consider the matrix $V \in \mathbb{R}^{n \times k}$ defined as:

$$V = \begin{pmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \ddots \\ \mathbf{v_k} \end{pmatrix} = \begin{pmatrix} \mathbf{1} & & & \\ & \mathbf{1} & & \\ & & \ddots & \\ & & & \mathbf{1} \end{pmatrix} R,$$

where R is some rotation matrix, $\mathbf{v_i} \in \mathbb{R}^{n_i \times k}$ is the sub-matrix with n_i rows viz. number of samples in i-th connected comoonent, and $\mathbf{v_i}$ is some vector $v_i \in \mathbb{R}^k$ repeated over n_i rows. In general if we have not sorted the matrix as above, how will the matrix V look like? Even though there are n rows, there are only k distinct vectors, $v_1, \ldots, v_k$.

When there are distinct connected components, we can then take the clusters to be the connected components of the graph, which can be found by getting the eigenvectors of the

Laplacian L as discussed above. This is also called geometric graph clustering. Thus one perspective of spectral clustering is a new algorithm to find the connected components of the graph. In other words, if I get the k eigenvectors collated as a matrix $V \in \mathbb{R}^{n \times k}$ with eigenvalue zero, then

Example 2 *Consider the graph*

and suppose that $W_{ij} = 1$ if and only if there is an edge between X_i and X_j. Then

$$W = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and the Laplacian is

$$L = D - W = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 0 \end{pmatrix}.$$

The eigenvalues of W, from smallest to largest are $0, 0, 1, 2, 3$. The eigenvectors are

$$v_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad v_3 = \begin{pmatrix} 0 \\ 0 \\ -.71 \\ 0 \\ .71 \end{pmatrix} \quad v_4 = \begin{pmatrix} -.71 \\ .71 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad v_5 = \begin{pmatrix} 0 \\ 0 \\ -.41 \\ .82 \\ -.41 \end{pmatrix}$$

Note that the first two eigenvectors correspond to the connected components of the graph.

How do we get a weighted adajency matrix/graph? One approach is to compute the near neighbor graph as:

$$W_{ij} = I(||X_i - X_j|| \leq \epsilon)$$

for some $\epsilon > 0$. Another, and the most commonly used set of weights for this purpose are

$$W_{ij} = e^{-||X_i - X_j||^2/(2h^2)}.$$

Other kernels $K_h(X_i, X_j)$ can be used as well.

for each point $i \in [n]$, I can assign it to the cluster $j \in [k]$ s.t. V_j

But more generally, what if the data does not consist of separate disconnected components, but instead consists of "clusters" that are only loosely connected i.e. very few edges between clusters. In such a case, the eigenvectors might not correspond exactly to indicator functions of the clusters. (Ng, Jordan, Weiss 2001) show that in such a case, by appealing to matrix perturbation theory, one can show that the bottom k eigenvectors of the Laplacian can be shown to be close to indicator functions of clusters. Now with most procedures to recover the first k eigenvectors, we usually have stronger guarantees on recovering the subspace spanned by these eignvectors, so that we are not guaranteed to get indicator vectors per se, but rotations of these i.e. some set of k orthonormal vectors. And in the case where the clusters are not fully disconnected, we will likely have each data point "cluster" around its cluster's orthonormal vector.

This thus suggests using the Laplacian to transform the data into a new coordinate system in which clusters are easier to find.

We define the **symmetrized Laplacian** $\mathcal{L} = D^{-1/2} W D^{-1/2}$ and the **random walk Laplacian** $\mathcal{L} = D^{-1} W$. These are all very similar.

We can then use the Laplacian to transform the data $\{X_i\}$ to within into a new coordinate system. Denoting the transformed data as $\{\widehat{X}_i\}$, in this new coordinate system if $\widehat{X}_i$ and $\widehat{X}_j$ are close in Euclidean distance, then they are part of the same cluster in the data i.e. connected by many high density paths through the data.

The steps are:

Input: $n \times n$ similarity matrix W.

1. Let D be the $n \times n$ diagonal matrix with $D_{ii} = \sum_j W_{ij}$.
2. Compute the Laplacian $\mathcal{L} = D^{-1} W$.
3. Find first k eigenvectors $v_1, \ldots, v_k$ of $\mathcal{L}$.
4. Project each X_i onto the eigenvectors to get new points $\widehat{X}_i = (\sqrt{\lambda_j}\, v_j(i))_{j \in [k]}$.
5. Cluster the points $\widehat{X}_1, \ldots, \widehat{X}_n$ using any standard clustering algorithm.

The numbers h (bandwidth for kernel) and k (number of eigenvectors) are tuning parameters. The hope is that clusters are easier to find in the new parameterization.

Example 3 *Figure 1 shows a simple synthetic example. The top left plot shows the data. We apply spectral clustering with Gaussian weights and bandwidth $h = 3$. The top middle plot shows the first 20 eigenvalues. The top right plot shows the the first versus the second eigenvector. The two clusters are clearly separated. (Because the clusters are so separated, the graph is essentially disconnected and the first eigenvector is not constant. For large h,*

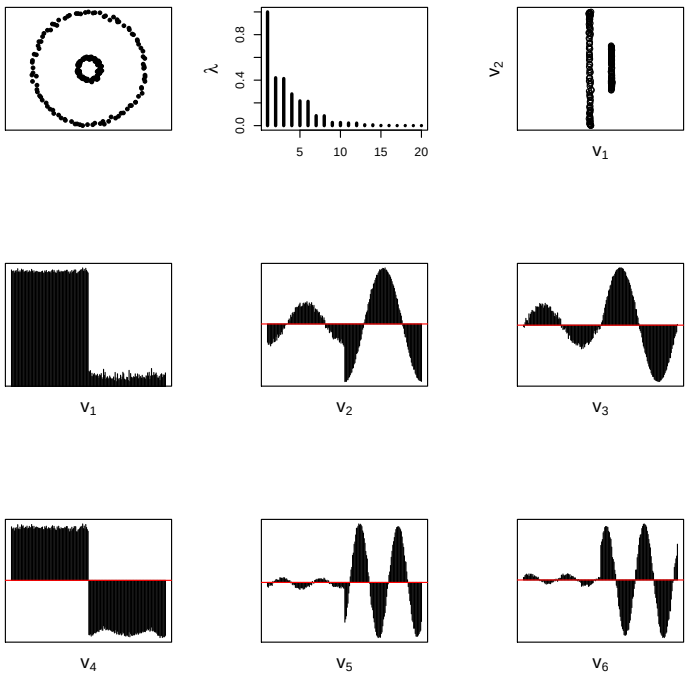

Figure 1: Top left: data. Top middle: eigenvalues. Top right: second versus third eigenvectors. Remaining plots: first six eigenvectors.

the graph becomes fully connected and v_1 is then constant.) The remaining six plots show the first six eigenvectors. We see that they form a Fourier-like basis within each cluster. Of course, single linkage clustering would work just as well with the original data as in the transformed data. The real advantage would come if the original data were high dimensional.

Example 4 *Figure 2 shows a spectral analysis of some zipcode data. Each datapoint is a 16 x 16 image of a handwritten number. We restrict ourselves to the digits 1, 2 and 3. We use Gaussian weights and the top plots correspond to $h = 6$ while the bottom plots correspond to $h = 4$. The left plots show the first 20 eigenvalues. The right plots show a scatterplot of the second versus the third eigenvector. The three colors correspond to the three digits. We see that with a good choice of h, namely $h = 6$, we can clearly see the digits in the plot. The original dimension of the problem is 16 x 16 =256. That is, each image can be represented by a point in $\mathbb{R}^{256}$. However, the spectral method shows that most of the information is captured by two eignvectors so the effective dimension is 2. This example also shows that the choice of h is crucial.*

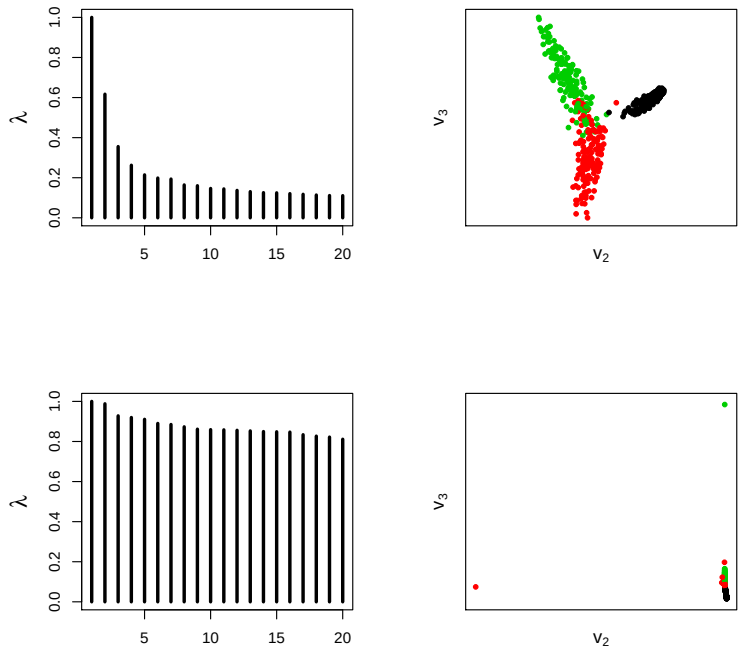

Figure 2: Spectral analysis of some zipcode data. Top: $h = 6$. Bottom: $h = 4$. The plots on the right show the second versus third eigenvector. The three colors correspond to the three digits 1, 2 and 3.

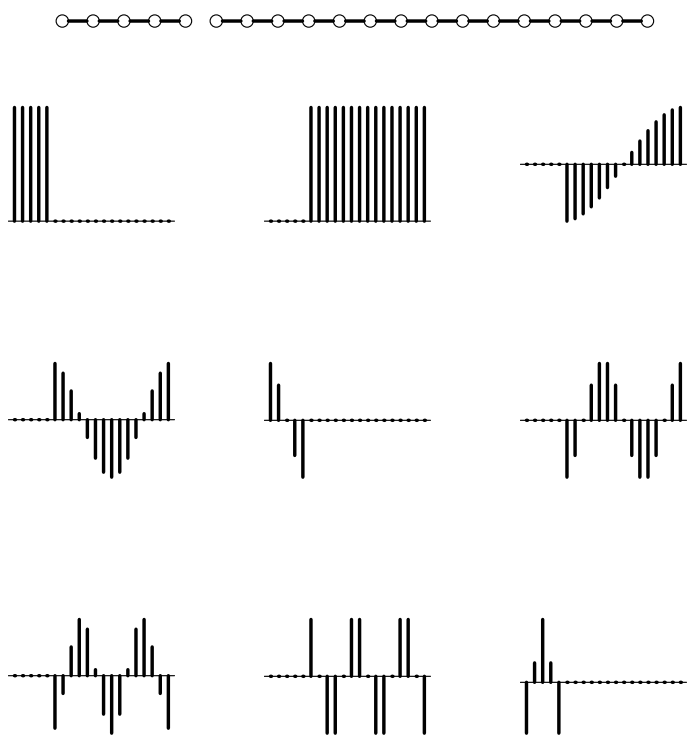

Figure 3: The top shows a simple graph. The remaining plots are the eigenvectors of the graph Laplacian. Note that the first two eigenvectors correspond to the two connected components of the graph.

2 Laplacian Eigenmaps

In the spectral clustering approach above, we extracted features that respected "clusters" corresponding to disconnected components of (level set of) the density. But even if there are no clear disconnected components, they nonetheless "respect" another intrinsic notion of structure: namely the manifold structure of the density. To see this, note that $f^T L f$ measures the smoothness of f relative to the graph. This means that the higher order eigenvectors generate a basis where the first few basis elements are smooth (with respect to the graph) and the later basis elements become more wiggly.

Example 5 *Figure 3 shows a graph and the corresponding eigenvectors. The two eigenvectors correspond two the connected components of the graph. The other eignvectors can be thought of as forming bases vectors within the connected components.*

Example 6 *Let us revisit the two concentric circles dataset in Figure 1. The top six eigenvectors can be seen to be increasingly less smooth.*

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

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

into k dimensions.

We can draw further intuitions from the basic properties of Rayleigh quotients and Laplacians. In particular, we have that the first nonzero eigenvector satisfies

$$y_1 \;=\; \arg\min y_1^T L y_1 \;=\; \arg\min \sum_{i,j} w_{ij} \left(y_{1i} - y_{1j}\right)^2 \tag{3}$$

$$\text{such that } \; y_1^T D y_1 = 1 \tag{4}$$

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,

$$f^T L f \approx \int_M \|\nabla f(x)\|^2 \, d_M(x) \tag{5}$$

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

As before, we could use the standard Laplacian $L = D - W$, or the **symmetrized Laplacian** $\mathcal{L} = D^{-1/2} W D^{-1/2}$ or **random walk Laplacian** $\mathcal{L} = D^{-1} W$. These are all very similar.

The random walk Laplacian $\mathcal{L}$ has a nice probabilistic interpretation (Coifman, Lafon, Lee 2006). Consider a Markov chain on $X_1, \ldots, X_n$ where we jump from X_i to X_j with probability

$$\mathbb{P}(X_i \to X_j) = \mathcal{L}(i,j) = \frac{K_h(X_i, X_j)}{\sum_s K_h(X_i, X_s)}.$$

The random walk Laplacian $\mathcal{L}(i,j)$ captures how easy it is to move from X_i to X_j. This Markov chain is a discrete version of a continuous Markov chain with transition probability:

$$P(x \to A) = \frac{\int_A K_h(x,y) dP(y)}{\int K_h(x,y) dP(y)}.$$

The corresponding linear operator $\widehat{A} : f \to \widetilde{f}$ is

$$(\widehat{A} f)(i) = \frac{\sum_j f(j) K_h(X_i, X_j)}{\sum_j K_h(X_i, X_j)}$$

is in turn an estimate of the population linear operator $A : f \to \widetilde{f}$ where

$$Af = \frac{\int_A f(y) K_h(x,y) dP(y)}{\int K_h(x,y) dP(y)}.$$

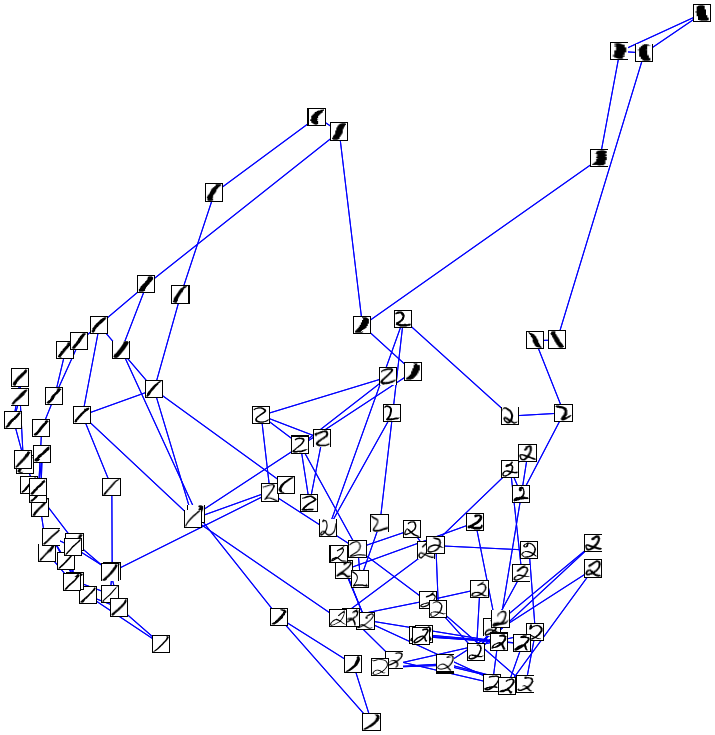

Figure 4: A portion of the similarity graph for actual scanned digits (1s and 2s), projected to two dimensions using Laplacian eigenmaps. Each image is a point in $\mathbb{R}^{256}$, as a 16×16 pixel image; the graph suggests the data has lower dimensional "manifold" structure

Given the form of this linear operator, it can be seen that the lower order eigenvectors of $\mathcal{L}$ are vectors that are smooth relative to the density P. Thus, projecting onto the first few eigenvectors parameterizes in terms of closeness with respect to the underlying density.

3 Reconstruction Error

Another approach to feature learning would be to specify coordinates in a different representation system, ideally in a near lossless fashion. We can capture this via a reconstruction error that arises from reconstructing the input given these new coordinates. Another way to view this is that we have a simpler space, and the projection of any input onto this simpler space is its new representation, and the projection error is simply the reconstruction error.

The simplest setting of this is where the simple space consists of linear manifolds, and the resulting representation learning approach is Principal components analysis (PCA).

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

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

where $\mu = \mathbb{E}(X)$. 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 7 *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 8 *We have:*

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

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 $\beta_{ij} = \langle X_i - \overline{X}, e_j \rangle$.

Example 9 *Figure 5 shows a synthetic two-dimensional data set together with the first principal component.*

Example 10 *Figure 6 shows some handwritten digits. The eigenvalues and the first few eigenfunctions are shown in Figures 7 and 8. A few digits and their low-dimensional reconstructions are shown in Figure 9.*

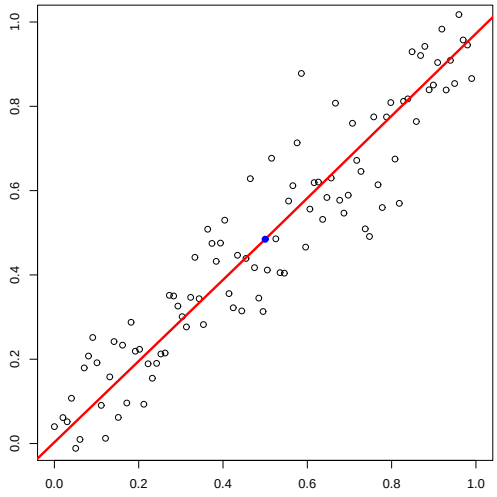

Figure 5: First principal component (red line) in a simple two dimensional example.

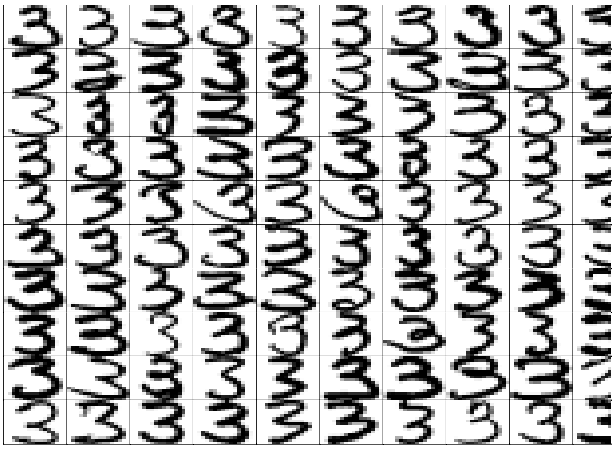

Figure 6: Handwritten digits.

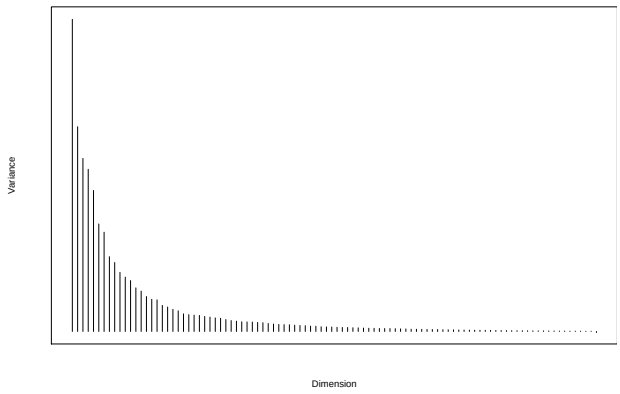

Figure 7: Digits data: eigenvalues

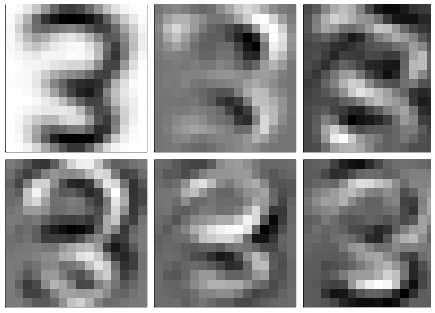

Figure 8: Digits: mean and eigenvectors

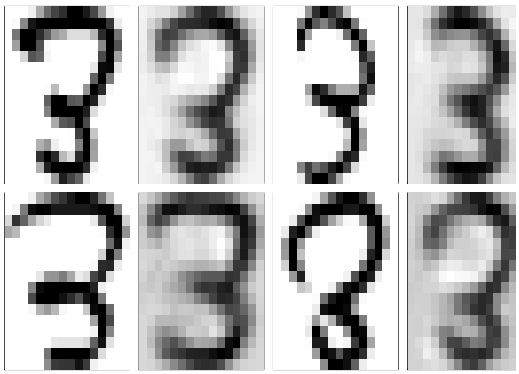

Figure 9: Digits data: Top: digits. Bottom: their reconstructions.

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.

3.1 Principal Curves and Manifolds

We can replace linear subspaces with more general manifolds.

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

$$R(f) = \mathbb{E}\left(\min_{z \in [0,1]^k} \|X - f(z)\|^2 \right). \tag{6}$$

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 $\mathbb{R}^d$.

Define the projection coordinates of any point $x \in \mathbb{R}^d$ onto f as $Z_f(x) = \arg\min_{z\in[0,1]^k} \|x - f(z)\|^2$, and the corresponding projection as $T_{\mathcal{F}}(x) = f(Z_f(x))$.

In (Hastie & Stuetzle, Principal Curves, 1989), they propose the *self-consistency constraint*:

$$\mathbb{E}[X|Z_f(X) = Z] = f(Z),$$

and propose an iterative algorithm to learn one-dimensional self-consistent principal curves by solving for the fixed point:

$$f^{(t)}(z) = \mathbb{E}[X|Z_{f^{(t-1)}}(X) = z].$$

Smola et al (2001) consider an RKHS version. They 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

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

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 11 *Figure 10 shows some data and four principal curves based on increasing degrees of regularization.*

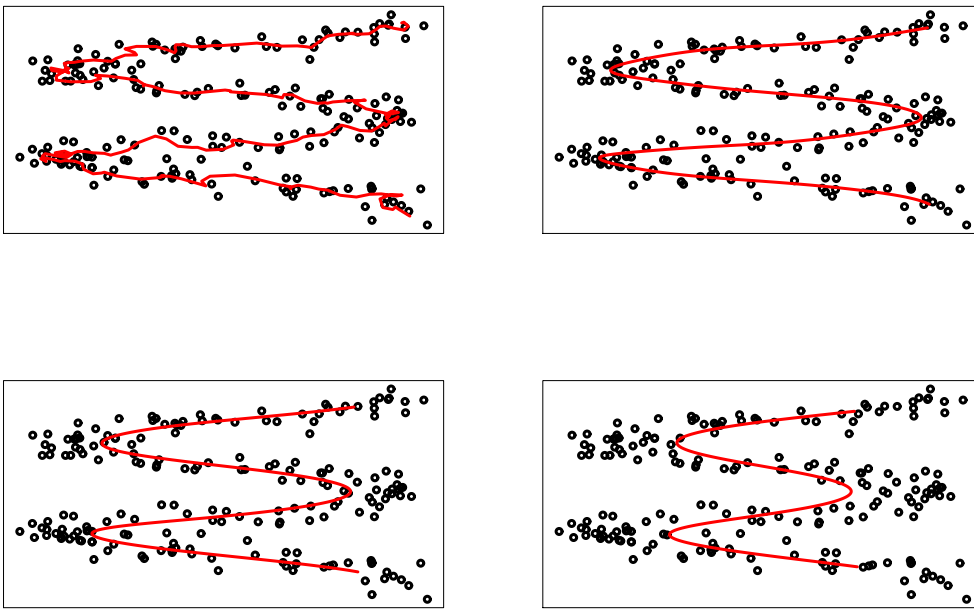

Figure 10: Principal curve with increasing amounts of regularization.

3.2 Autoencoders

The manifolds in the previous sections were represented as $\{f(z) : z \in \mathbb{R}^k\}$. The caveat with these is that we need to solve a separate optimization problem to find the coordinates z corresponding to *each data point*. This is not scalable to large datasets. A simpler approach would be to have a fixed function, called an encoder $z = g(x)$, that corresponds to the projection onto the manifold and learn this as a function, simultaneously with the decoder $x = f(z)$ as in the earlier principal manifolds section. This results in what is known as an auto-encoder, since we are learning both an encoder and a decoder so that $x \approx f \circ g(x)$.

Let's first consider the linear setting, with linear encoders $E(X) = WX$, and linear decoders $D(Z) = MZ$ where $Z \in \mathbb{R}^K$ and $M \in \mathbb{R}^{d \times K}$. Then, the composition of the encoder and decoder results in a so-called auto-encoder that computes $\widehat{X} = D(E(X)) = MWX$. We would ideally like this to as close to X as possible. Suppose we care about the expected ℓ_2 error, so that we wish the optimal auto-encoder matrices to solve for:

$$\inf_{W,M} \mathbb{E}[\|X - MWX\|^2].$$

[?] showed that the optimal encoder matrix $W = CU$, and $M = U^T C^{-1}$, where $U = [u_1, \ldots, u_k]$ is the matrix obtained by row-stacking the top K eigenvectors of $\Sigma = \mathrm{Cov}(X)$, and $C \in \mathbb{R}^{K \times K}$ is any invertible orthonormal matrix. Thus, the optimal linear auto-encoder results in the PCA transform.

Let us see why this might be the case. Suppose we fix the encoder $E(X) = WX$, for some orthonormal W. Given such an encoder, the optimal decoder with respect to the ℓ_2 error can be written as

$$D(Z) = \arg\inf_X \|E(X) - Z\|^2 =: \|WX - Z\|^2,$$

which is an under-determined problem for $k < p$, Under the additonal restriction that we desire for smallest ℓ_2 norm $\widehat{X}$ such that $W\widehat{X} = Z$, we recover $D(Z) = \widehat{X} = (W^TW)^+W^TZ = W^TZ$, under the assumption that the encoder matrix W is orthonormal.

Similarly, suppose we fix the decoder as $D(Z) = W^TZ$ for some orthonormal matrix W. The optimal encoder with respect to the ℓ_2 error is then given as:

$$E(X) = \arg\inf_Z \|X - D(Z)\| =: \|X - W^TZ\|,$$

which can be seen to be given by $E(X) = (WW^T)^{-1}WX = WX$, under the assumption that the decoder matrix W is orthonormal.

We thus see that even if we just specify a linear decoder or linear encoder, the optimal encoder or decoder respectively is also linear, and under the assumption that one of the linear transformations is orthonormal, the optimal other transformation is simply its transpose.

Thus, the overall ℓ_2 error of a linear auto-encoder is given by:

$$\mathbb{E}[X - W^TWX] = \mathbb{E}[\|X - \sum_{j=1}^{K} w_j(w_j^TX)\|^2,$$

so that we get the following objective comprising the minimum mean-squared error between X and its projection onto rows of W:

$$\min_W \mathbb{E}[\|X - W^TWX\|^2$$
$$\text{s.t. } WW^T = I.$$

This can be seen to be equivalent to the earlier PCA objective.

More generally, we can train non-linear auto-encoders f, g so that:

$$\min_{f,g} \mathbb{E}\|X - f(g(X))\|_2^2,$$

which are called non-linear or simply auto-encoders, and where the families for the encoders and decoders are typically neural networks. When using such highly flexible families, one pitfall is that they can simply learn invertible maps f, g so that $fog = \text{Id}$. To prevent this, one can use $k \ll d$, and also have *denoising* auto-encoders that have to reconstruct the input given noisy versions of the input:

$$\min_{f,g} \mathbb{E}\|X - f(g(n(X)))\|_2^2,$$

where $n(X)$ is a noisy version of the input. Another approach for regularization would be variational auto-encoders that we studied in the context of deep density estimation.

But notwithstanding such regularizations, the non-linear setting however is horribly ill-posed: for any invertible function $h : \mathbb{R}^k \mapsto \mathbb{R}^k$, we have that $f(g(X)) = (foh)(h^{-1}og)(X)$.

4 Information

We can also ask for features that preserve the most information. But how do we quantify information? One crude approach might be to use variance. Let us consider the linear setting.

Let $X \in \mathbb{R}^d$ be a zero mean random vector, and consider the linear encoder $E(X) = WX =: Z$ which maps $X \in \mathbb{R}^d$ to the latent $Z = WX \in \mathbb{R}^k$. Setting the objective to be the sum of individual variances of the latent variables, subject to orthogonality of the rows of W, we get theobjective:

$$\begin{aligned} \max_W \sum_{j=1}^K \mathbb{E}(Z_j^2) &= \mathbb{E}\|WX\|^2 \\ \text{s.t. } WW^T &= I, \end{aligned}$$

where the constraint states that the rows of W are orthonormal. This can be seen to be the PCA objective, and as before solved by an eigendecomposition of $\Sigma = \mathbb{E}[XX^T]$. To see this, expand the mean-squared error objective as:

$$\begin{aligned} &\mathbb{E}[\|X\|^2] - 2\mathbb{E}[X^T W^T WX] + \mathbb{E}[X^T W^T WW^T WX] \\ &= \mathbb{E}[\|X\|^2] - \mathbb{E}[X^T W^T WX] \\ &= \mathbb{E}[\|X\|^2] - \mathbb{E}[\|WX\|^2] \end{aligned}$$

where the second equality is due to orthonormality of rows of W, where $WW^T = I$.

What about the non-linear case with $Z_j = g_j(X)$, for $j \in [k]$? We could maximize the variance $\sum_j \text{Var}(g_j(X))$, but this is ill-posed without additional constraints on $\{g_j\}$. In the absence of such constraints, note that the variance can be made very large by scaling g_j. Note that even in the linear case above, we imposed orthonormality of the different linear g_j, and it is not exactly clear how to translate that to the non-linear case.

Let $X \sim \mathcal{N}(0, \Sigma)$. Then, $Z = WX \sim \mathcal{N}(0, W\Sigma W^T)$. Given Z, we could then decode X via the conditional distribution $X|Z = z$, which can be seen to be Gaussian as well. We could thus ask for an encoding with minimum "decoding" uncertainty as specified by the conditional entropy $H(X|Z)$:

$$\min_W H(X|Z := WX).$$

It can be shown [**?**] that the optimal $W = CU$ as earlier. This can also be alternatively stated as maximizing mutual information between X and Z since $I(X, Z) = H(X) - H(X|Z)$. Thus, an alternative characterization of PCA is as a linear map that maximizes mutual information with original input, but under the strong proviso that the input is Gaussian distributed. When implemented for general non-linear maps, this is known as the Infomax, and together with approximations such as InfoNCE, have been some of the pre-cursors of modern self-supervised representation learning.

References

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. Density modeling of images using a generalized normalization transformation. *arXiv preprint arXiv:1511.06281*, 2015.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Scott Saobing Chen and Ramesh A Gopinath. Gaussianization. In *Advances in neural information processing systems*, pages 423–429, 2001.

David Inouye and Pradeep Ravikumar. Deep density destructors. In *International Conference on Machine Learning*, pages 2167–2175, 2018.